

UNIVERSITY OF MICHIGAN

# Automatic Testing of the Trigger Data Serializer ASIC for the Upgrade of the ATLAS Muon Spectrometer

by

Reid F. Pinkham

An honors thesis submitted in partial fulfillment for the  
degree of Bachelor of Science

under the supervision of

Junjie Zhu

Department of Physics

April 2017





*“Lego my Eggo like Prego”*

UNIVERSITY OF MICHIGAN

# *Abstract*

Department of Physics

Bachelor of Science

by Reid F. Pinkham

The Trigger Data Serializer (TDS) is a custom designed Application Specific Integrated Circuit (ASIC) designed at the University of Michigan to be used on the ATLAS New Small Wheel (NSW) detector. The TDS is a central hub of the NSW trigger system. It prepares the trigger data for both pad and strip detectors, performs pad-strip matching, and serializes the matched strip data to other circuits on the rim of the NSW. In total, 6000 TDS chips will be produced. As part of the TDS' initial production run, a test platform was developed to verify the functionality of each chip before being sent to users. The test platform consisted of multiple FPGA evaluation boards with custom designed mezzanine boards to hold the TDS chip during testing and control software running on a local computer. Of the initial run of 200 chips, 161 chips were tested with the automatic setup of which 158 passed. Detailed description of the TDS and automatic test fixture can be found in this thesis.

## *Acknowledgements*

I would like to thank Professors Junjie Zhu, Tom Schwarz, and Bing Zhou for their continued support and guidance while I have worked with the Michigan ATLAS group. I would also like to thank Jinhong Wang, Xueye Hu, and Liang Guan who are the great post-docs who taught me firmware development and introduced me to the world of logic design. Lastly, I would like to thank all of my friends and family for listening to my stories of the adventures I have gone on and experiences I've had as a result of my work in the Physics department.



# Contents

|                                                |             |
|------------------------------------------------|-------------|
| <b>Abstract</b>                                | <b>iv</b>   |
| <b>Acknowledgements</b>                        | <b>v</b>    |
| <b>List of Figures</b>                         | <b>ix</b>   |
| <b>List of Tables</b>                          | <b>xi</b>   |
| <b>Abbreviations</b>                           | <b>xiii</b> |
| <b>1 Introduction and Background</b>           | <b>1</b>    |
| 1.1 CERN and the ATLAS detector . . . . .      | 1           |
| 1.2 The New Small Wheel (NSW) . . . . .        | 2           |
| 1.2.1 Motivation . . . . .                     | 3           |
| 1.2.2 Detector Technology . . . . .            | 3           |
| 1.2.3 NSW Electronics . . . . .                | 4           |
| 1.2.4 sTGC Trigger Path . . . . .              | 7           |
| <b>2 The TDS ASIC</b>                          | <b>9</b>    |
| 2.1 Design Requirements . . . . .              | 9           |
| 2.2 Physical Description . . . . .             | 10          |
| 2.3 Strip Mode . . . . .                       | 10          |
| 2.4 PAD Mode . . . . .                         | 13          |
| 2.5 Built In Test Functionality . . . . .      | 13          |
| <b>3 Test Setup</b>                            | <b>15</b>   |
| 3.1 Methodology . . . . .                      | 15          |
| 3.2 Firmware Development . . . . .             | 16          |
| 3.2.1 TDS Interface and Data Checker . . . . . | 16          |
| 3.2.2 I <sup>2</sup> C Interface . . . . .     | 19          |
| 3.2.3 Ethernet Interface . . . . .             | 20          |
| 3.3 Software Development . . . . .             | 23          |
| 3.4 Hardware . . . . .                         | 25          |
| <b>4 Chip Testing Procedure</b>                | <b>29</b>   |
| 4.1 List of Tests . . . . .                    | 29          |
| 4.1.1 I <sup>2</sup> C . . . . .               | 29          |



# List of Figures

|     |                                                                                                                                                                                                                                                              |    |
|-----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 1.1 | The muon spectrometer system of the ATLAS detector. Modified from diagram in [4]. . . . .                                                                                                                                                                    | 2  |
| 1.2 | The current trigger system for the Muon Spectrometer accepts high $p_T$ muons. False triggers are caused by background particles originating from the end-cap region which appear to come from the interaction point. Modified from a figure in [5]. . . . . | 4  |
| 1.3 | Diagram of the NSW. There are sixteen sections of detectors around the wheel. Each section has eight layers of sTGC detectors and eight layers of MM detectors for both triggering and tracking. [6] . . . . .                                               | 5  |
| 1.4 | Cross section of an sTGC layer. [6] . . . . .                                                                                                                                                                                                                | 5  |
| 1.5 | The pads are used to select the relevant strips to be readout for a candidate track. [6] . . . . .                                                                                                                                                           | 6  |
| 1.6 | The complete NSW Electronics data flow diagram. [6] . . . . .                                                                                                                                                                                                | 7  |
| 1.7 | The sTGC Trigger path. [6] . . . . .                                                                                                                                                                                                                         | 8  |
| 2.1 | Layout of the TDSV2 ASIC. . . . .                                                                                                                                                                                                                            | 10 |
| 2.2 | Block diagram of the strip-TDS. . . . .                                                                                                                                                                                                                      | 11 |
| 2.3 | VMM charge output format and timing. . . . .                                                                                                                                                                                                                 | 11 |
| 2.4 | 640 Mbps signals from the pad-trigger board. . . . .                                                                                                                                                                                                         | 12 |
| 2.5 | Data and NULL packet structures. . . . .                                                                                                                                                                                                                     | 12 |
| 2.6 | Pad-TDS block diagram. . . . .                                                                                                                                                                                                                               | 13 |
| 3.1 | Block diagram of the firmware. . . . .                                                                                                                                                                                                                       | 17 |
| 3.2 | Internal structure of the Ethernet core. . . . .                                                                                                                                                                                                             | 21 |
| 3.3 | Screenshot of the GUI during a test. . . . .                                                                                                                                                                                                                 | 24 |
| 3.4 | Top layout of the TDS mezzanine card. . . . .                                                                                                                                                                                                                | 25 |
| 3.5 | FPGA and mezzanine board with BGA socket. . . . .                                                                                                                                                                                                            | 26 |
| 3.6 | TDS placed in BGA socket with QR code. . . . .                                                                                                                                                                                                               | 27 |
| 4.1 | A sample of TDS chips as packaged from the factory. . . . .                                                                                                                                                                                                  | 32 |
| 4.2 | Illustration of preparing setup for testing . . . . .                                                                                                                                                                                                        | 32 |



# List of Tables

|     |                                                           |    |
|-----|-----------------------------------------------------------|----|
| 3.1 | General Ethernet Packet Structure . . . . .               | 22 |
| 3.2 | Commands supported for PC to FPGA communication . . . . . | 22 |
| 3.3 | Commands supported for FPGA to PC communication . . . . . | 23 |



# Abbreviations

|              |                                                                                                                                     |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------|
| <b>ARP</b>   | <b>A</b> ddress <b>R</b> esolution <b>P</b> rotocol                                                                                 |
| <b>ART</b>   | <b>A</b> ddress in <b>R</b> eal <b>T</b> ime                                                                                        |
| <b>ASD</b>   | <b>A</b> mplifier <b>S</b> haper <b>D</b> iscriminator                                                                              |
| <b>ASIC</b>  | <b>A</b> pplication <b>S</b> pecific <b>I</b> ntegrated <b>C</b> ircuit                                                             |
| <b>ATLAS</b> | <b>A</b> <b>T</b> oroidal <b>L</b> H <b>C</b> <b>A</b> pparatu <b>S</b>                                                             |
| <b>BC</b>    | <b>B</b> unch <b>C</b> rossing                                                                                                      |
| <b>BCID</b>  | <b>B</b> unch <b>C</b> rossing <b>ID</b>                                                                                            |
| <b>BGA</b>   | <b>B</b> all <b>G</b> rid <b>A</b> rray                                                                                             |
| <b>BRAM</b>  | <b>B</b> lock <b>R</b> andom- <b>A</b> ccess- <b>M</b> emory                                                                        |
| <b>CERN</b>  | <b>C</b> onseil <b>E</b> uropen pour la <b>R</b> echerche <b>N</b> uclaire<br><i>The European Organization for Nuclear Research</i> |
| <b>CMS</b>   | <b>C</b> ompact <b>M</b> uon <b>S</b> pectrometer                                                                                   |
| <b>COTS</b>  | <b>C</b> ommercial <b>O</b> ff the <b>S</b> helf                                                                                    |
| <b>DDR</b>   | <b>D</b> ouble <b>D</b> ata <b>R</b> ate                                                                                            |
| <b>FEB</b>   | <b>F</b> ront <b>E</b> nd <b>B</b> oard                                                                                             |
| <b>FELIX</b> | <b>F</b> ront <b>E</b> nd <b>L</b> ink <b>E</b> xchange                                                                             |
| <b>FIFO</b>  | <b>F</b> irst <b>I</b> n <b>F</b> irst <b>O</b> ut buffer                                                                           |
| <b>FMC</b>   | <b>F</b> PGA <b>M</b> ezzanine <b>C</b> ard                                                                                         |
| <b>FPGA</b>  | <b>F</b> ield <b>P</b> rogrammable <b>G</b> ate <b>A</b> rray                                                                       |
| <b>Gbps</b>  | <b>G</b> igabits <b>p</b> er <b>s</b> econd                                                                                         |
| <b>GBT</b>   | <b>G</b> iga <b>B</b> it <b>T</b> ranciever <b>A</b> SiC                                                                            |
| <b>GUI</b>   | <b>G</b> raphical <b>U</b> ser <b>I</b> nterface                                                                                    |
| <b>IP</b>    | <b>I</b> nteraction <b>P</b> oint                                                                                                   |
| <b>LUT</b>   | <b>L</b> ookup <b>T</b> able                                                                                                        |
| <b>MAC</b>   | <b>M</b> edia <b>A</b> ccess <b>C</b> ontrol                                                                                        |

|                       |                                                                                    |
|-----------------------|------------------------------------------------------------------------------------|
| <b>MDT</b>            | <b>M</b> onitored <b>D</b> rift <b>T</b> ube                                       |
| <b>Micromegas, MM</b> | <b>M</b> icro <b>M</b> esh <b>G</b> aseous <b>S</b> tructure detectors             |
| <b>MSB</b>            | <b>M</b> ost <b>S</b> ignificant <b>B</b> it                                       |
| <b>NSW</b>            | <b>N</b> ew <b>S</b> mall <b>W</b> heel                                            |
| <b>L1DDC</b>          | <b>L</b> evel- <b>1</b> <b>D</b> ata <b>D</b> river <b>C</b> ard                   |
| <b>LHC</b>            | <b>L</b> arge <b>H</b> adron <b>C</b> ollider                                      |
| <b>PCB</b>            | <b>P</b> rinted <b>C</b> ircuit <b>B</b> oard                                      |
| <b>PRBS</b>           | <b>P</b> suedo- <b>R</b> andom <b>B</b> inary <b>S</b> equence                     |
| <b>ROC</b>            | <b>R</b> eadout <b>A</b> SiC                                                       |
| <b>SGMII</b>          | <b>S</b> erial <b>G</b> igabit <b>M</b> edia <b>I</b> ndependent <b>I</b> nterface |
| <b>SCA</b>            | <b>S</b> low <b>C</b> ontrol <b>A</b> SiC                                          |
| <b>SW</b>             | <b>S</b> mall <b>W</b> heel                                                        |
| <b>sTGC</b>           | <b>s</b> mall-strip <b>T</b> hin <b>G</b> ap <b>C</b> hamber                       |
| <b>TDS</b>            | <b>T</b> rigger <b>D</b> ata <b>S</b> erializer                                    |
| <b>TEMAC</b>          | <b>T</b> ri- <b>M</b> ode <b>E</b> thernet <b>M</b> AC                             |
| <b>TGC</b>            | <b>T</b> hin <b>G</b> ap <b>C</b> hamber                                           |
| <b>TMR</b>            | <b>T</b> riple <b>M</b> odule <b>R</b> edundancy                                   |
| <b>UDP</b>            | <b>U</b> ser <b>D</b> atagram <b>P</b> rotocol                                     |

# Chapter 1

## Introduction and Background

This thesis covers the work completed at the University of Michigan to create an automatic test fixture for the Trigger Data Serializer (TDS). This work is done as part of the ATLAS collaboration, a high energy physics experiment at CERN.

### 1.1 CERN and the ATLAS detector

The European Organization for Nuclear Research (CERN) is an international physics laboratory which was founded in 1954 in Geneva, Switzerland, and located on the Swiss-French border. In its history, CERN has been home to many large experiments and discoveries. Most recently, CERN is home to the Large Hadron Collider (LHC). The collaborations operating detectors at the LHC are interested in answering physics' most fundamental questions such as the origin of mass, extra dimensions, dark matter, and the matter-antimatter asymmetry.

The LHC is the world's largest and most powerful particle accelerator [1]. The accelerator is located underground at an average depth of 100 meters in a 27 kilometer circular tunnel. The accelerator collides two proton beams at multiple locations around the ring to achieve collision energies of up to 14 TeV. At each of the collision points, there are ongoing experiments which use detectors to study various aspects of the proton collisions. There are two large general purpose detectors on the ring, the ATLAS (A Toroidal LHC ApparatuS) and the Compact Muon Spectrometer (CMS). Both of these detectors were built by large international collaborations involving thousands of physicists and engineers from around the world. In 2012, both ATLAS and CMS discovered the Higgs boson [2][3] which lead to the 2013 Nobel prize in physics.

The ATLAS detector is among the largest and most intricate machines ever built. It is housed in an underground cavern 92 m below the surface and is 46 m long, and 24 m in diameter weighing over 7000 tonnes. The detector was built by the ATLAS collaboration alongside the LHC's construction and finished in 2008. Since it's completion, the detector has been operating at the LHC collecting data from the proton-proton collisions at energies up to 13 TeV.

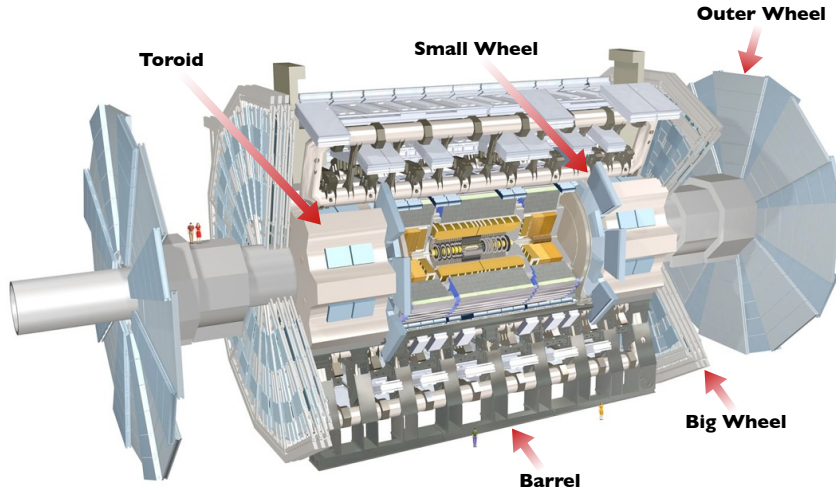


FIGURE 1.1: The muon spectrometer system of the ATLAS detector. Modified from diagram in [4].

The ATLAS detector is built of many concentric layers of particle detectors. These include an inner tracker around the interaction point (IP), calorimeters for measuring the energy of hadrons and electrons, and a muon spectrometer to measure the energy and path of the highly penetrating muons. Together, these systems collect snapshots of the collisions which are then used to analyze each event in order to better understand the fundamental physics of the subatomic world.

In the future, the LHC will be upgraded to greatly increase the collision rate. These upgrades will allow the experiments to probe deeper in the effort to search for new phenomena. To handle the unprecedented energy level and collision rate, the ATLAS detector will need to be upgraded. The first step of this upgrade is called the phase-I upgrade. During this upgrade, a major portion of the muon spectrometer, the Small Wheel (SW) will be replaced.

## 1.2 The New Small Wheel (NSW)

The muon spectrometer consists of a central tracking region in the barrel and an end-cap system to cover the forward regions. The end-cap region consists of three major portions, the small wheel, the big wheel, and the outer wheel as shown in figure 1.1. The SW will be replaced as part of the phase-I upgrade by the NSW. The NSW will be used as part of both the triggering and precision tracking system.

### 1.2.1 Motivation

The current small wheel has Monitored Drift Tubes (MDT) and Cathode Strip Chambers (CSC) used for precision measurement. The small wheel is the innermost part of the muon spectrometer and is not currently used for triggering. The motivation for replacing the current small wheel is twofold. The first motivation is to use detector technologies which can better handle the increased rates of the high-luminosity LHC. The current MDT detectors may saturate at the highest possible hit rates. Extrapolating from current rates, the high luminosity rate would be greater than the MDTs can handle. This would cause a random loss of data in that region which would hinder reconstruction efforts. The new technology used is designed to accommodate a muon rate of at least  $15 \text{ kHz/cm}^{-2}$ . These new technologies are described further in 1.2.2.

The second motivation is to add an additional trigger component to the end-cap muon system. Proton-proton collision which produce highly energetic muons indicate an event of interest and must be saved. To identify these events at the first trigger level in the current system, angular measurements made by the big wheel are used to identify muons which originate from the interaction point (figure 1.2). Unfortunately, using only the measurement at the big wheel for the level-1 trigger allows background particles originating from within the end-cap toroid to cause triggers. These background particles take a path similar to that of a muon coming from the interaction point and will activate the trigger. These fake triggers account for the majority of the triggers coming from the end-cap system.

To solve this problem, an angular measurement of the muons will be made by the small wheel and used in the trigger decision. The NSW will provide an angular resolution of 1 mrad to the trigger. This information and the measurement at the big wheel will be correlated in the trigger decision to eliminate the spurious triggers.

### 1.2.2 Detector Technology

In order to allow for the increased hit rate and the needed angular resolution for the trigger, both new detector technologies and accompanying electronics are needed. The two detector technologies chosen are small Strip Thin Gap Chambers (sTGC) and MicroMesh Gaseous structure detectors (Micromegas, MM) [6]. The sTGCs are a modification of the existing thin gap chambers to provide the needed precision. Both detector technologies are designed to be used in both precision tracking and triggering. The sTGCs have a higher rate capability than the Micromegas, however they have a lower spatial resolution. By using both detector technologies, the NSW will have excellent spacial resolution with the Micromegas, and the ability to accommodate a high trigger rate with the sTGCs.

A cross section of the sTGC detector is shown in figure 1.4. The sTGCs consist of a grid of  $50 \mu\text{m}$  gold-plated tungsten wires with 1.8 mm pitch placed between two cathode planes which are 1.4 mm from the wire plane. The cathode planes are graphite-epoxy sprayed onto a  $100 \mu\text{m}$  thick G10 plane. On one cathode plane, there are strips which run perpendicular to the wires with a 3.2 mm pitch, and on the other plane there are 8 cm x 8

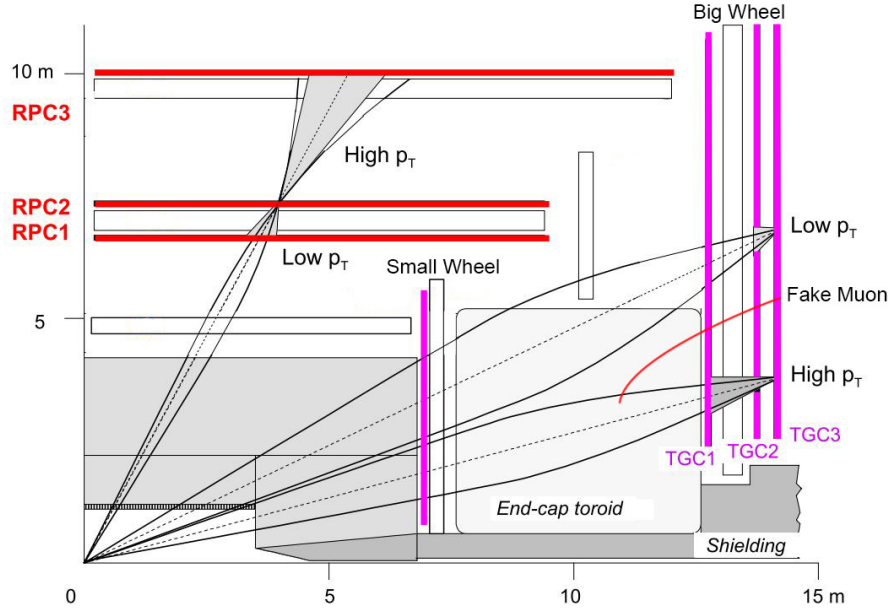


FIGURE 1.2: The current trigger system for the Muon Spectrometer accepts high  $p_T$  muons. False triggers are caused by background particles originating from the end-cap region which appear to come from the interaction point. Modified from a figure in [5].

cm pads. The pads are used for initial track triggering, and to narrow the ranges of strips that are read out for online muon candidate selection. The strips measure the bending coordinate of a track, and the wires are used to measure the azimuthal coordinate. After passing a detector-level trigger, the charge of all wires, strips, and pads are readout for offline track reconstruction.

The orientation of the sTGC layers is such that the corner of one pad corresponds to the center of a pad on the next layer. When a muon passes through the detector, using just the pad signals the region of strips which may have charge is reduced to a band of roughly 13 strips. Only this small region of strips needs to be readout for the trigger.

### 1.2.3 NSW Electronics

New electronics have to be designed and constructed for the NSW. The new electronics will incorporate the next generation of silicon and fiber optic technology to provide the necessary functionality. Electronics being designed for the NSW must operate through the high luminosity runs and for the duration of the ATLAS detector operation. This requirement provides an exceptional challenge for radiation tolerance.

Because of the harsh radiation environment, the front-end electronics must be Application Specific Integrated Circuits (ASICs) which are custom designed to operate in high radiation environments. Electronics located at the rim level and in the back-end, they can consist of commercial off the shelf (COTS) components, but must be certified for the radiation levels expected.

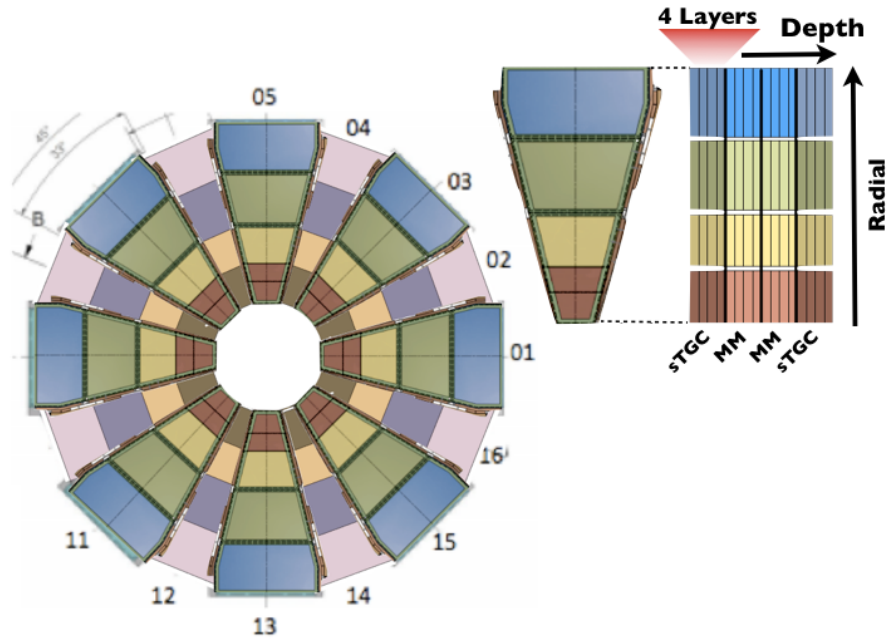


FIGURE 1.3: Diagram of the NSW. There are sixteen sections of detectors around the wheel. Each section has eight layers of sTGC detectors and eight layers of MM detectors for both triggering and tracking. [6]

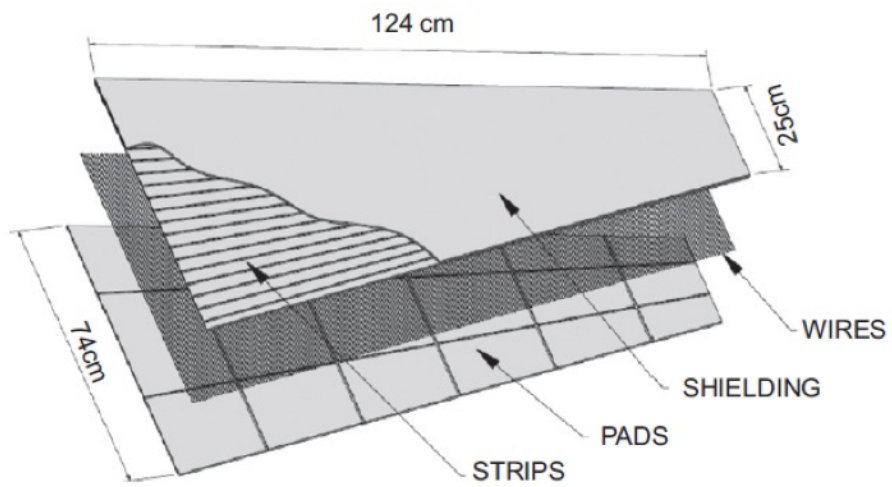


FIGURE 1.4: Cross section of an sTGC layer. [6]

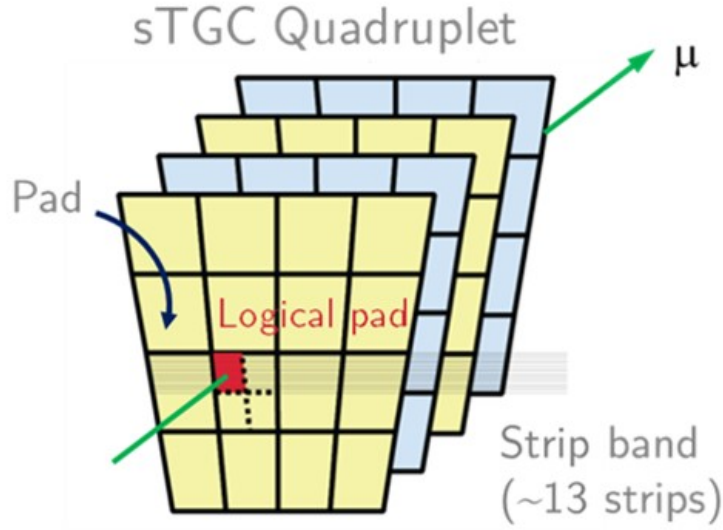


FIGURE 1.5: The pads are used to select the relevant strips to be readout for a candidate track. [6]

An overview of the NSW electronics is shown in figure 1.6. The front-end boards are located on each detector layer, and are responsible for digitizing the analog signals from the gaseous detectors, sending the needed information for a trigger, buffering all information while a trigger decision is made, then sending all data for an accepted event.

All of the analog readout is done with a custom ASIC called the VMM [7]. This is a large chip designed by the Brookhaven National Lab electronics group specifically for this application. Each VMM can readout up to 64 channels. Accompanying the VMM on the front-end boards are two additional radiation hard ASICs, the Slow Control ASIC (SCA) [8] for configuring the chips and the Readout ASIC (ROC) [9]. The front-end boards for the sTGC detector will have an additional ASIC, the Trigger Data Serializer (TDS).

There are different paths for trigger and readout data leaving the front-end boards. For the Micromegas detectors, the data is readout by the Address in Real Time (ART) ASIC located on separate boards on the chamber [10]. This chip performs the required roadmapping and passes the accepted data to the trigger processor located in a cavern adjacent to the ATLAS detector. For the sTGCs, pad trigger data is first passed to the pad trigger board located on the rim [11]. Information on the required strips needed is passed back to the front-end where the required strip information is streamed out through the signal packet router located on the NSW rim [12], and then to the trigger processor. Both the Micromegas and sTGC trigger path has a latency requirement of roughly  $1 \mu\text{s}$ . To achieve this, both paths utilize multi-gigabit links over fiber.

The path for the precision tracking data is very similar between the Micromegas and sTGC detectors. The data is passed from the VMM to the ROC once a trigger has been accepted. Then, the data is passed out to the Gigabit Transceiver (GBT) ASIC [13] from which it is streamed to the data acquisition system called FELIX, or the Front End Link eXchange [14].



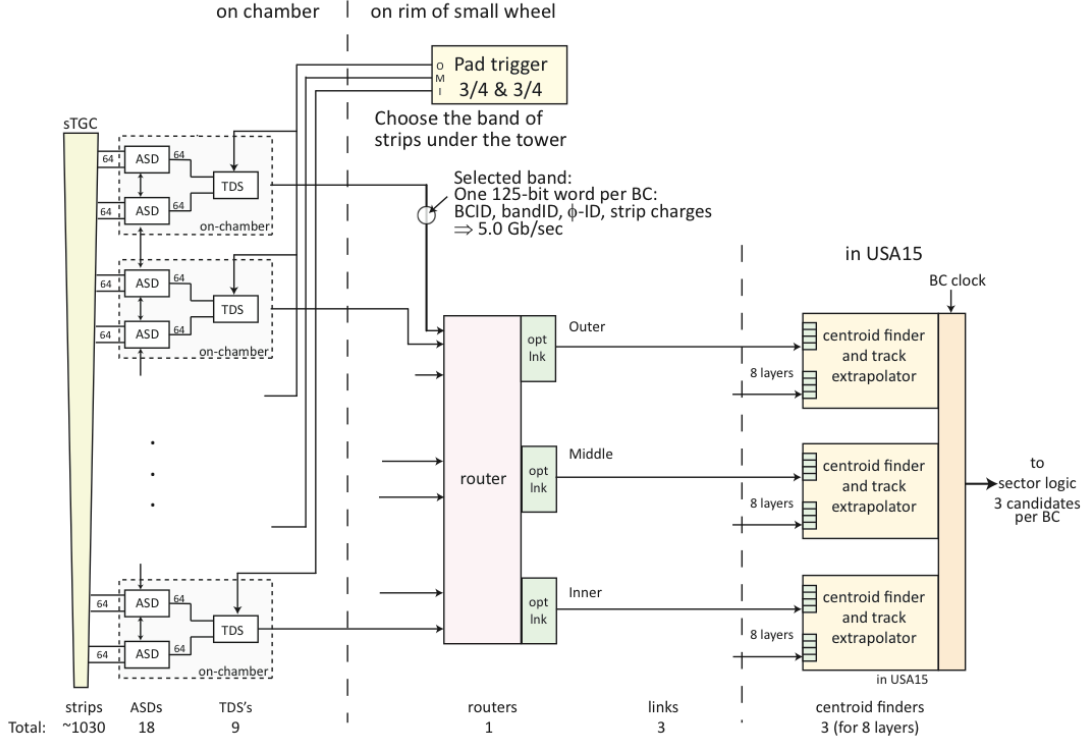


FIGURE 1.7: The sTGC Trigger path. [6]

over four 4.8 Gbps optical lines. Once the data is at the trigger processor, the data is analyzed to find centroids and do basic track reconstruction. The result of this analysis is then passed to the sector logic to be used as part of the overall trigger decision.

As stated above, the latency requirement between the collision and sector logic receiving the required information is roughly  $1 \mu\text{s}$ . In order to meet this requirement, the trigger electronics make use of multi-gigabit serial links and fixed latency algorithms. The output links from the TDS chips to both the pad trigger board and router are 4.8 Gbps.

## Chapter 2

# The TDS ASIC

The TDS chip was designed at the University of Michigan and is responsible for the preparation of trigger data coming from both the pads and strips. The chip operates in two different modes, the strip-TDS and pad-TDS, selectable with a hardware pin. In both modes, the TDS reads parallel digital input information from the VMM for each channel. In the pad-TDS mode, binary indications are passed to the TDS which indicate whether a pad fired or not. In strip-TDS mode, a 6-bit charge is read from the front-end VMM chip for each channel. The chip is implemented using the IBM 130 nm manufacturing process. This is the same process as used in the VMM chip so both chips can be manufactured at the same to reduce costs.

### 2.1 Design Requirements

*This section summarizes the contents of [16].*

The largest design challenges for the TDS chip are creating the radiation hard 4.8 Gbps serial links for output, meeting the latency requirements for each mode, efficiently handling the parallel input data, and operating in high radiation environments. In pad mode, the latency must be less than 100 ns, while in strip mode it must be less than 150 ns. To meet the radiation hardness requirement, triple module redundancy (TMR) is used throughout. The serializer was modified from another CERN group to use a different manufacturing process and take a 160 MHz clock input instead of 40 MHz [17].

The TDS chip design began in 2013 with the evaluation of the serializer components. The serializer used in the TDS is a modified version of the serializer from CERN's GBT chip. After this important part was verified, an initial version of the TDS was designed and fabricated in 2015. After thoroughly testing the functionality of this chip, a second version (TDSV2) was designed and manufactured in 2016. The TDSV2 was also thoroughly tested and contains all of the needed functionality.

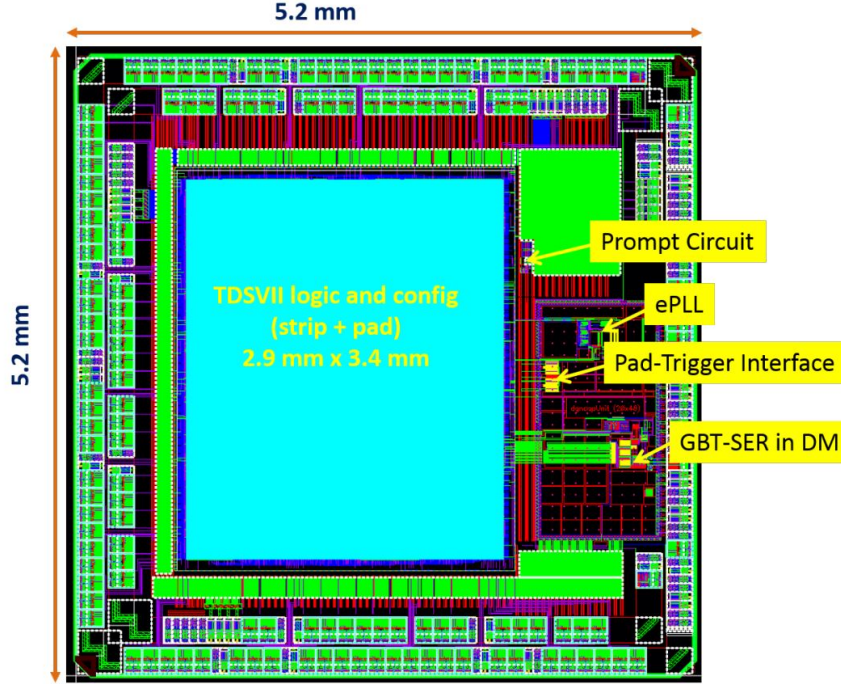


FIGURE 2.1: Layout of the TDSV2 ASIC.

## 2.2 Physical Description

The TDSV2 ASIC is designed to occupy a 5.2x5.2 mm die area. The designed layout of the TDS chip is shown in figure 2.1. As seen in the image, a large portion of the chip die is occupied by the interfacing pads. The packaging for the chip is a 2 cm per side 400-pin BGA package.

## 2.3 Strip Mode

In strip mode, the strip-TDS takes 128 inputs from multiple VMMs with the addition of two upper and lower strip inputs. These extra four strip inputs are needed in the case that a charge centroid lies on the boundary between TDS domains. The overall functionality of this mode is to buffer the strip inputs from the VMMs until they are selected for trigger output. When that happens, the charge is packaged into a larger packet along with the corresponding band/phi ID and Bunch Crossing ID (BCID) and sent out over the serial output. To accomplish this, the strip-TDS is divided into three major components: VMM interface, preprocessor, and serialization. A block diagram of the strip-TDS is shown in figure 2.2.

The VMM interface interprets the charge coming from the VMMs. The VMM outputs a 6-bit charge for each channel. Each VMM can output up to 64 channels, and thus each TDS operating in strip mode is typically responsible for taking input from two VMMs. The pattern and timing of the 6-bit charge can be seen in figure 2.3. An initial long pulse is sent when the peak of the detector pulse is found. This pulse edge represents the BCID

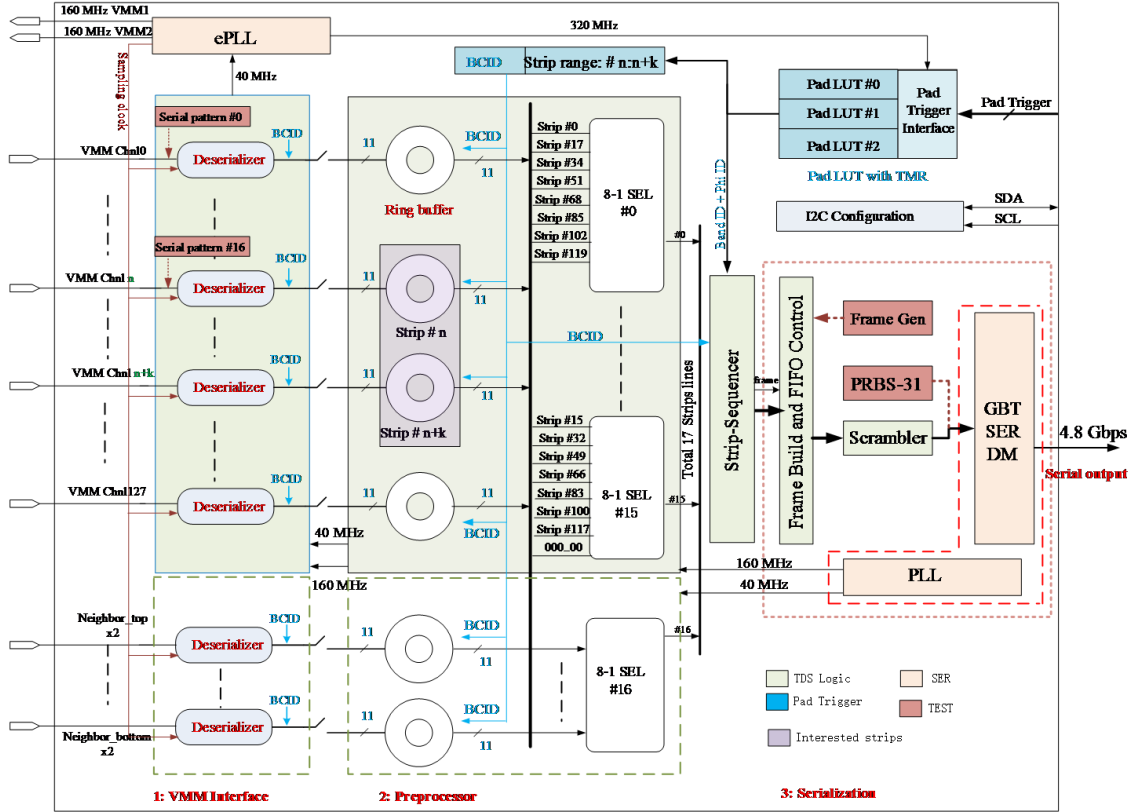


FIGURE 2.2: Block diagram of the strip-TDS.

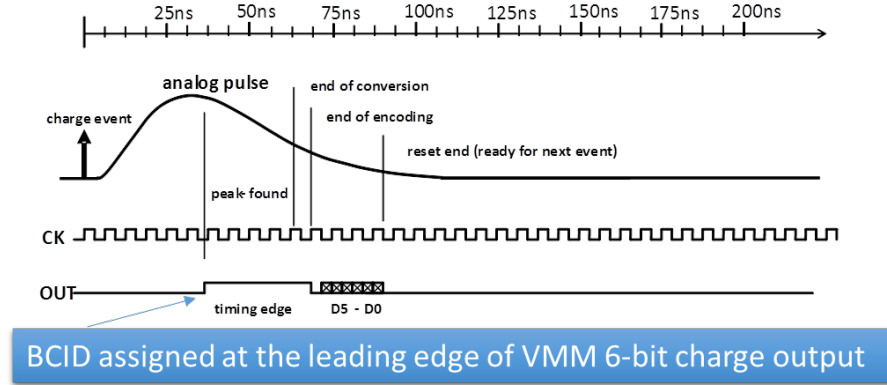


FIGURE 2.3: VMM charge output format and timing.

of the event which is assigned by the TDS. When the VMM has finished digitizing the input pulse, the initial pulse goes to zero, followed by a half clock cycle of rest, then the 6-bit charge information. The data from the VMM is sent with a 160 MHz clock at double data rate (DDR) Most-Significant bit (MSB) first. The 160 MHz clock used by the VMM is provided by the TDS and has a programmable phase.

In the preprocessor, ring buffers are used to temporarily store the 6-bit charge from each channel along with the 12-bit BCID and 1-bit BCID flag corresponding with the event. The pad-trigger board transmits the requested band/phi ID and BCID to the TDS. The TDS then uses its internal lookup table (LUT) to match the band/phi ID to

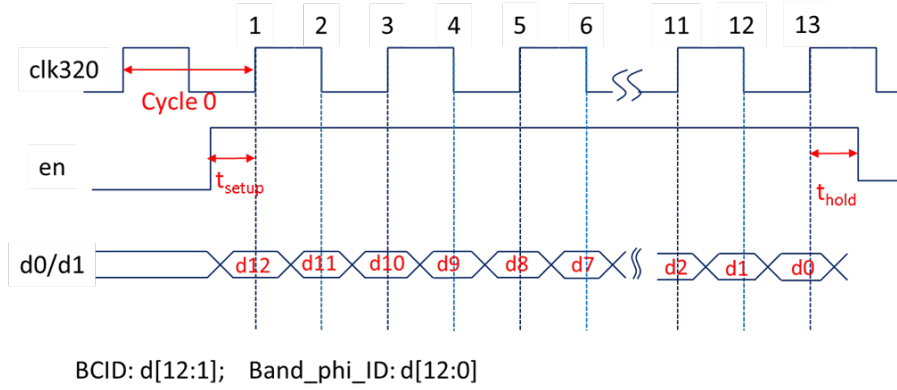


FIGURE 2.4: 640 Mbps signals from the pad-trigger board.

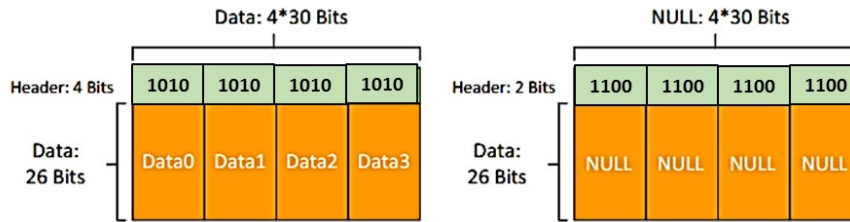


FIGURE 2.5: Data and NULL packet structures.

the corresponding range of 17 strip channels. It then searches in the ring buffers for a BCID within the matching window of the requested BCID and transmits the upper or lower 15 strips. The TDS checks for the presence of charge on the first strip to determine if the upper or lower 15 of the 17 strips should be transmitted.

The data lines coming from the pad-trigger board operate on a 320 MHz clock for a transfer rate of 640 Mbps DDR. The pad-trigger board is located on the rim and transmits data to the TDS via four differential pairs: clock, enable, and data0/data1 (d0/d1). It is very important that the data coming from the pad-trigger is aligned between the four lines so that the correct data is received by the TDS. a diagram of the pad-trigger interface signals is shown in figure 2.4.

The serialization on the TDS is done with a modified GBT core. The TDS's serializer core uses the 130 nm IBM process and accepts a 30 bit input at 160 MHz. Because this is a synchronous serial link, there must always be data transmitted. When no strip data is being transmitted, 30 bit null packets are inserted. These null packets contain a 40bit identifying header of 1100. Data packets are 120 bits long separated into four 30 bit sub-packets. Each sub-packet contains a 4-bit header of 1010 followed by 26 bits of data. A summary of these packets is shown in figure 2.5.

In order to balance the communication, a scrambling algorithm is used. This is accomplished with a standard polynomial scrambler using  $1 + x^{39} + x^{58}$ , which is the same used for 10 G Ethernet. See the IEEE standard for more information[18].

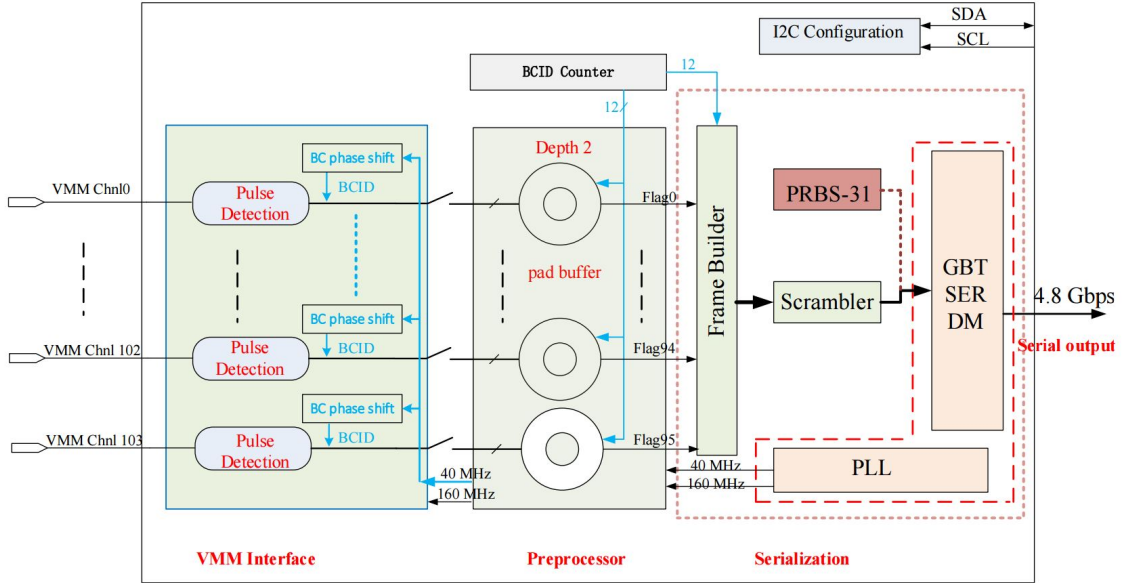


FIGURE 2.6: Pad-TDS block diagram.

## 2.4 PAD Mode

In pad mode, the TDS accepts 104 inputs from multiple VMMs. For each bunch crossing (BC), the VMM will send the pad-TDS a one bit indication of whether the corresponding pad fired. In order to match the BCID for each TDS chip with each other, a programmable 12-bit BCID offset is used. The clock used to sample the VMM inputs can also be shifted to one of eight phases in 3.125 ns steps to ensure the accepted VMM input is paired with a consistent BCID. The paired BCID and hit value is saved in the channel ring-buffer. At the end of the BC, the TDS will tally up the hits for transmission. On each BCID, the TDS outputs one 120 bit packet to the pad-trigger board. This packet contains a 4-bit 1010 data header, followed by the 104 indicator bits, and the associated 12 bit BCID. The serializer used in pad mode is the same one used in strip mode. A block diagram of the pad-tds is shown in figure 2.6.

## 2.5 Built In Test Functionality

Built into the TDS are some testing modes. These features allow specific portions of the chip to be isolated for testing. The first option is a 31 bit Psudo-Random Binary Sequence (PRBS) generator[19] in the serializer. When enabled, the PRBS signal is directly fed into the 4.8 Gbps serializer. This mode is used to check the functionality and stability of the serializer.

The next testing option is to enable or disable the scrambler. With this option, the data coming out of the TDS can be easily recognized. In this mode, the data from the pad/strip logic is passed directly to the serializer to be transmitted.

In strip mode, there is a built in router-protocol test. When enabled, a frame generator injects complete router frames to the scrambler. This includes the 4-bit data headers and 26 bit data fields. The contents of the fields include a counter which increments on each packet. The purpose of this test mode is to ensure the TDS can generate correct frames, and the router can correctly identify them and descramble the contents.

When no pad-trigger is available, the trigger may be bypassed. In this mode, the first group of 15 channels which contain charge is transmitted on each BC. This mode uses a selection rule which will ignore charge inputs on channels outside of the initial charge range.

The final testing mode is the internal VMM generator. In this mode, VMM emulators are used to output a 6-bit charge to the first 16 channels with a known pattern. In this method, the full circuitry path for the first 16 channels may be tested without the need for external input from a VMM.

## Chapter 3

# Test Setup

The creation of an automatic test fixture for the TDS chips is motivated by the initial testing of the TDS chip by hand. Upon the arrival of the TDSV2 chips, their functionality was thoroughly checked by hand. Each operational mode and option of the TDS chip must initially be checked to verify the design. The initial effort to check the first chip’s functionality took roughly one month. A large portion of this time was spent engineering the firmware to administer the tests and debugging the testing apparatus. Once the first chip had been tested, the time it took to test each additional chip was about 8 hours. This testing still stressed each feature of the TDS chip, but could be completed much quicker since the firmware and tests had already been created.

Moving forward to check the functionality of the first batch of chips, it is clear that spending one day’s work on each of the 200 chips was not practical. The two options were to decrease the complexity of the administered tests, or to automate them. The first option is not desirable because it does not guarantee a ‘passed’ chip is completely functional. This leaves the option of automating the chip testing to dramatically increase the number of chips which can be tested per day, and to maintain the same depth of tests being performed.

### 3.1 Methodology

The evaluation of a single chip involves multiple tests which require configuring the TDS, simulating the inputs, and monitoring the output of the TDS. Each individual test is designed to stress some aspect of the chip, and requires accurate timing and reliable evaluation. The direct connection to the chips are handled by a high end FPGA to interface the simulated VMM inputs, pad-trigger connection, and I2C configuration. A Xilinx VC707 evaluation board[20] was chosen as the appropriate FPGA, as discussed in 3.4. In addition to the VC707, a computer is used to provide a human interface to the setup.

Where to place the automation was a big question when designing the system. It is easiest to place the entire checking logic and testing methods in software, however it is

not practical to transfer the TDS output data to the computer in real time. Running three test setups in parallel, the computer would need to process almost 2 Gigabytes per second of data, while controlling the inputs at a similar data rate. The impracticality of this necessitates some level of checking logic to be handled by the FPGA.

In the final setup, the computer and software were responsible for preparing the TDS for the tests, starting and stopping them, and deciding what course of action to take based on the test feedback. The software behavior is described in more detail in 3.3. The firmware running on the FPGA was responsible for configuring the TDS chip, administering the tests, and checking the serial output from the TDS. This methodology put a majority of the workload on the firmware which allowed the software to be simpler and focus on higher level aspects of testing the chip.

## 3.2 Firmware Development

The firmware is split into three major components: the TDS interface and data checker, the I2C configuration module, and the Ethernet interface to the computer. The firmware was separated in this way to minimize the clock domain crossings which need to occur internally. All of the TDS interface logic runs on a fast clock domain synchronized to with the TDS chip output. The I2C interface runs at a very slow speed compared to the other inputs and outputs of the TDS, and was thus separated. Finally, all communication over Ethernet is synchronized to its dedicated onboard oscillator. A block diagram of the firmware is shown in 3.1.

At the highest level, there is no direct communication between the I2C module and the TDS data checking module. The only communication with the data checking module consists of predominately static signals. All communication coming out from the data checking module is done through a FIFO (First In First Out) buffer. This minimal communication scheme makes it easier to consolidate and refine the critical checking logic and minimize the interference with other portions of the firmware.

The firmware steps to setup and run a particular TDS test are as follows. First, the computer sends a configuration command to the FPGA with a TDS configuration. The firmware then programs the TDS via the I2C link and passes the relevant information to the checking logic. The checking logic then prepares and begins checking the streamed data from the TDS. The software then initiates the test monitoring by sending a start command which enables the FIFO output from the checking logic. This data is then streamed to the Ethernet interface where it is sent back to the computer in bursts. When the test has completed, the computer sends a stop command which disables writing to the FIFO.

### 3.2.1 TDS Interface and Data Checker

The TDS interface and checking logic is the most critical portion of the firmware. This part of the firmware handles all of the high speed digital communication with the TDS

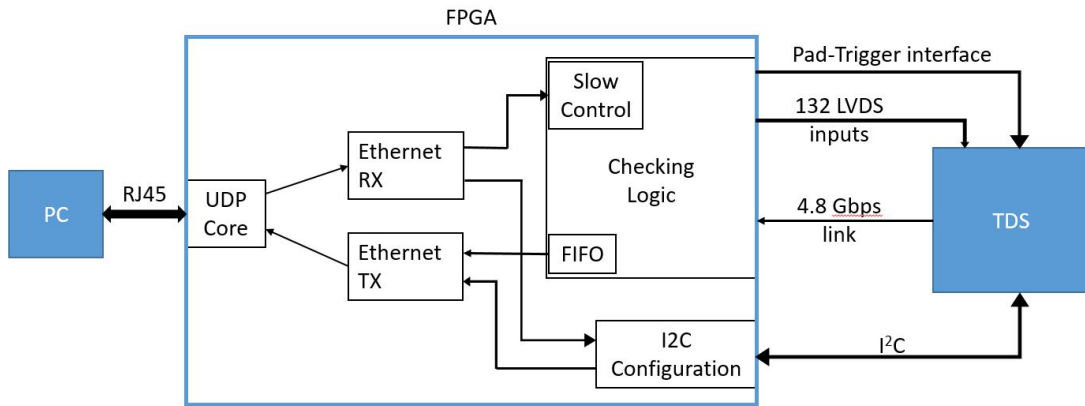


FIGURE 3.1: Block diagram of the firmware.

chip. The high speed transceiver provides the clock which is used in this portion of the logic. The clock used here comes from the same source as the TDS and is 160 MHz. Inside the firmware, the TDS serial data is split into 20 bit chunks by the high speed transceiver.

Both the VMM inputs and pad-trigger interface are simulated in this portion. In strip mode, the VMM outputs produce a simulated, fixed-length charge pulse followed by a 6-bit output. These outputs are synchronized between all 132 channels such that for a given BCID, all outputs will arrive to the TDS at the same time. In pad-mode, the TDS produces a hit on each channel in intervals according to a 104 bit channel mask. This mask is set over Ethernet, and is thus not quickly modifiable.

There are a total of six test types which require checking the high speed TDS output. Each of these tests has its own logic written in this portion of the firmware, as well as a different result format. The six tests are: PRBS, router-protocol, global test, triggerless strips, LUT for strips, and pad mode. Each test is described in greater detail in the following paragraphs. When the firmware is running, each of these test modules is always receiving the TDS output data and performing the checks. When the firmware ‘switches’ between tests, it is only switching the output multiplexer to reflect the correct test.

### *PRBS*

The PRBS test is the most straightforward of these tests. In this mode, the firmware checks the 20 bit chunks using the inverse polynomial of the transmitter. The number of errors are counted in each 20-bit packet and added to a counter. This counter size is large enough such that if all bits are incorrect, it will not overflow within the period of saving the results. When a reset is recieved, the error counter is set to zero. Output from this module is only the 16 bit error counter at a rate of roughly 160 kHz.

### *Unpacking the router-protocol*

In the Strip mode, all of the relevant data is encapsulated within the TDS-Router protocol. First, the 20-bit output from the GTX transceiver is buffered to create 30-bit outputs. The 4-bit identifying headers for the router protocol are not scrambled, and are thus used to align the packet’s position within the 20-bit frames. Initially, the location of the header within the packet is random. A trial position is chosen and the data stream is checked

for the header. If the headers are not found for many consecutive cycles, the position is incremented. This is repeated until the headers are locked to the 4 MSB of the 30-bit packets.

Once the location of the header is locked, the contents is passed through a descrambler to produce unscrambled 26-bit packets which are fed to the various checking modules. The headers continue to be monitored for misalignments. If there are more than 8 packets with incorrect headers, the alignment process is repeated.

#### *Router Protocol Test*

The Router Protocol test utilizes the built in frame generator in the TDS. In this mode, the TDS sends correctly formed data packets with the data contents as a counter. For each 104 bits of data (four 26-bit packets), the fifteen 8-bit counters are checked and a one bit result is produced for each. The final result output is 13-bits for every 120-bit packet. In this mode, null and data frames alternate. Thus, 13-bits of output are produced at a rate of 20 MHz, or 32.5 MB/s.

#### *Global Test*

In this mode, the TDS gives a fixed charge input of 6'b011110 to the first 15 channels. These outputs are then paired with a fixed band-phi ID of 13'b10101010101 and a consistent 6-bit BCID. Comparing the 14 charges, the band-phi ID, and the BCID produce a 16 bit result at a rate of 625 kHz, or 1.25 MB/s.

#### *triggerless strips*

This test requires the fake VMM input to the TDS chip. In this mode, a single channel is enabled in the TDS on which the simulated VMM charge is sent. The charge sent to each channel corresponds to its channel number. For example, channel 0 receives 6'b000000, channel 1 receives 6'b000001, etc. At channel 64, the process restarts such that the 6 LSB of the channel number are used.

The data checker monitors the data packet's band-phi ID to be a constant 13'h555 and the correct charge to be transmitted on the channel. These two binary results along with the 7 bit channel number for a total of 9 bits are sent as a result for each data packet.

#### *Strips LUT*

In the final strip testing mode, the full data path through the TDS is checked. In this mode, a LUT is used to match the strips to a band/phi ID and send triggers to the TDS. To start, the LUT must be programmed into both the TDS and checking logic. The data checker then monitors the returning packets to ensure the band/phi IDs match those in the LUT and all of the charge contents is correct. These three binary results along with the 13 bit result of matching the charge is saved as a result for each data packet. This mode represents the final functionality of the TDS chip on the front-end boards.

In this final mode, the TDS accepts triggers to indicate what data needs to be sent out. These triggers will eventually come from the pad-trigger board. The input data rate to the TDS is 640 Mbps in DDR. These lines must be simulated by the FPGA. There is a very tight window on which the DDR signals must be aligned. For 640 Mbps, the time for each bit is 1.56 ns. Taking rising and falling times into account, the data coming

out of the FPGA must be aligned within 1ns of each other. This is very difficult for the FPGA to accomplish using only constraints. It was found that even small changes in the firmware would change the output timing by more than the 1 ns allowed. To overcome this, dynamic phase shifting blocks were used on the output lines. This feature is only available on some channels and is an advantage of the Virtex-7 FPGA which was used.

### *PAD Mode*

This test allows for the testing of each pad channel individually. In Pad mode, the TDS takes binary input from the VMM to indicate whether a pad channel was hit in each BC. These signals need to be tightly constrained on the final Front End Boards (FEB), however to allow for small variations the TDS can select one of eight clock phases to sample the channel with. This acts like a delay compensator. This is the main functionality which must be tested in the pad mode. Because each pad channel has dedicated processing logic within the TDS, it was chosen for the automated test to only check each individual channel. This also eases the timing requirements of the testing firmware.

The method used to check this interface uses a binning method. For a particular channel and phase, the TDS is first programmed to only enable that channel and set it for the phase offset.

To start the test, a synchronization procedure must be completed between the TDS and FPGA which synchronizes the BCID counters in the two devices. During this procedure, the FPGA adjusts it's local counter iteratively until it can correctly identify the BCID of hits coming back from the TDS. Once this is completed, the test is carried out.

The inputs to the TDS are given according to the unshifted FPGA clock, while the TDS samples them with its phase shifted clock. In this manner, a proportion of the inputs will be identified as the intended BCID, while others will be offset by one. This forms the two bins which are used.

The number of hits falling in each bin is proportional to the fractional shift. Using eight steps, this gives eight possible proportions. For example, at the third phase offset,  $3/8$  of the hits should fall in the shifted bin, while  $5/8$  are expected to fall in the original. Taking the ratio, at any given time the shifted bin should have  $3/5$  the counts as the original bin.

In the FPGA, the two bins are 12 bit registers. When a test is run, the FPGA knows which channel to expect hits from, and begins sending and counting hits at a regular interval. Upon each received hit, one of the bins is incremented. When one of the bins reaches a maximum value of `12'hFFF`, the value of the other register is sent as the result. In this way, the ratio can be calculated by assuming the other value.

### **3.2.2 I<sup>2</sup>C Interface**

For the TDSv2, all configuration must be done over I<sup>2</sup>C. The TDS has a total of 1296 bits for configuration which need to be initialized. These are divided between 12 individually addressed registers. In addition, there are two registers for monitoring the status of the

TDS. Because the I<sup>2</sup>C interface runs at a comparatively slow frequency, it was separated from the rest of the logic and placed in its own clock domain.

A dedicated state machine was written to complete the I<sup>2</sup>C reads and writes. The state machine runs at a frequency four times that of the bus. It accepts an input of 16 bytes for writing, a read/write indicator, the number of bytes to read/write, and a start signal. If a read is selected, a 127 bit output will be updated with the read values. This state machine only interacts with the rest of the logic through a small control state machine running at the same frequency.

There are two methods which can be used to perform reads and writes. The primary method is over Ethernet where there are various commands to write and read all or parts of the TDS configuration. The secondary method is over a Xilinx-VIO (Virtual Input/Output) core which allows individual registers to be controlled through the Vivado software. This interface is mostly used for debugging.

The Ethernet method allows for two types of I<sup>2</sup>C writes to the TDS. The first is a complete configuration. In this case, the entire configuration is sent over Ethernet to the FPGA, then the I<sup>2</sup>C interface handles the transactions needed to write the configuration. The second method is used to configure the TDS for an individual test where only some of the registers need to be configured. To read from the bus over Ethernet, a command is sent to the I<sup>2</sup>C module to initiate the reads. Once the entire configuration is read, it is sent back over Ethernet.

If at any point there were errors in the I<sup>2</sup>C transaction, a flag is raised which activates a warning packet to be sent out over Ethernet. In this way, if there are problems with the configuration the software will be informed and may choose the course of action.

### 3.2.3 Ethernet Interface

The FPGA evaluation board used has a built in Ethernet connection which is used to communicate with the control software. The communication protocol used is UDP (User Datagram Protocol) which uses ports and IP addresses to direct the packets. This protocol is very easy to use through standard software libraries, however does not provide any assurance a packet reaches the destination. This, however, is easy to add in a point to point setup. In the firmware, UDP is much easier to implement than a more common protocol such as TCP/IP.

An overview of the internal structure and data flow of the Ethernet core is shown in figure 3.2. The physical layer interface is SGMII (Serial Gigabit Media Independent Interface). As the name suggests, the Ethernet data is streamed to and from the FPGA over a differential 1.25 Gbps DDR serial line. The FPGA decodes the data stream using a high speed transceiver which results in 8-bit transactions with a frequency of 125 MHz. The signals at this step are raw physical-layer signals which need to be decoded.

To translate between physical layer signals and more user-oriented Media Access Control (MAC) signals. This is done with the Xilinx Tri-Mode Ethernet MAC (TEMAC) [21]. The core communicates using the AXI-Stream protocol to the UDP core.

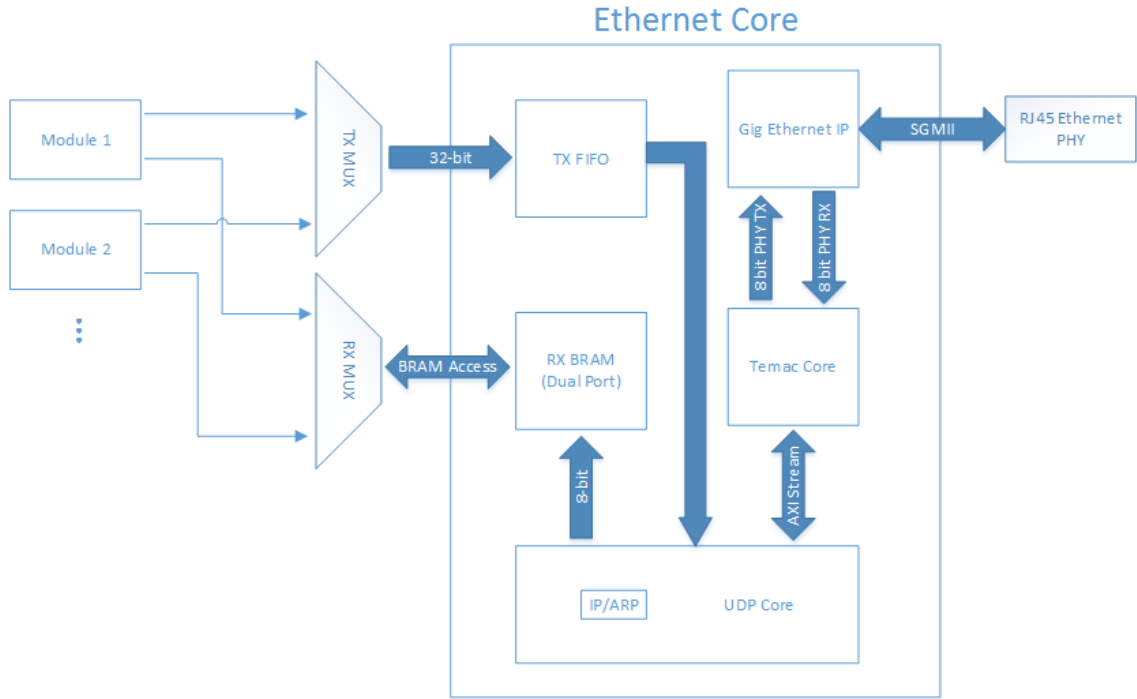


FIGURE 3.2: Internal structure of the Ethernet core.

The UDP core chosen is from open-cores[22] and implements the UDP protocol, IPv4 addressing, and handles ARP (Address Resolution Protocol) requests. ARP is used in networking to map the IP address of a device with its MAC address. This way all transactions which take place are based on the IP address of a device which is easily changeable. Before a transaction can take place on the network, an ARP request is broadcasted asking who owns a particular IP address. The device with that address must reply with it's MAC address. This information is stored by the devices in the path between the sender and receiver. After this has taken place, the UDP packets can be sent between the two devices.

The user interface to this UDP core is based on custom data types to transfer information related to the packets. For a received packet, a flag is raised to indicate the MAC is receiving data. Following this flag, the MAC and IP address of the sender, UDP send/receive ports are updated, and 8-bit parallel data begins to stream from the core. This data has zero latency from the data coming out of the TEMAC core, and thus must be saved or processed in real time. At the end of a received packet, an end of packet flag is raised and the core waits for the user to release the data. The core waits to accept a new transmission until this flag is cleared. At any time during the receive, the user can reject the packet to end the transmission and allow the core to accept another packet.

To send a packet with the core, a similar procedure is followed. First, the destination IP address and the send/receive ports are updated. After this, the core will indicate when it is ready to send a packet. At this point, the content of the Ethernet packet is streamed to the core in 8-bit parallel chunks. A end-of-packet flag is raised upon the last 8-bit value written. After this, the core calculates the CRC for the UDP and sends the entire packet.

TABLE 3.1: General Ethernet Packet Structure

|                     |   |
|---------------------|---|
| 31                  | 0 |
| Header / Identifier |   |
| Command / Unique ID |   |
| Packet Defined Data |   |
| ...                 |   |

TABLE 3.2: Commands supported for PC to FPGA communication

| Command | Bits     | Name              | Description                                                   |
|---------|----------|-------------------|---------------------------------------------------------------|
| 0x1     | [1295:0] | Configure         | Command to configure the TDS                                  |
| 0x3     | -        | TDS Config Status | Triggers a read of the TDS configuration and status registers |
| 0x5     | -        | Bind              | Binds the IP address of this packet as the host PC            |
| 0xF     | -        | Ping              | No function, only send the reply packet back                  |
|         |          |                   |                                                               |
| 0x10    | [543:0]  | Test Config       | Send test configuration to FPGA and TDS                       |
| 0x11    | -        | Test Start        | Start configured test                                         |
| 0x12    | -        | Test Stop         | Stop configured test (if needed)                              |
| 0x13    | -        | Reset PRBS        | Resets the PRBS error counter                                 |
| 0x14    | -        | Reset CNT         | Resets the Pad count                                          |
| 0x15    | -        | Reset phase       | Resets the Pad phase                                          |
| 0x16    | -        | Phase Shift       | Shifts phase of clock used to drive TDS inputs                |
| 0x17    | [31:0]   | Output Delay      | Sets the output delay of d0, d1, and enable                   |

For our purpose, only basic read and write functionality is needed since the only communication will be with the control computer. A simple protocol was adopted to filter the packets based on their command which is used for both TX and RX Ethernet communication. The complete specification is detailed in [23].

The packet structure discussed here is placed in the user-data field of the UDP packets. The general packet structure for RX and TX is shown in table 3.1. In this implementation, the Header / Identifier field is always a constant `32h'decafbad` which is used for easily identifying the test fixture's packets among other network traffic.

The command / unique ID field of the packet dictates its contents. For this field, different commands are used for TX and RX communication. When a packet is received, this command is used to direct it to the appropriate portion of the firmware. For TX packets, this field tells the software what to expect. A summary of the commands for RX and TX is given in tables 3.2 and 3.3 respectively.

Upon the receipt of an Ethernet packet, the firmware begins filtering it as the data is streamed in. First, the IP address is checked to be from the paired PC's address<sup>1</sup>. Then, the header is checked to match `32h'decafbad`. The final filter is on the command. If the command field is recognized, the packet is entire packet is saved into local dual-port BRAM (Block Random-Access-Memory). The access to this BRAM is then turned over to the destination module which has been identified based on the command. This is done

<sup>1</sup>This address can be changed by sending a bind command from any IP

TABLE 3.3: Commands supported for FPGA to PC communication

| Command | Bits     | Name              | Description                                 |
|---------|----------|-------------------|---------------------------------------------|
| 0x0     | [31:0]   | Reply             | Packet echoed for each receive              |
| 0x1     | -        | Ack Error         | Sent if error occurred in I2C communication |
| 0x2     | -        | Config Done       | Sent when I2C configuration completes       |
| 0x3     | [1503:0] | TDS Configuration | Contains the 1504 bits in the TDS registers |
| 0X10    | Variable | Test Data         | Variable length packet of test results      |

with a multiplexer. Once the module relinquishes access to the BRAM, the controller sends a reply packet back to the PC to indicate the packet was received and processed. This reply is used by the software to ensure anything it send makes it to the FPGA.

All interface modules which read from the local BRAM use a common 125 MHz clock provided by the Ethernet core. The signals to interface to the BRAM and dictate access are encapsulated in their own data type to ease the transfer of the signals through the firmware.

When a module wants to send data to the PC, it first requests access to the sending FIFO. The highest priority is given to the Ethernet RX reply packets, and further priority is given on a first-come first-serve basis. When a module is granted access to the sending FIFO, it can write a single packet to be sent. When the end of packet signal sent to the FIFO, the module loses control of the signals and must request access again.

The sending FIFO was created with the standard Xilinx FIFO generator[24]. It has a write width of 32 bits and a read width of 8 bits. The write depth is large enough to hold multiple full-length Ethernet packets. Paired with the sending FIFO is a length FIFO. The TX firmware keeps track of the packet size as it is written into the sending FIFO. When the end of packet signal is asserted, the length is saved into the length FIFO. Packets are sent from this module whenever the length FIFO is not empty.

### 3.3 Software Development

The accompanying software GUI (Graphical User Interface) was written in Python. This software automates the testing of chips and allows for multiple chips to be tested at once. The software is comprised of three major portions, the user interface, the camera input processing, and the test control. The internal working of the software is only briefly covered here.

The user interface allows the user to scan the TDS chips to initialize them for testing, and carry out the tests on multiple boards. A screenshot of the GUI during a test is shown in figure 3.3. The interface is minimal and will be updated for future tests as described in section 5.2. When the tester wants to begin a chip test, they first type their name into the “Testers Name” field and press “Scan Board”. This brings up a live view of the connected camera. The tester must scan the QR code attached to both the TDS chip and

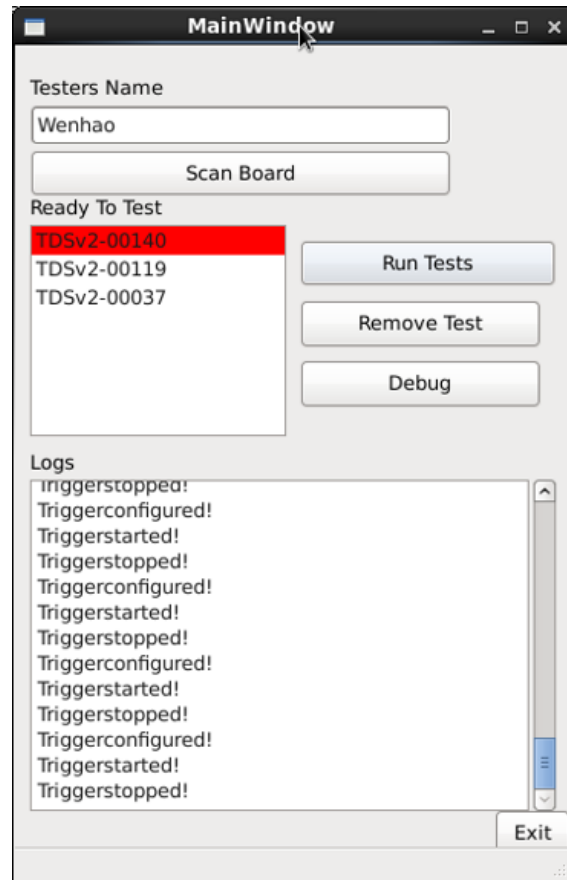


FIGURE 3.3: Screenshot of the GUI during a test.

FPGA board. This lets the software know which board has the TDS chip in it. After the chip has been scanned, to begin a test the “Run Tests” button is pressed. In the case of a test failure where the chip will not be retested, the test is selected and removed with the “Remove Test” button. While tests are running, a portion of the results is shown in the “Logs” area.

The camera portion of the GUI is used to scan the QR codes which identify the individual TDS chip and FPGA. Each FPGA is programmed with a unique IP address which is associated with the QR code. This way, after the codes have been scanned, the software knows where to direct the commands for that TDS chip. Using the camera also eliminates any errors which could occur if the values were entered by hand.

The final portion of the software is the data processing and test control. This is the part of the software which administers the tests to the chip. The software must process the incoming data in real time and decide how to proceed. This portion of the software also creates the log files which can be used to look back on the specifics for an individual chip test. Because of the high rate of data which is fed back during some tests, the software must efficiently process the data. To do this, the use of hash tables were selectively used to improve the performance.

In order for the various tests and parts of the program to run simultaneously each portion was given its own thread. The GUI, camera portion, and each running test operates

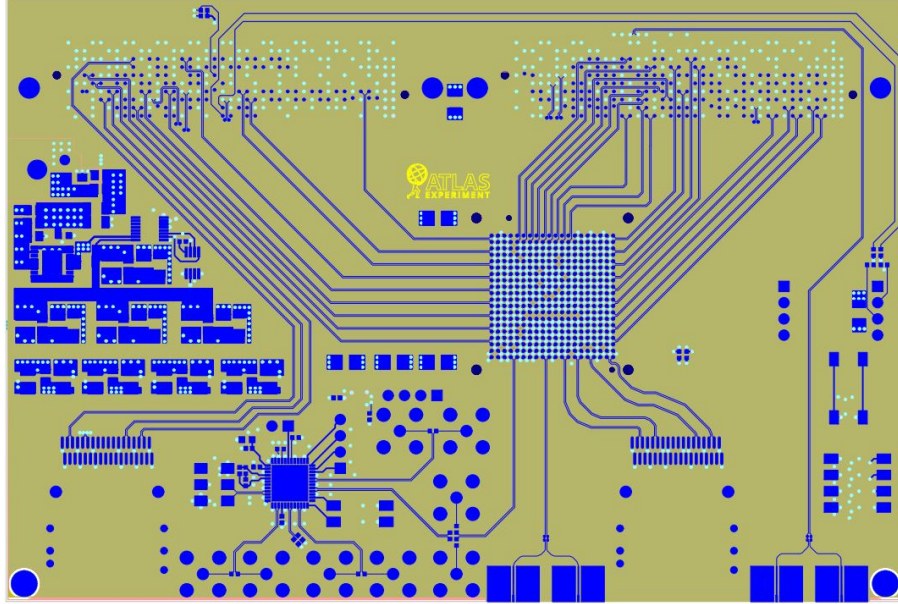


FIGURE 3.4: Top layout of the TDS mezzanine card.

within its own thread. The threading tools built into PyQT [25] were used to accomplish this.

### 3.4 Hardware

The TDS chip has a 400 pin BGA package which must be interfaced to the FPGA evaluation board. A custom designed Printed Circuit Board (PCB) was made for this purpose. The board attaches to the FPGA evaluation board with two high density FMC (FPGA Mezzanine Card) connectors. These connectors provide the interface for the simulated VMM signals and clocks. There are paths in the design for the high speed data and I<sup>2</sup>C configuration, however these were not fully tested. Therefore, external connections were used. In early tests, a wire-bonded mezzanine board was used, followed by a directly bonded chip. Final versions of the mezzanine board feature a BGA socket which allows for chips to be inserted and removed without bonding.

The BGA socket used is from Ironwood Electronics [26]. The socket is design to handle 6.5+ GHz signals and can be mounted to the PCB without solder and the same pinout as a regular 400 pin BGA chip. This allows the same design to be used for the BGA socket and soldered package mezzanine boards.

The mezzanine board has an onboard clocking chip which serves as a clock source for the system. The chip is a Texas Instruments CDCE62005 [27]. The clock chip is programmable with an external programmer to synthesize the 40 MHz TDS clock, and 160 MHz reference clock for the FPGA.

In order for the TDS chip to properly decode the charge coming from the VMM, the signal paths must be very close. On the mezzanine board, the differential lines are constrained

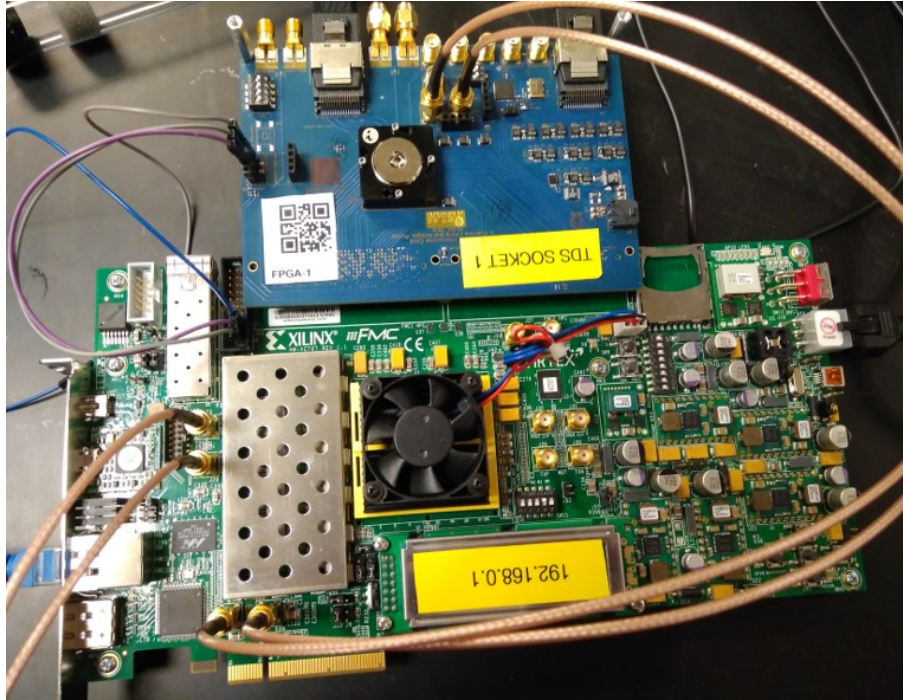


FIGURE 3.5: FPGA and mezzanine board with BGA socket.

such that the maximum variation due to signal propagation is minimized. The paths and pin mapping were chosen to make the routing easier, as seen in 3.4.

As discussed in 3.3, each the TDS and FPGA are identified with a unique QR code. The FPGA QR code can be seen in figure 3.5 and an example of a TDS QR code in figure 3.6. The QR code placed on the TDS chip serves as a unique serial number which can be used in the future to reference the test results. The QR codes were automatically generated, and printed on a special label. These labels designed to withstand the heat of a reflow soldering assembly method and can be printed with a regular laser printer.



FIGURE 3.6: TDS placed in BGA socket with QR code.



## Chapter 4

# Chip Testing Procedure

A standard testing procedure was developed to provide consistent results between setups and TDS chips. This procedure includes the actual tests which were chosen to exercise and test the TDS' functionality.

### 4.1 List of Tests

A total of eight tests were chosen to evaluate the TDS chip's functionality. These tests begin with very straightforward functionality checks and move to global chip tests which emulate the actual use of the TDS. Specific details and flow-charts of each test procedure can be found in [28]. The tests are as follows, in order.

- I<sup>2</sup>C
- PRBS
- Router - Protocol
- Strip - Global
- Strip - Individual channel
- Strip - LUT - Individual
- Strip - LUT - Complete
- Pad Mode - Individual

#### 4.1.1 I<sup>2</sup>C

For this test, the I<sup>2</sup>C configuration of the TDS is tested. Because of the simplicity of the I<sup>2</sup>C logic inside the TDS, this can serve as an indicator if the chip is properly powered and alive. During the test, three arbitrary configurations are written to the configuration and read back. The first two are inversions of each other so that each bit is checked. The final configuration sets the TDS up for future tests. In a successful test, the configuration read back identically matches the one written.

In the case that this test fails, it is normally an indication the setup is not configured properly. If after several attempts the chip still fails the I<sup>2</sup>C test, it is considered a broken chip.

#### **4.1.2 PRBS**

For this test, the built-in PRBS generator is enabled and the output is checked for five minutes. Each TDS must have a very low rate of errors in its data stream, however to reach the required bit error rate, the TDS must be tested for over one day. This is not practical for every chip, so a shorter time was chosen. After five minutes, there is a high likelihood the serializer will meet the required bit error rate. From previous experience testing the serializer, the most common mode of failure is complete and thus would be caught after a short amount of time testing. In the case that the chip produces errors within the first five minutes, the chip is retested. If after retesting twice the chip still cannot sustain the PRBS pattern for five minutes, the chip is marked as broken.

#### **4.1.3 Router - Protocol**

During the router - protocol test, the TDS is configured to output the router frames containing only counters. This test is run for a short amount of time over which all values returned must be correct. This test produces a large amount of data and care must be taken to ensure not more than one board reaches this test at any given point. In the case that this test fails three times, further strip-mode tests are skipped and the chip is marked as broken.

#### **4.1.4 Global Test**

Similar to the Router - Protocol test, this test uses the internal word generation in the TDS to output the result of the internal data generator. The data rate for this test is comparatively low. If all of the returned data matches what is expected for the fake VMM hits, the chip may move to the next strip-mode test. Otherwise, further strip-mode tests are skipped and the chip is considered broken.

#### **4.1.5 Strip - Individual Channels**

Before testing the lookup tables and trigger functionality, each individual channel is tested in bypass trigger mode. In this mode, the first channel with charge present and the following 16 channels are processed and sent. To make data checking easier, all channels were always sent a fixed charge corresponding to their channel number. To test an individual channel, all other channels were masked using the TDS' channel disable functionality. For this test, each channel was tested in this way and the corresponding charge was checked to be correct and stable.

Due to the design of the TDS, the highest channels in each group of 17 cannot be tested in this mode. These channels are consistent between TDS chips and were skipped. Their functionality is checked on the LUT tests.

#### 4.1.6 Strip - LUT - Individual

In order to use the LUT functionality, the table must be programmed with corresponding band/phi ID to TDS channel correspondence. In the final setup on the NSW, each chip will be uniquely programmed with a mapping, however these mappings can have any range of values. For this reason, the LUT performance is designed to be independent to the configuration stored within it. Before testing the complete LUT, the 16 individual locations within the LUT were tested. This was done by storing zeros in all other locations, checking that the TDS correctly matched the band/phi ID from the simulated pad trigger to the corresponding channels, and processed the charge correctly. In total there are 16 locations in the table, and thus this test was repeated 16 times.

#### 4.1.7 Strip - LUT - Complete

Once all LUT positions are confirmed to be working, the entire strip-mode data path may be tested. To do this, the LUT is programmed with a complete mapping which covers all TDS input channels and a range of band/phi IDs. The FPGA loops through each band/phi ID pairing and the stream of data is matched with the expected values. Two such LUT configurations were used, and the tests were repeated for a longer amount of time than previous strip-mode tests.

#### 4.1.8 Pad Mode - Individual

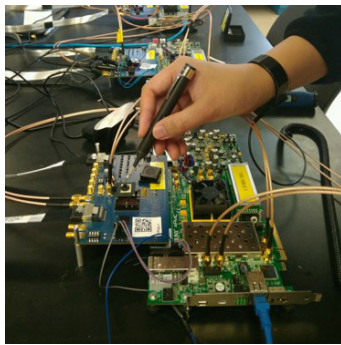
The test method for pad mode is to check each of the 104 channels individually using the scheme presented in section 3.2.1. The feedback progression expected for the 12 bit counter is  $0 \rightarrow 585 \rightarrow 1365 \rightarrow 2457 \rightarrow 4095 \rightarrow 2457 \rightarrow 1365 \rightarrow 585 \rightarrow 0$ . Each channel was individually enabled by masking the other channels with the TDS' channel disable register. The phase was incremented and if the reply count was close to the amount expected, the next phase was tested. If a bad count was found, the channel test was restarted because this usually corresponds with an error in synchronization. Only if a channel repeatedly gave incorrect phase counts was the chip marked as broken.

## 4.2 Test Preparation

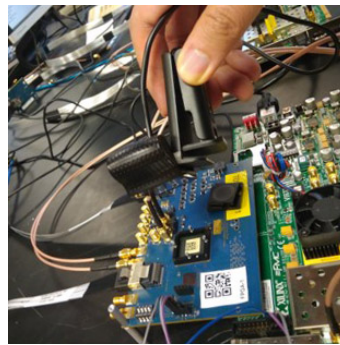
Before running the tests, the chips need to be prepared in their sockets, the setup's QR codes scanned, the sockets secured, and the boards programmed. From the packaging company, the TDS chips come in large box arrays, as seen in figure 4.1. From here, the chips labels are affixed and the chips are placed in the socket using a vacuum pen. When



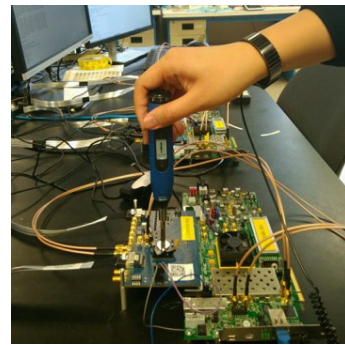
FIGURE 4.1: A sample of TDS chips as packaged from the factory.



(a) TDS being placed into socket with vacuum pen



(b) TDS and FPGA QR code being scanned



(c) Socket being tightened with torque wrench

FIGURE 4.2: Illustration of preparing setup for testing

handling the chips, a static discharge wristband is worn to avoid damaging the chips. The gold pin-0 indicator on the TDS chips is aligned with the pin-zero indicator for the socket and the chip and board are scanned.

After a successful scan, the socket is closed by tightening the screw on its lid. The socket manufacturer recommends a torque between 5 and 7 lbs-ft of torque to be applied[26]. For consistency, a torque wrench is used to close the socket. At this point, the FPGA can be powered on and automatically configured from the onboard flash memory. Now, the test can proceed. The preparation is illustrated in figure 4.2.

### 4.3 Running and Monitoring the Tests

For each chip, a logfile is created upon the start of the testing. This logfile stores the unabridged output from the GUI, including individual test results and any errors which occur while the test is running. The decision making on which test to run and how to proceed if errors occurred is done by the software. In this way, the user only needs to check the summary at the end of the test to see the results. All logfiles are saved for reference from end-users if there are problems with TDS chips after they have been installed.

## 4.4 Post-Test Procedure

Once the chips have been tested, they must be stored in static safe boxes. Each box holds six chips and is labeled with the chips it contains. Removing the chips from the socket is the opposite procedure as putting them in. Chips which failed the automatic test and require further testing are set aside.

## 4.5 Duration of Test

A big factor in creating the automatic test fixture was to reduce the testing time from many hours to many minutes. When the final batch of a six thousand chips is delivered, we will have only a few months to test all of them before the assembly sites will need them. After testing the first batch of 180 chips, the average time spent testing each chip was 30 minutes. Running three setups in parallel, the effective time per chip reaches 10 minutes per chip.

The actual time each TDS requires is heavily variant. If no errors are encountered for the entire test, the chip completes the test in as little as 12 minutes. The most time consuming tests are the PRBS test, the individual channel test, and the pad mode test. The PRBS is a fixed length of five minutes, so this cannot be avoided. The time consuming part of the individual channel tests are the phase alignment. The software attempts to find one phase which works for all channels, however there are 118 phases available in the FPGA. Typically, the FPGA will find a correct phase setting in the first 20 phases, however a few chips required trying 70 or more phases. These chips took upwards of an hour to complete.

In the pad mode test, a similar phase problem can cause a large delay in testing time. The reset and synchronization procedure the FPGA uses to synchronize the BC counters is unstable. It correctly finds the synchronization the majority of the time, however this must be performed on each channel. As a result, it can take up to five attempts for a single channel to pass the test.



## Chapter 5

# Testing Results

The first production run of TDSv2 chips yielded 200 dies and 189 chips after packaging. The chips which were not packaged were used by the packaging company to fine-tune the production process or wire-bonded. Of the 189 chips which were successfully packaged, twenty five were tested by hand and sent to users before the automatic test fixture was complete. The remaining 161 TDS chips were tested with the automatic test fixture, of which 158 passed all tests.

The total testing time for the 161 chips was four days at more than eight hours per day. As the bugs were worked out of the system, the efficiency of testing increased. In the last two days, more than 75% of the chips were tested. This balances the extra hours spent testing the chips each day, so taking an average over the four days at eight hours per day results in an average of five chips per hour, or twelve minutes per chip with three setups running in parallel. This figure should be used as a ceiling for testing time because future improvements in efficiency will only reduce this number.

Taking eight hours per day of work for five days a week, this means 200 chips can be tested each week by one person. With an eventual number of 6000 chips to be tested, this results in a worst-case estimate of seven months. With an additional setup and efficiency improvements this time can be reduced.

### 5.1 Failed Chips

The three chips which failed the automatic test failed for different reasons. The first (chip 37) could not pass the LUT tests. The second (chip 119) failed the pad mode test. Finally, the third (chip 140) failed the PRBS test. In addition to these three chips, one hand tested chip was found to have no meaningful serializer output and would have failed the PRBS test.

The chips that failed the PRBS test had no correct data coming back from it. The problem was most likely not due to a configuration problem because the chip's configuration abilities were already checked in the I<sup>2</sup>C test. Looking at the data which came back during

the PRBS test, the error count is constantly increasing. This is an indication that there is a problem in the serializer component.

The second failed chip has a problem with channel 15 in pad mode. The chip was tested a total of six times on two different setups, but each time failed channel 15. This is interesting because the chip passed the same channel each time in strip mode. This indicates a problem with the pad mode input logic for the individual channel and not in the processing logic of the TDS which is shared between all channels.

The final chip which failed could not pass the LUT or pad mode tests. This chip was tested on all three setups a total of twelve times. When the chip failed pad mode, channel 43 always showed problems. However, in strip-mode channel 43 did not show any issues. The chip did struggle to find a correct phase for the strip mode tests, in a few tests failing all 118 phases.

With four of the 189 packaged chips showing a failure, the resulting yield is 97.9%. This rate is comparable to other ASICs manufactured for high energy physics and suggests the budgeted number of extra chips to be manufactured are sufficient to cover the proportion of bad chips.

## **5.2 Future Testing and Improvements**

Within the next year, the remaining 6000 TDS chips will be manufactured and tested. Before then, new improvements will be made to the test fixture.

The first will be the addition of a remote database. Currently, the chip's test results are hosted on the Git repository along with the testing software. The results are kept in the log files which are cumbersome to read if not familiar with the setup. The next version of the software will upload the testing results directly to the online SQL database. This database will be accessible by assembly sites to use as reference if problems are found with a TDS chip.

The next series of improvements will be to the software and user interface. The current version of the software has no debugging options which are useful when a chip fails. Experts are needed to manually run tests on a chip. The next version of the software will have a panel which can be used to create custom configurations to send to the TDS as well as manually start and stop tests. This will be very convenient moving forward and allow problems with the setup to be located and fixed easily.

Additionally, the feedback during a test will be formatted and displayed in the main window. Currently, the log files must be directly monitored via the terminal and are difficult to interpret if not familiar with the system. The next version of the GUI will feature real-time feedback for tests which are being run.

In addition to these user-oriented improvements, a series of efficiency gains will be made. At the current rate of 200 chips per week, the testing of all 6000 chips will take over six months. Finding the appropriate phase for each chip consumes most of the time for the

test. By optimizing this process, the pad mode testing time can be significantly reduced, and the variance in time of the strip-mode tests will decrease. It is reasonable to reduce the pad mode time by 40%, thus reducing the overall time by five minutes per chip. This would increase the rate to 340 chips per week, or about four months total.

Another point of optimization is the data processing in the software. For tests such as the router-protocol and PRBS, the rate of data coming back to the software can exceed the rate at which it can be processed. This leads to backups which sometimes will cause problems in other tests which are running concurrently. By optimizing the data processing by creating precompiled c-functions, this can be eliminated. Doing so would reduce the likelihood of false failure results and thus the number of chips which must be retested. This would have an overall effect on the order of tens of chips per week.

The final improvements to the system will be built in error checking for the test setup. By using image recognition techniques, the orientation of the chip can be checked when the QR code is scanned. This will ensure the chip is not powered on in the wrong orientation which would damage the chip. The framework for this feature is already written, but not implemented and trained.



## Chapter 6

# Conclusion

The TDS ASIC will be used as part of the ATLAS detector's new small wheel. It is responsible for passing trigger information from the front-end boards out to processors on the rim. The chip features 132 parallel inputs which it processes and selects for output on its 4.8 Gbps serial link.

Because of the need for reliability once the TDS is installed on the FEBs, each chip must be verified as working before it is passed to users. Testing each of the 6000 needed by hand is not practical given the time constraints. Therefore, an automatic test fixture is needed to speed up and standardize the testing.

The automatic test fixture created consists of a mazzanine card to hold the TDS chip, an FPGA to administer each test, and software running on a computer to control the system. The FPGA firmware was designed to produce reliable results and pass only the test information needed back to the computer. This approach allows the software to decide when a chip has passed a test, or when to repeat a test.

In total, 162 of the 189 packaged TDS chips were tested with the automatic test fixture. Of these chips, three failed for various reasons. Counting the chips which were tested by hand, the first batch has a yield of 97.9%. Future improvements to the test fixture will increase the rate of chips which can be tested and new features which make the system more user friendly.

The use of an automatic test fixture to check the functionality of ASICs is essential in high energy physics experiments where a single chip failure can lead to missing data and missing physics. Michigan's automatic testing of the TDS chip has set a benchmark for other groups to follow in their own testing of ASICs for the NSW upgrade.



# Bibliography

- [1] CERN. Lhc: the guide. 2008.
- [2] ATLAS collaboration et al. Observation of a new particle in the search for the standard model higgs boson with the atlas detector at the lhc. *Physics Letters B*, 716(1):1–29, 2012.
- [3] CMS collaboration et al. Observation of a new boson at a mass of 125 gev with the cms experiment at the lhc. *arXiv preprint arXiv:1207.7235*, 2012.
- [4] Georges Aad, Brad Abbott, Jalal Abdallah, AA Abdelalim, A Abdesselam, O Abidinov, B Abi, M Abolins, H Abramowicz, H Abreu, et al. Studies of the performance of the atlas detector using cosmic-ray muons. *The European Physical Journal C*, 71(3):1593, 2011.
- [5] ATLAS collaboration, G Aad, et al. The atlas experiment at the cern large hadron collider. *Journal of Instrumentation*, 3(08):S08003, 2008. URL <http://stacks.iop.org/1748-0221/3/i=08/a=S08003>.
- [6] T Kawamoto, S Vlachos, L Levinson, C Amelung, G Mikenberg, L Pontecorvo, D Lelouch, J Dubbert, C Dallapiccola, R Richter, et al. New small wheel technical design report. Technical report, 2013.
- [7] Georgios Iakovidis, Gianluigi De Geronimo, and Venetios Polychronakos. Vmm-anasic for micropattern detectors. Technical report, ATL-COM-MUON-2015-055, 2015.
- [8] A Caratelli, S Bonacini, K Kloukinas, A Marchioro, P Moreira, R De Oliveira, and C Paillard. The gbt-sca, a radiation tolerant asic for detector control and monitoring applications in hep experiments. *Journal of Instrumentation*, 10(03):C03034, 2015.
- [9] R-M Coliban, S Popa, T Tulbure, D Nicula, Mihail Ivanovici, S Martoiu, L Levinson, and J Vermeulen. The read out controller for the atlas new small wheel. *Journal of Instrumentation*, 11(02):C02069, 2016.
- [10] L Yao, V Polychronakos, H Chen, K Chen, H Xu, S Martoiu, N Felt, and T Lazovich. The address in real time data driver card for the micromegas detector of the atlas muon upgrade. *Journal of Instrumentation*, 12(01):C01047, 2017.
- [11] V Izzo, S Perrella, and R Vari. Pad trigger logic board, 2015. URL <http://indico.cern.ch/event/354058/contributions/832505/attachments/701568/963203/PadTriggerLogicBoard.pdf>.

- [12] Jinhong Wang, Xueye Hu, Thomas Schwarz, Junjie Zhu, JW Chapman, Tiesheng Dai, and Bing Zhou. Fpga implementation of a fixed latency scheme in a signal packet router for the upgrade of atlas forward muon trigger electronics. *IEEE Transactions on Nuclear Science*, 62(5):2194–2201, 2015.
- [13] P Leitao, S Feger, D Porret, S Baron, K Wyllie, M Barros Marin, D Figueiredo, R Francisco, JC Da Silva, T Grassi, et al. Test bench development for the radiation hard gbtx asic. *Journal of Instrumentation*, 10(01):C01038, 2015.
- [14] J Anderson, A Borga, H Boterenbrood, H Chen, K Chen, G Drake, D Francis, B Gorini, F Lanni, G Lehmann Miotto, et al. Felix: a high-throughput network approach for interfacing to front end electronics for atlas upgrades. In *Journal of Physics: Conference Series*, volume 664, page 082050. IOP Publishing, 2015.
- [15] Panagiotis Gkountoumis. Level-1 data driver card of the atlas new small wheel upgrade compatible with the phase ii 1 mhz readout scheme. In *Modern Circuits and Systems Technologies (MOCAST), 2016 5th International Conference on*, pages 1–4. IEEE, 2016.
- [16] J Wang. Trigger data serializer for the atlas nsw upgrade. Technical report, 2015.
- [17] Jinhong Wang, Liang Guan, Ziru Sang, JW Chapman, Tiesheng Dai, Bing Zhou, and Junjie Zhu. Characterization of a serializer asic chip for the upgrade of the atlas muon detector. *IEEE Transactions on Nuclear Science*, 62(6):3242–3248, 2015.
- [18] Ieee std 802.3-2012 - ieee standard for ethernet - section 4, 2012. URL <http://ieeexplore.ieee.org/document/6419738/>.
- [19] Kurt Tomlinson. Prbs (pseudo-random binary sequence), 2015. URL <http://blog.kurttomlinson.com/posts/prbs-pseudo-random-binary-sequence>.
- [20] Ug885: Vc707 evaluation board for the virtex-7 fpga user guide. URL [http://www.xilinx.com/support/documentation/boards\\_and\\_kits/vc707/ug885\\_VC707\\_Eval\\_Bd.pdf](http://www.xilinx.com/support/documentation/boards_and_kits/vc707/ug885_VC707_Eval_Bd.pdf).
- [21] Pg051: Logichore ip tri-mode ethernet mac v8.2 product guide. URL [http://www.xilinx.com/support/documentation/ip\\_documentation/tri\\_mode\\_ethernet\\_mac/v8\\_2/pg051-tri-mode-eth-mac.pdf](http://www.xilinx.com/support/documentation/ip_documentation/tri_mode_ethernet_mac/v8_2/pg051-tri-mode-eth-mac.pdf).
- [22] 1g eth udp / ip stack. URL [http://opencores.org/project,udp\\_ip\\_stack](http://opencores.org/project,udp_ip_stack).
- [23] R Pinkham. Ethernet protocol specification for tds test board. URL [https://docs.google.com/document/d/1cDEZK5L5lN8r\\_2Wef0Dqwynd\\_u3yw6t9ajlAR4eXAU8/edit?usp=sharing](https://docs.google.com/document/d/1cDEZK5L5lN8r_2Wef0Dqwynd_u3yw6t9ajlAR4eXAU8/edit?usp=sharing).
- [24] Pg057: Fifo generator v13.1 logichore ip product guide. URL [http://www.xilinx.com/support/documentation/ip\\_documentation/fifo\\_generator/v13\\_1/pg057-fifo-generator.pdf](http://www.xilinx.com/support/documentation/ip_documentation/fifo_generator/v13_1/pg057-fifo-generator.pdf).
- [25] Pyqt. URL <https://wiki.python.org/moin/PyQt>.
- [26] URL <http://www.ironwoodelectronics.com/>.

- 
- [27] Cdce62005 3:5 clock generator, jitter cleaner with integrated dual vcos. URL <http://www.ti.com/lit/ds/symlink/cdce62005.pdf>.
- [28] J Wang R Pinkham. Tds packaged chip test specifications. URL [https://docs.google.com/document/d/16Ehj\\_b9C0AaBa0jJCyjic9zo\\_5lvamrknoYCGm0gW1k/edit?usp=sharing](https://docs.google.com/document/d/16Ehj_b9C0AaBa0jJCyjic9zo_5lvamrknoYCGm0gW1k/edit?usp=sharing).