A Proposal for a DAQ System for the MATHUSLA Detector

L. Ruckman, J. Russell, C. Young, Z. Xu

October 11, 2021

Abstract

Text for this section is assigned to Charlie Young. We can write this abstract at the end when this document is more mature. **[CY: Abstract will be written later.]**

Contents

1 Introduction **5**

2 DAQ System Requirements **5**
2.1 Requirements . . . 6
2.1.1 SiPM Sensors . . . 6
2.1.2 Readout ASIC . . . 6
2.1.3 Hit/Trigger Network and Trigger Decision . . . 7
2.1.4 Timing Network . . . 8
2.1.5 Data Network . . . 8
2.1.6 CMS Network . . . 9
2.2 Amplitude Information to improve timing . . . 9
2.3 Product Lifetime and Mitigation Strategies . . . 9
2.4 Custom versus Commercial Solutions . . . 9

3 Hardware **9**
3.1 Overview . . . 9
3.2 Timing Distribution . . . 10
3.3 Front End Board (FEB) . . . 10
3.4 Link Aggregator Board . . . 11
3.4.1 Simple Link Aggregator . . . 12
3.4.2 Complex Link Aggregator . . . 12
3.5 Trigger Latency Estimates . . . 13
3.6 Bar Hit Rates . . . 14
3.7 Private Ethernet Network . . . 15
3.8 Server Farm . . . 15
3.9 Fiber Optic Networks and Transceivers . . . 16
3.10 Space Estimates . . . 17
3.11 Power/Cooling Estimates . . . 17

4 FPGA Firmware **17**
4.1 Revision Control . . . 17
4.2 AMBA Industry-Standard between Firmware modules . . . 17
4.3 SLAC Ultimate RTL Framework (SURF) . . . 18
4.4 Ethernet . . . 18
4.5 Timing Receiver . . . 18
4.6 Front End Board (FEB) . . . 19
4.6.1 ASIC Control/Monitoring . . . 19
4.6.2 Hit Primitive Messaging . . . 19
4.6.3 Data Primitive Messaging . . . 20
4.6.4 Programmable Logic Resource Estimations . . . 20
4.7 Layer Link Aggregator . . . 20
4.7.1 Channel Bonding Hit Messaging . . . 21
4.7.2 Programmable Logic Resource Estimations . . . 22
4.8 Tower Link Aggregator . . . 22
4.8.1 Channel Bonding Hit Messaging . . . 23
4.8.2 Channel Bonding Hit Messaging from Mesh Network . . . 23
4.8.3 3 x 3 Tower Trigger Decision . . . 23
4.8.4 Trigger Decision Messaging . . . 23
4.8.5 Programmable Logic Resource Estimations . . . 23
4.9 Trigger Link Aggregator . . . 23
4.9.1 Trigger Primitive Message Aggregation . . . 24
4.9.2 Programmable Logic Resource Estimations . . . 24
4.10 System Link Aggregator . . . 25
4.10.1 Timing Generator . . . 25
4.10.2 Trigger Decision Aggregation . . . 25
4.10.3 Programmable Logic Resource Estimations . . . 25

5 DAQ Software **26**
5.1 Revision Control 26
5.2 Rogue 26
5.3 Remote access to external software 26
5.4 RUDP to FPGA Networked Nodes 27
5.5 Channel Access to FPGA Networked Nodes 27
5.6 Asynchronous Data Collection in Software Ring Buffer 27

6 Prototyping **27**

7 Installation and Commissioning **28**

8 Costing **28**
8.1 M&S 28
8.2 Labor 28

9 Schedule **28**

10 Summary **28**

AC	Alternating current
ASIC	Application-Specific Integrated Circuit
CMS	Compact Muon Solenoid
CDR	Clock and Data Recovery
CRC	Cyclic Redundancy Check
DAQ	Data AcQuisition
DC	Direct Current
DHCP	Dynamic Host Configuration Protocol
DMA	Direct Memory Access
EPICS	Experimental Physics and Industrial Control System
FEB	Front End Board
FIFO	First In First Out
FPGA	Field-Programmable Gate Array
GbE	Gigabit Ethernet
GPS	Global Positioning System
GT	Gigabit Transceiver
GUI	Graphical User Interface
HL-LHC	High-Luminosity era of the Large Hadron Collider
HW	Hardware
ID	IDentification
IP	Intellectual Property (Firmware/Software Related)
IP	Internet Protocol (Ethernet related)
IP	Interaction Point (physics related)
JTAG	Joint Test Action Group
LCLS	Linac Coherent Light Source
LHC	Large Hadron Collider
LLP	Long Lived Particles
MAC	Media Access Control
MATHUSLA	MAsive Timing Hodoscope for Ultra Stable neutraL pArticles
M/S	Material and Supplies
OP-Code	OPerational Code
PGP	Pretty Good Protocol
PLL	Phase-Locked Loop
PPS	Pulse Per Second
PROM	Programmable Read-Only Memory
PV	Process Variable (related to EPICS)
QSFP	Quad Small Form-Factor Pluggable
RSSI	Reliable Slac Streaming Interface
RTL	Real Time Logic
RUDP	Reliable User Datagram Protocol
RX	Receiver
SACI	SLAC ASIC Control Interface
SiPM	Silicon Photomultiplier
SFP	Small Form-Factor Pluggable
SLAC	Stanford Linear Accelerator Center
SM	Standard Model
SRP	SLAC Register Protocol
SURF	SLAC Ultimate RTL Framework
SW	Software
TID	Technology Innovation Division
TID-ID	TID - Instrumentation Division
TID-ID-ES	TID-AIR - Electronic Systems
TCP	Transmission Control Protocol
TX	Transmitter
UDP	User Datagram Protocol

1 Introduction

The MAsive Timing Hodoscope for Ultra Stable neutraL pArticles (MATHUSLA) experiment has been proposed to operate during the high-luminosity era of the Large Hadron Collider (HL-LHC) at CERN to search for potential long lived particles (LLP) in their decay to two or more Standard Model (SM) particles. The detector is situated above ground to be sensitive to the SM final state while being well shielded from the harsh environment near the primary p-p collision point. It has a footprint of 100 m x 100 m, where the closest point is approximately 70 m longitudinally from the CMS interaction point (IP). Horizontally, the detector is made up of 100 identical "towers"[1] in a 10x10 grid. Each tower is made up of ten layers of scintillators, spanning 30 m vertically, subdivided into bars of approximately 5 cm wide and 5 m long. There are approximately 400,000 bars in the entire detector. Scintillation light is wavelength shifted and transmitted to both ends of each bar where silicon photomultipliers (SiPM) are located.

The spatial coordinates of a hit is given by the plane (for height), the bar (for width direction) and time difference between the two SiPMs (for length direction). Its time coordinate is given by the time average of the two SiPMs. The MATHUSLA detector design calls for a hit time resolution of 1 nsec (or better) and a longitudinal position resolution of $\sim$20 cm (or better).

This proposal is for the development of a Data AcQuisition (DAQ) system prototype that can eventually be scaled up to the full MATHUSLA detector. There are several features and requirements of MATHUSLA that makes it unusual:

- Signal events are expected to be extremely rare; indeed, there is no assurance that such LLP's exist. Scintillation rate is expected to be dominated by cosmic rays with an average rate of 35 Hz per scintillator bar. This rate is in turn dwarfed by the dark count rate (DCR) of SiPM in the 10K to 1M Hz range. In order to be sensitive to signal events, it is mandatory that deadtime from cosmic ray hits and dark counts be kept minimal.
- Hits from signal events do not have unique timing relationship to p-p collision time. Detector elements in MATHUSLA are $\sim$200 m from the IP, therefore an LLP with a velocity 90% the speed of light will arrive $\sim$60 nsec later than one traveling at speed of light. It is also 35 nsec and 10 nsec later than one traveling at speed of light but produced at the two subsequent beam crossings. The conventional ideas of time as relative to p-p collision time should be reexamined.
- Triggers corresponding to candidate LLP's are to be sent to CMS to record the full 4-pi event topology for a joint analysis. The maximum trigger latency, including all transit and processing times, is 9.0 microsec.
- The MATHUSLA detector is physically huge - it takes $\sim$1 microsec to go from one corner to the farthest away corner. This needs to be taken into account in the DAQ design.

2 DAQ System Requirements

We envision that *all* hits that are the coincidence of two ends of a scintillator bar are written out to software buffer storage without selection by trigger. This rate is dominated by cosmic rays and is easily within even today's recording technology for both rate and capacity.

In parallel, we envision hits are processed to form triggers. The signal trigger is two or more upward going tracks that come from a common 4-dimensional vertex in the MATHUSLA volume. There is also a calibration trigger of single upward going tracks (E.g. muon from CMS p-p collision). These triggers have to be sent back to CMS within their trigger latency of 9.0 microsec (measured from the time of p-p collision).

A software based high-level trigger will use hardware trigger times to select hits from software ring buffer storage and record in permanent storage. Given the size of MATHUSLA, the time window for each selection is O(1-2) microsec.

In addition, we record cosmic rays for monitoring and other purposes. We believe that no dedicated hardware trigger is necessary. Randomly selected time windows of the same O(1-2) microsec duration is sufficient.

The DAQ System requirements are the following:

[1]Tower can also refer to "module" detector. We use tower in this document because module detector can be confused with the generic engineering term of "module".

- Hit time must be measured with an RMS of < 1 nsec.
- Hit time difference must be measured with an RMS of < 1.5 nsec.
  - This is the expected uncertainty of subtracting two times each known to 1 nsec.
- Both hit time and hit time difference (equivalent to longitudinal position) must be available within the above stated RMS uncertainties
  - Offline analysis may lead to smaller RMS uncertainties - in time for trigger purposes
  - The time available for acquisition will depend on how we partition overall time budget among the steps such as DAQ, track finding, vertex fitting and so on
- The DAQ system should impose no deadtime beyond the double-pulse separation of an individual SiPM, expected to be O(20) nsec.

2.1 Requirements

2.1.1 SiPM Sensors

While the detector group does have ownership of the SiPM sensor design, we will introduce this sensor in this section and present the interface requirements for the DAQ's Front End Board (FEB).

The output of a SiPM is a signal with a rise-time of $\sim$1 nsec and a signal duration of $\sim$20 nsec with exponentially decaying falling edge. Typical SiPM sizes are few mm x few mm.

SiPMs are likely to be mounted either at the bars or on FEBs. There are design trade offs of where the SiPM would be mounted. SiPM on FEB reduces cabling to the bar but may be difficult to light shield the fiber from bar to FEB and attenuation lose over longer bar fibers

There is a voltage bias, usually operated at no higher than 50 V. This tunable bias voltage will likely be generated from either the FEB or from an external power supply source. For a group of SiPMs, tunable bias voltages may come of a common source (cost optimized but lower flexibility) or individually controlled (more flexibility but higher cost)

SiPM performance is sensitive to temperature and must be controlled (e.g. thermoelectric cooler). SiPM temperature stability control ($\pm$ °C) is to be determined. A temperature sensor must be placed as close as possible to the SiPMs for thermal control feedback. Readout of the temperature sensor and control of the thermoelectric cooler can come from either a commercial thermal controller or the FEB's FPGA.

2.1.2 Readout ASIC

The readout Application-Specific Integrated Circuit (ASIC) functional requirements are the following:

- ASIC noise *must* be small enough to allow clear observation of single pixel
- ASIC *must* have dynamic range of at least XXX pixels
- ASIC *must* be able to survive without being damaged from a discharge due to all pixels firing at the same time
- ASIC TDC's resolution (psec/count) must be small enough to not significantly contribute to the total RMS of $<$ 1nsec hit timing jitter
- ASIC *must* support multiple bar pair channels
- ASIC or FPGA must calculate (particle transit time) & (time difference of two SiPMs of the same scintillator bar).
  - The particle transit time should be given in a global time system (with a calibratable fixed offset bar-by-bar) with an RMS resolution of < 1 nsec.
  - Time difference should have an RMS resolution of < 1.5 nsec.
  - Support 50 nsec off overlay and correlation between channel pairs

- ASIC *may* provide amplitude information for the two SiPM signals that are in time coincidence.
- ASIC *must* be able to synchronize to timing system
- ASIC *must* have the ability to read/write internal registers from FPGA controller
- ASIC *must* be low power
- *Must* be able to operate from the coldest winters to hottest summers with respect to the MATHUSLA building's ambient air temperature
- ASIC performance likely to be sensitive to temperature and must be controlled (e.g. thermoelectric cooler)
- ASIC temperature stability control ($\pm$ °C) is to be determined
- Readout of the temperature sensor and control of the thermoelectric cooler can come from either a commercial thermal controller or the FEB's FPGA.

2.1.3 Hit/Trigger Network and Trigger Decision

The hit/trigger network and trigger decision requirements are the following:

- Able to trigger on upper going tracks (e.g. LLP event) and not trigger on downward tracks (e.g. cosmic rays)
- *Must* trigger on two or more upward going tracks that come from a common 4-dimensional vertex in the MATHUSLA volume
- *Must* trigger on single upward going tracks (E.g. muon from CMS p-p collision) for calibration
- All triggers will be readout locally but not all triggers will be sent to CMS
- CMS trigger *must* be rate limited and prescaled to protect against over triggering CMS
- *Must* be able to make a trigger decision and message CMS to save event in $<$ 9.0 microsec
- CMS Trigger message *must* include the LHC bunch crossing number(s) that we want them to record
- CMS Trigger message *must* locally correct for the bunch crossing offset with respect CMS
- *Must* distribute hit primitives between a 3 x 3 tower grid for the trigger decision
- *Must* automatically recover from a cosmic shower event when all sensors light up
- *Must* Able run even if there is an occasional dropped hit primitive packet due to CRC error or broken fibers

Each hit/trigger TX endpoint must also have a dedicated fiber output. Each hit/trigger RX endpoint must also have a dedicated fiber input. The hit primitive messages and trigger decision messages are unidirectional. Thus the hit/trigger network and timing network can be shared among the same bidirectional optical transceiver (e.g. SFP) but on separate fibers.

There is no length matching requirements in the hit/trigger network cabling because the trigger messages will be channel bonded together in firmware. However, it is important to keep the hit/trigger network cable length a reasonably short as possible because cable length reduces the system's total trigger latency budget.

2.1.4 Timing Network

The timing network functional requirements are the following:

- Distribute the timing across the entire MATHUSLA detector
- Likely to be phase locked to 40 MHz LHC clock and LHC timing system
- Support a standalone mode when testing without the actual LHC timing system
  - This can be achieved with local 40 MHz clock and timing system emulator firmware
- Each timing receiver endpoint must have a Clock and Data Recovery (CDR) circuit to recover the clock from the timing serial protocol
  - CDR's recovered clock used to synchronize the detector at the timing receiver endpoint
- Same length cable matching for all timing receiver endpoints to remove large systematic time offsets.

Each timing RX endpoint must also have a dedicated fiber input. Each timing TX endpoint must also have a dedicated fiber output. The timing messages are unidirectional. Thus the hit/timing network and trigger network can be shared among the same bidirectional optical transceiver (e.g. SFP) but on separate fibers.

The trigger decision firmware logic and system level calibration becomes a lot easier if each timing receiver endpoint have the same absolute T_0 time. This can be achieved by making the cable length to all the timing receiver endpoints the same length (as reasonably possible).

2.1.5 Data Network

We plan to implement the data network using standard Ethernet. By using common commercial Ethernet electronics, we can help keep the cost down.

The data network functional requirements are the following:

- 1 Gigabit (or faster) links for FPGA network nodes
- 10 Gigabit (or faster) links for server farm nodes
- FPGA communication to the server farm must be reliable either using Transmission Control Protocol (TCP) or Reliable User Datagram Protocol (RUDP)
- FPGA Ethernet communication need to support both channel access (A.K.A. peak/poke registers) and asynchronous message streaming
- FPGA's MAC address must be unique and stored using on-board non-volatile memory with write protection
- Server farm must support a DHCP server for FPGA IP address assignments
- There must be one and only one DHCP server within the data network that's hosted by the server farm
- FPGAs must support DHCP client
- Able to reprogram the FPGA boot image via Ethernet communication (A.K.A. remote reprogramming)
- Able to recover from a corrupted remote reprogramming event without JTAG intervention.
- Ancillary systems must also have Ethernet interfaces using either TCP, UDP or RUDP protocols
- This Ethernet network must be a private network that is not shared by offline data processing or public access.

- Access to the server farm must be behind a gateway computer to prevent unauthorized access to the DAQ system.
- Gateway computer must not have direct access to this DAQ private network and must go through one of the server farm nodes.

The FPGAs must have a dedicated SFP interface for Ethernet. This Ethernet interface can be either connected with fiber optics or copper cables. If copper, then the SFP-to-RJ45 adaptor must include magnetic isolation (prevent grounding loops). The fiber optic interface is required because some of the distances between network ports and the server farm will be >100 meters. Commercial Ethernet switches must be used to aggregate the FPGA links to the server farm. All ancillary systems (e.g. high voltage power supplies, low voltage power supplies, etc) must also have Ethernet interface.

2.1.6 CMS Network

The CMS interface is still under development with respect to HL-LHC. However, we know that we need to deliver the trigger over fiber. The trigger fiber must be either super high quality multi-mode fiber (OM4 or better) with multi-mode transceivers that support >250m range. Else a more expensive single-mode fiber with single-mode transceivers will need to be used. We also know that a LHC timing fiber will needed to deliver timing from CMS to MATHUSLA. Similar to the trigger fiber requirements, either multi-mode that supports >250m range or single-mode must be used.

2.2 Amplitude Information to improve timing

[LR: Amplitude information to improve timing (ToT, waveform capture, etc)]

2.3 Product Lifetime and Mitigation Strategies

[LR: Discuss how this is a long term project and need to keep in mind that any commerical solution or IC used needs to have a product lifetime such that it does not go obsolete (or does not have drop in replacement) from prototyping to installation to end of operations]

2.4 Custom versus Commercial Solutions

[LR: Discuss how we will use commercial solutions as much as possible. Custom solutions used when commercial solutions do not exists that meet our requirements or commercial solution requires other custom solutions to make them work (force design to "bend over backwards" to make a specific solution work and in return increase cost)]

3 Hardware

3.1 Overview

A block diagram of the DAQ networks is shown in Figure 1. Each ASIC services multiple SIPM sensors and exists on the FEB. We are assuming that each FEB services 40 bars (80 bar fibers) via these readout ASICs. Each layer has 400 bars and is serviced by 10 FEBs. Each tower has 10 layers. There are a total of 100 towers in the system.

At the FEB, the particle transit time and difference of two SiPMs of the same scintillator bar is timestamped and forwarded upstream as a message called "hit primitives" message on the hit primitives network. For each layer, the 10 hit primitives links from each FEB are aggregated together by a "link aggregator" board. There are many types of link aggregator boards in this DAQ system design. This specific link aggregator board is the "layer link aggregator".

For each tower, there are 10 layer link aggregator outputs. These 10 links are aggregated together by a "tower link aggregator". The tower link aggregator is where the 3x3 tower trigger decisions happens. The nearest neighbor towers transmit their local hit primitives and receive remote hit primitives using a mesh network. The output of the tower link aggregator is the 3x3 trigger decisions.

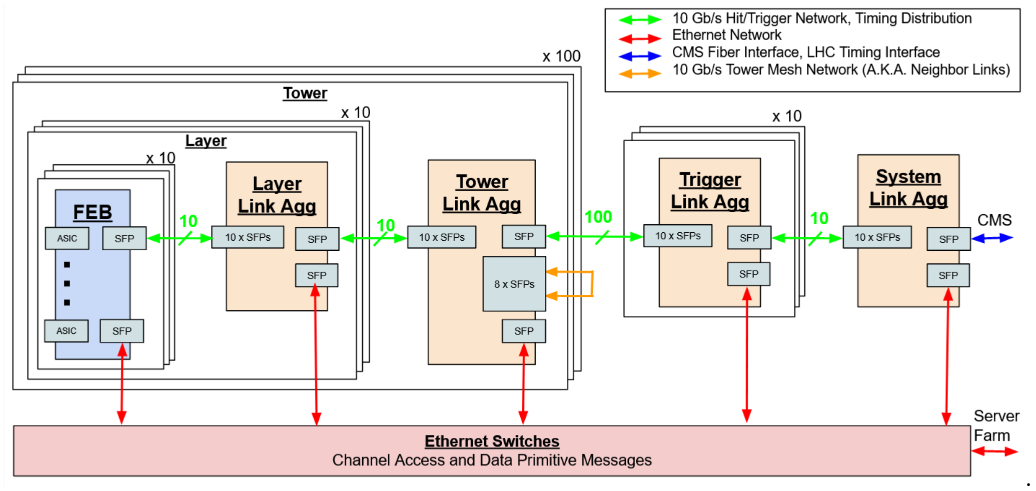

Figure 1: DAQ Networks Block diagram

These 100 trigger decisions links are aggregated together by a ”trigger link aggregator” into 10 links. These 10 links are then aggregated together by a ”system link aggregator”. The system link aggregator collects all the trigger decisions and forms the final event trigger message that is sent to both CMS and MATHUSLA server farm.

The channel access and streaming data primitives from each of the FPGA boards are connected to the server farm nodes via standard Ethernet network.

The hit primitive network uses a unidirectional, point-to-point connections. The timing network uses fan out ICs at the link aggregators to distribute the timing to the FEBs. The data network uses commercial Ethernet switches to aggregate the slow FEB links into faster server node links. The mesh network uses high speed, bidirectional links that moves hit primitives messages between adjacent towers to form the 3x3 trigger decision. A diagram of this mesh network is shown in Figure 2.

3.2 Timing Distribution

We plan to distribute the LHC timing to all FPGA nodes in the system. The link aggregator will include fanout ICs to repeat the serialized protocol to the FEBs and one copy sent to the local FPGA for synchronization. It will be likely that we will need to length match the timing fibers such that T_0 is the same for all FEBs. The reason for this is that it will be difficult to calibrate out and require the calibration to be applied in real-time for the trigger decision firmware (increased firmware complexity and increased FPGA resources).

During initial testing and development, the external LHC timing will not be available. This is why we are planning to implement a LHC timing generator at the system link aggregator. The system link aggregator can select to use either the external LHC timing system or local emulation based on local 40 MHz clock to distribute to the MATHUSLA.

3.3 Front End Board (FEB)

A block diagram of the FEB hardware is shown in Figure 3. There are two SFPs. One SFP for connecting to the layer Link aggregator and the other to the server farm private Ethernet network. The timing system’s recovered clock from the CDR will be sent to a discrete jitter cleaner PLL to generate multiple clock frequencies all phased aligned with the timing system. The jitter cleaner clock outputs will have jitter on the order $\sim$1 ps-RMS. We are assuming that one FEB will service up to 40 bars (80 SiPM channels). There will be either 1 or more ASICs to service all the SiPM interfaces. If the SiPM exist on the FEB, then this ”SiPM interface” will be the light fiber from the bar. If the SiPM does not exist on the FEB, then this ”SiPM interface” will be the electrical output of the SiPM, SiPM voltage bias and thermal controls. A design study is necessary to determine if the SiPM should exist on the FEB or not. The board power will come from an isolated supply. Likely this isolated supply will exists in the rack.

Here’s a break down of the number of FEBs for this system:

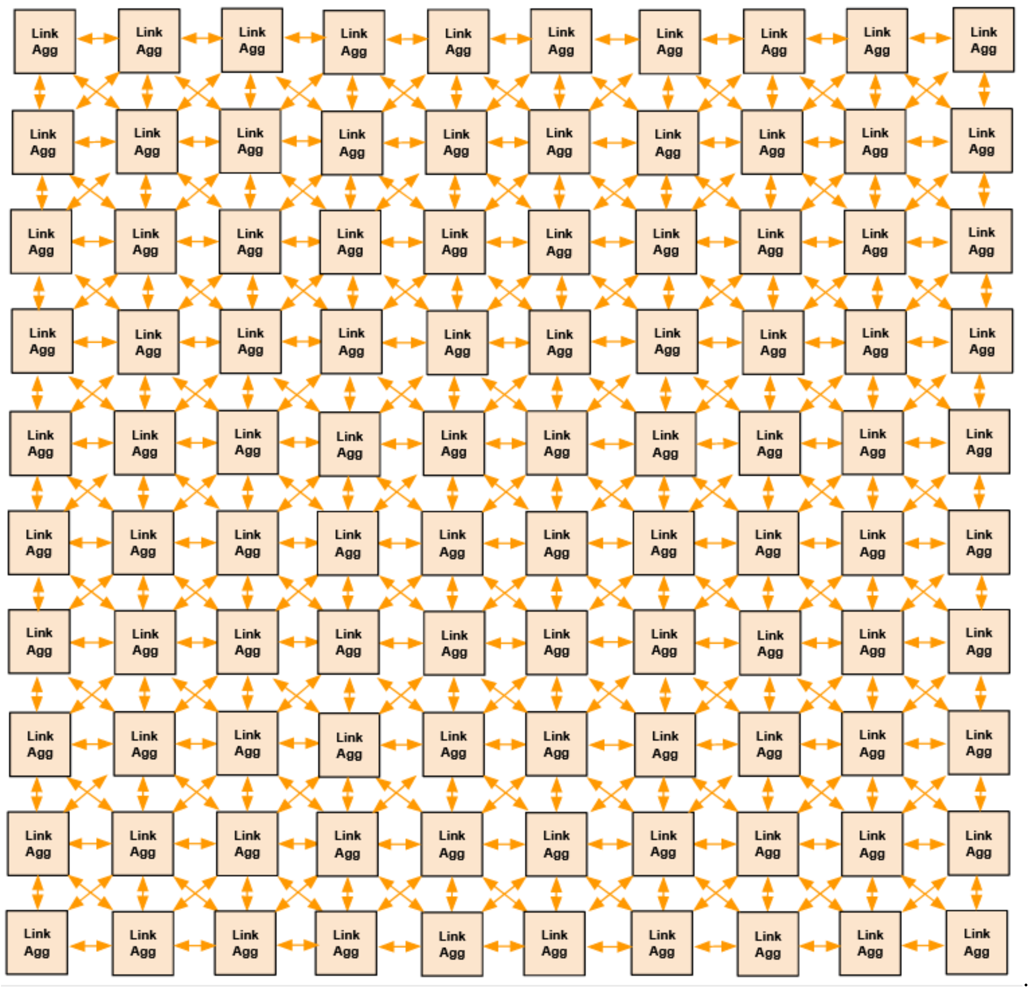

Figure 2: Tower's Hit Primitive Mesh Network Block diagram

- 10 FEBs per layer
- 100 FEBs per tower
- 10,000 FEBs for entire system

3.4 Link Aggregator Board

There are four types of Link Aggregator modules in this design: layer, tower, trigger, system. Within a given layer, the "layer aggregator" aggregates the hit primitive links from the FEBs to the tower Aggregator. Within a given tower detector, the "tower aggregator" aggregates hit primitives from the layer aggregators, creates the 3x3 hit primitive mesh network and makes the 3x3 decision trigger decision. The "trigger aggregator" aggregates the trigger decision messages from the tower aggregator to the system aggregator. The "system aggregator" aggregates the trigger decision messages from the trigger aggregator and form the system readout trigger that's sent to CMS and to server farm for data ring buffer readout. Here's a break down of the number and type of link aggregator boards for this system:

- Layer Link Aggregators:
  - 1 Layer aggregator per layer
  - 10 Layer aggregator per tower
  - 1,000 Layer aggregator for entire system
- Tower Link Aggregators:

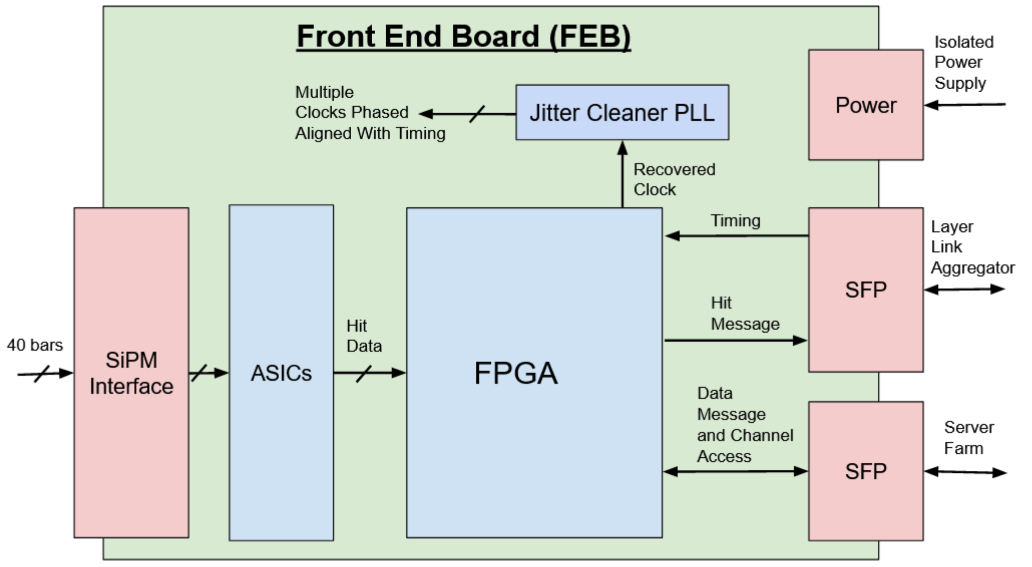

Figure 3: Block Diagram of the FEB hardware

  - 1 Tower aggregator per tower
  - 100 Tower aggregator for entire system
- Trigger Link Aggregators:
  - 10 Trigger aggregator for entire system
- System Link Aggregators:
  - 1 System aggregator for entire system

We expect that there will be two types of link aggregators board designs: Simple Link Aggregator & Complex Link Aggregator.

We are planning that both simple and complex designs will be rack mountable modules. Having the link link aggregator boards located in a rack will make the cabling easier to manage (especially for the mesh network). Alternatively, you could have the link aggregator board located next to the detector tower that it is servicing but the "rat nest" of cabling for the 3x3 tower mesh network would make maintenance difficult for the FEBs and bars. Another alternative is to only have the layer aggregator located at the layer of the tower and all other aggregator modules located in the racks.

3.4.1 Simple Link Aggregator

A block diagram of the simple link aggregator board hardware is shown in Figure 4. The simple link aggregator board design is be a cost optimized FPGA board design that requires minimal amount of resources and high speed interconnects. We are targeting a Artix Ultrascale+[1] for this FPGA (Part Number XCAU25P), which is the same targeted FPGA as the FEB. The layer and trigger link aggregators could share this common "simple" hardware design. There will be 10 SFPs for collecting hit/trigger primitive messages and transmitting the timing. 1 SFP to transmitting the aggregated hit/trigger primitive messages. 1 SFP for connecting to the server farm's private Ethernet network. All hit/trigger SFPs will operate at 10 Gb/s switching speeds. The Ethernet SFP can either operate at 1 Gb/s or 10 Gb/s. The both board power will come from an isolated supply. Likely this isolated supply will exists in the rack.

3.4.2 Complex Link Aggregator

A block diagram of the complex link aggregator board hardware is shown in Figure 5. The complex link aggregator will have higher cost than simple design, but support a bigger FPGA

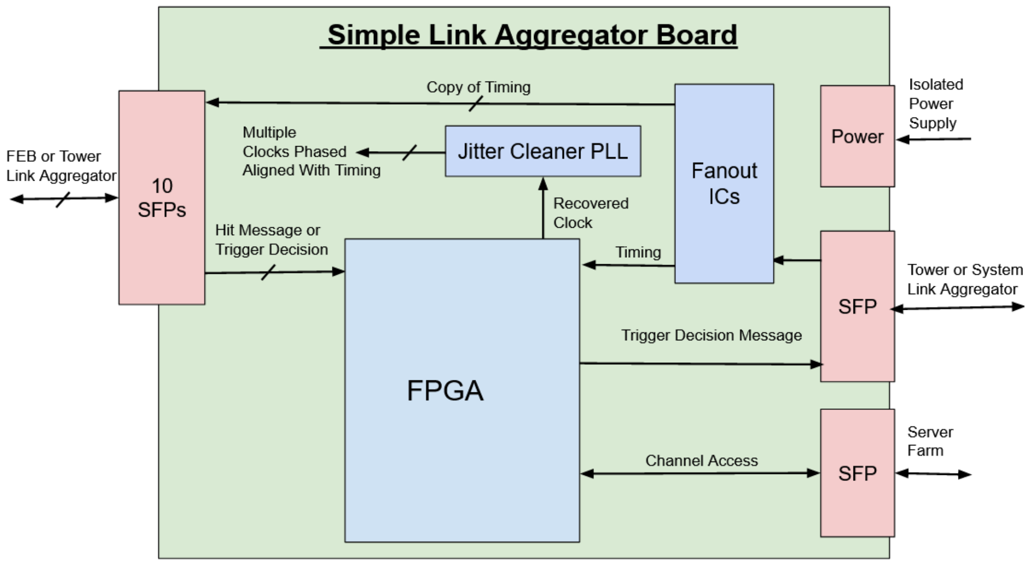
Simple Link Aggregator Board
Copy of Timing
Multiple Clocks Phased Aligned With Timing
Jitter Cleaner PLL
Fanout ICs
Power
Isolated Power Supply
FEB or Tower Link Aggregator
10 SFPs
Recovered Clock
Hit Message or Trigger Decision
Timing
SFP
Tower or System Link Aggregator
Trigger Decision Message
FPGA
Server Farm
Channel Access
SFP

Figure 4: Block Diagram of the simple Link Aggregator Board hardware

for the additional FPGA resources required for mesh network and 3x3 tower trigger decision. We are targeting a Kintex Ultrascale+[3] for this FPGA (Part Number XCKU15P). We expect the tower link aggregator to use this complex link aggregator design. We also expect that system link aggregator will be a complex link aggregator to support the FPGA resources required for forming the trigger and stand alone timing generator emulation. There will be 10 SFPs for collecting hit/trigger primitive messages and transmitting the timing. 1 SFP to transmitting the trigger decision (or CMS message). 1 SFP for connecting to the server farm's private Ethernet network. 8 SFPs for connecting to the 3x3 tower mesh network. All hit/trigger SFPs will operate at 10 Gb/s switching speeds. The Ethernet SFP can either operate at 1 Gb/s or 10 Gb/s. The both board power will come from an isolated supply. Likely this isolated supply will exists in the rack.

3.5 Trigger Latency Estimates

We have performed a design study of the trigger latency for this DAQ system and present the results in Table 1. We are assuming that the speed of light inside the optical data fiber is 2.14 x 10^8 m/s (4.67 nsec/m) and the speed of light inside the wavelength shifting bar fiber is O(15-16) cm/nsec.

We have classified the estimates into three types: calculation, simulation and measurement.

- *Calculation* type is any latency estimate based on first order principles calculations and not derived from computer simulation model or empirical measurement, *e.g.* calculate a fiber latency by knowing the length and speed of light in fiber.
- *Simulation* type is any latency estimate based on a computer simulation model and/or a firmware simulation testbed to determine the latency through a firmware module.
- *Measurement* type is any latency estimate based on the component, sub-system block and system block *e.g.* measuring latency on a fiber cable with a pulse laser, photo-diode and oscilloscope.

Measurement types have the highest confidence level for accuracy while "calculations" have the lowest confidence level.

There is a lot of detail in the table, which is explained later in this paper. The key point is that satisfying the 9.0 microsec latency requirements appears to be possible. Where possible, additional work after the TDR should be done to fully simulate or measure the latency of each element (instead of calculating) to increase confidence level of estimate accuracy.

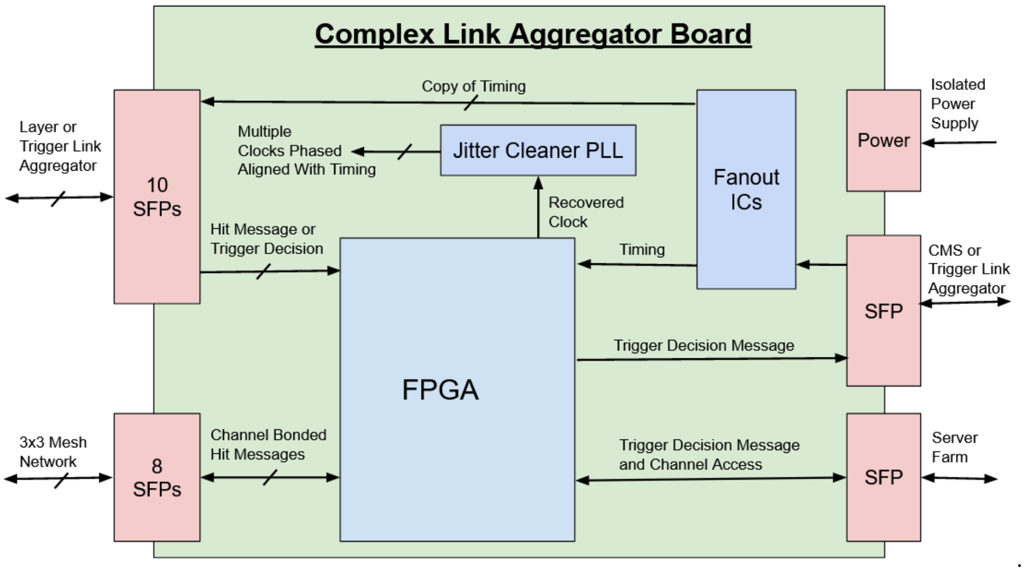
Complex Link Aggregator Board
Copy of Timing
Multiple Clocks Phased Aligned With Timing
Jitter Cleaner PLL
Fanout ICs
Power
Isolated Power Supply
Layer or Trigger Link Aggregator
10 SFPs
Recovered Clock
Hit Message or Trigger Decision
Timing
CMS or Trigger Link Aggregator
SFP
FPGA
Trigger Decision Message
3x3 Mesh Network
8 SFPs
Channel Bonded Hit Messages
Trigger Decision Message and Channel Access
SFP
Server Farm

Figure 5: Block Diagram of the complex Link Aggregator Board hardware

Table 1: Trigger Latency Budget

Item	Location	Latency Description	Minimum (microsec)	Maximum (microsec)	Estimate Type
1	Particle Track	CMS p-p to MATHUSLA	0.363	1.314	Calculation
2	Cabling	Mean Hit Time in Bar to SiPM	0.070	0.100	Calculation
3	FEB	SiPM to ASIC, 1 meters	0.005	0.005	Calculation
4	FEB	ASIC to FPGA, 1 meters	0.005	0.005	Calculation
5	FEB	TDC + Time Coincidence	0.031	0.031	Simulation
6	FEB	Hit Primative Message	0.200	0.400	Simulation
7	FEB	Event Building Hits Messages	0.400	0.400	Simulation
8	FEB	PGPv4 Protocol + GT TX	0.131	0.134	Simulation
9	Cabling	Hit Primative cable, 15 meters	0.070	0.070	Calculation
10	Layer Link Agg	PGPv4 Protocol + GT RX	0.131	0.134	Simulation
11	Layer Link Agg	Event Building Hits Messages	0.013	0.213	Simulation
12	Layer Link Agg	PGPv4 Protocol + GT TX	0.131	0.134	Simulation
13	Cabling	Hit Primative cable, 1 meters	0.005	0.005	Calculation
14	Tower Link Agg	PGPv4 Protocol + GT RX	0.131	0.134	Simulation
15	Tower Link Agg	Event Building Hits Messages	0.013	0.213	Simulation
16	Tower Link Agg	PGPv4 Protocol + GT TX	0.131	0.134	Simulation
17	Cabling	Mesh Network cable,15 meters	0.070	0.070	Calculation
18	Tower Link Agg	PGPv4 Protocol + GT RX	0.131	0.134	Simulation
19	Tower Link Agg	Event Building Mesh Network	0.000	0.000	TBD
20	Tower Link Agg	3x3 Tower Track Messages	0.000	0.000	TBD
21	Tower Link Agg	PGPv4 Protocol + GT TX	0.131	0.134	Simulation
22	Cabling	Trigger Decision cable, 150 meters	0.701	0.701	Calculation
23	Trigger Link Agg	PGPv4 Protocol + GT RX	0.131	0.134	Simulation
24	Trigger Link Agg	Event Build Track Messages	0.013	0.213	Simulation
25	Trigger Link Agg	PGPv4 Protocol + GT TX	0.131	0.134	Simulation
26	Cabling	Trigger Decision cable, 1 meters	0.005	0.005	Calculation
27	System Link Agg	PGPv4 Protocol + GT RX	0.131	0.134	Simulation
28	System Link Agg	Event Build Track Messages	0.013	0.213	Simulation
29	System Link Agg	10x10 Tower Vertex Trigger Module	0.000	0.000	TBD
30	System Link Agg	Trigger Decision Messaging	0.000	0.000	TBD
31	Cabling	CMS Trigger copper cable, 150 meters	0.782	0.782	Calculation
			Total Min (microsec)	Total Max (microsec)	
			4.068	6.081	

3.6 Bar Hit Rates

Table 2 shows the true particle hit rate and the hit rates from random coincidence of SiPM that we design towards for this DAQ system. Particle hit rate in each Bar is dominated by cosmic rays at 35 Hz, and the rate from LHC collisions is negligible in comparison. Random coincidence of dark counts from the two SiPM's on a Bar is calculated as the 40-nsec coincidence time window, corresponding to the light transit time over the length of the Bar, multiplied by the square of a single SiPM's dark count rate (DCR). We assume that this random coincidence gives the same rate as the true particle hit rate, which implies that SiPM DCR is 24 kHz. Therefore, the total coincidence rate requirement is 70 Hz per Bar. Table 3 shows Ethernet streaming bandwidths for the data primitives. These low bandwidth requirements shows how there is more than enough bandwidth to support both streaming data and channel access on the FEB using a 1 GbE connection. With a 1 GbE at the FEB, we calculated that we could support up to 2.4 MHz SiPM dark count rate (230 kHz coincidence rate), which would be 890 Mb/s at the FEB for streaming the data primitives.

However running at such a high dark rate will have a cascading effect on the DAQ system:

- Makes the 3x3 trigger decision difficult (if not impossible to do) in real-time
- Significantly increase the server node cost (89 Gb/s ring buffer per tower server node)
- Significantly increase the Ethernet networking cost

So the sensor group and DAQ group will need to work closely with each other to optimize the sensor/DAQ design with respect to sensor performance versus. 3 x 3 tower trigger decision efficiency versus DAQ M/S cost.

Table 2: Expected SiPM hit rates

	End(kHz)	Bar(Hz)	FEB[kHz)	Layer(kHz)	Tower.Trig(kHz)	Tower.Data(kHz)
SIPM Dark Rate	30.0	36.0	1.4	14.4	86.4	144.0
Cosmic		35.0	1.4	14.0	84.0	140.0
Total		71.0	2.8	28.4	170.4	284.0

Table 3: Ethernet Streaming Bandwidths for the Data Primitives based on Table 2

	Bar(kb/s)	FEB[Mb/s)	Layer(Mb/s)	Tower(Mb/s)	System(Gb/s)
Data Primitive Streaming	6.8	0.3	2.7	27.3	2.7

3.7 Private Ethernet Network

To keep the system cost down we plan to use commercial Ethernet electronics to transport the data primitives from the FEB and channel access to all network nodes in the system. Figure 6 is an example of a possible Ethernet switch topology. In this topology, we would have a "low speed" and a "high speed" Ethernet switch.

An example of a low speed switch is the Fiberstore's S3910-24TS [2] (24-Port 1GbE + 4-port 10GbE). Such a low speed Ethernet switch would service 20 FEB links, 2 layer link aggregators and switch monitor link. It would require 5 low speed Ethernet switches per tower for a total of 500 switch modules. The tower link aggregator would connect to 1 of the 5 tower's low speed Ethernet switches.

An example of a high speed switch is the Fiberstore's N5860-48SC [3] (48-Port 10GbE + 8-port 40/100GbE). Such a high speed Ethernet switch would service 40 x 10GbE tower links (5 links per tower), 8 x 40/100 GbE server node links (1 link per server node) and switch monitor. Because this high speed switch service 8 tower, it would only require 13 switch modules to server all 100 towers.

3.8 Server Farm

We are planning the server farm to be a cluster of 1U server nodes. Each server node would have the following specs:

- More than 128 hyperthreads
- More than 32 GB RAM
- Diskless booting
- Hard drive (256 GB or better) for data primitive ring buffer only
- 40 Gb/s (or better) Ethernet port to the detector private network

[2]https://www.fs.com/products/108712.html
[3]https://www.fs.com/products/110478.html

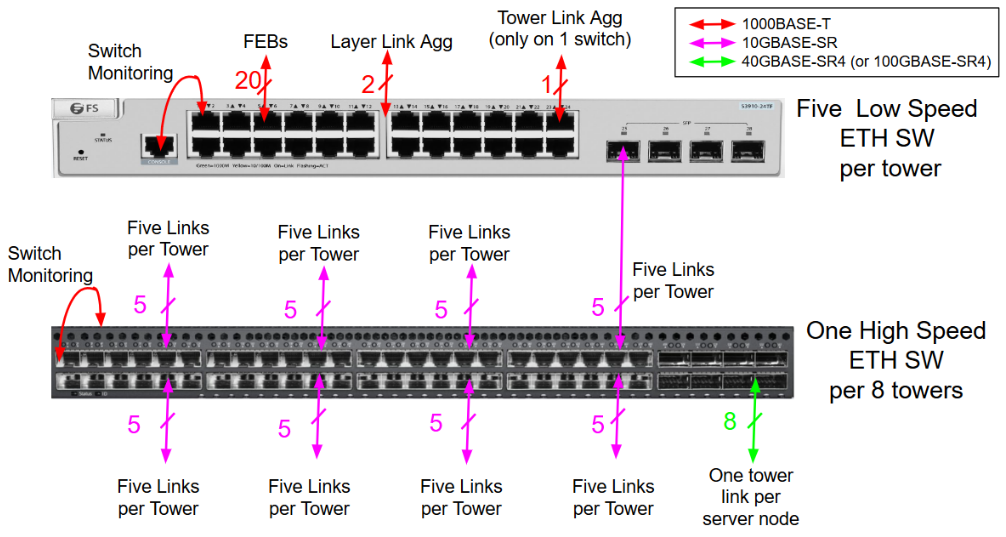
1000BASE-T
10GBASE-SR
40GBASE-SR4 (or 100GBASE-SR4)
Switch Monitoring
FEBs
20
Layer Link Agg
2
Tower Link Agg (only on 1 switch)
1
Five Low Speed ETH SW per tower
Switch Monitoring
Five Links per Tower
5
One High Speed ETH SW per 8 towers
8
One tower link per server node

Figure 6: Example of a possible Ethernet Switch Topology

- 1 Gb/s (or better) Ethernet port to the detector's gateway computer

Hard drive should have good write endurance but does not requires high throughput. So a slow SATA (instead of fast solid state drive) could be used. The 128 hyperthreads (or more) will be necessary if we want to run 100 (or more) RUDP links into a single server node (refer to Section 5.4). It will be likely that we will partition the system into 1 sever node per detector tower (about 100 FEB RUDP links per server node). This architecture is also flexible and scalable that we can upgrade to more than 1 server node per detector tower if needs be.

3.9 Fiber Optic Networks and Transceivers

The fiber networks (timing, trigger and CMS) need to have the following characteristics:

- Low Cost density ($ per channel)
- Lower power density (mW per channel)
- High channel density (channel per optical transceiver)
- Support cable lengths greater 150 meters but less than 400 meters

We performed a survey of possible options and list the results in Table 4. During this investigation, we assume to only use multi-mode 850nm short range wavelength optics because single-mode fiber would significantly increase the cost of the DAQ system. We also assume that we should only consider optical transceivers that are not obsolete (e.g. SNAP12) or end-of-life (e.g. microPOD).

We identified that QSFP+ has the best cost density and would be ideal for the link aggregator board. Using a QSFP between multiple SFP endpoints would require either a patch panel or a custom MPO-to-LC breakout cable. For cable planning, it might not make sense to use QSFP in the mesh network where there are lots of connects that cannot easly be fiber bundled to the same location. In this case multiple SFPs would be a prefer option with a slightly increase cost density with respect to QSFP+ for the link aggregator board.

CXP would satisfy the range requirement but factor of ∼4 in channel density cost with respect to QSFP+. Firefly is also factor of ∼4 in channel density cost with respect to QSFP+ and does not satisfy the range requirement.

For the FEB boards, the hit primitive SFP would need to be the SFP+ (SFP-10GSR-85) instead of SFP (like SFP1G-SX-85) because of the 10 Gb/s requirement. The FEB's Ethernet SFP could be SFP (like SFP1G-SX-85) if using fiber optics for the 1 Gb/s ETH links.

Table 4: Survey of Possible Optical Transceivers

Type	lanes	Range	Data Rate	Cost	Cost Density	Manufacture	Part Number
SFP	1	550m	1.25 Gb/s	$7	$7/lane	fs.com	SFP1G-SX-85
SFP+	1	300m	10 Gb/s	$20	$20/lane	fs.com	SFP-10GSR-85
QSFP+	4	400m	10 Gb/s	$65	$16/lane	fs.com	QSFP-CSR4-40G
Firefly	12	100m	14 Gb/s	$900	$75/lane	Samtec	ECUO-Y12-14
CXP	12	300m	11 Gb/s	$830	$69/lane	Finisar	FTLD10CD3C

3.10 Space Estimates

[LR: This can be written up at the end of the design or FY22 funding There are basically two volumes that we should talk about. First is the FEB space requirements at the detector. Send the number of racks and rack space requirement.]

3.11 Power/Cooling Estimates

[LR: This can be written up at the end of the design or FY22 funding Talk about the FEB power and cooling. Talk about the link agg power and cooling in the racks. Talk about the server nodes and other rack modules power and cooling. Probably add a table that breaks power the power estimates and total power required to operate the DAQ.]

4 FPGA Firmware

4.1 Revision Control

All firmware will be revision controlled using git. We plan to host this git repository on Github. The register mapping in software to the firmware registers will be located in the firmware git repository. Keeping the software mapping and firmware in the same git repository prevents the mapping from getting out of sync with respect to the actual firmware source code. We plan to have the system DAQ git repository separate to the firmware repository. The system DAQ git repository will include the firmware repository as tagged release submodule.

”ruckus” is an open-source firmware build system that is supported and maintained by the SLAC's TID-ID-ES division. Here's a list of some important functions that this build system provides for building FPGAs:

- Include the 160-bit git hash into firmware register space
- Includes a build string firmware register
- Build string includes project's name, build time, Vivado version, and user who built it
- Help with tag releasing a project
- Auto-generated tag release notes

For previous experiments, we have found the ability to access the git hash from firmware register is very important in debugging what the actual state of the programmed firmware is and traceability back to the original source code. This build system is a public git repository and can be found on Github:

- `https://github.com/slaclab/ruckus`

4.2 AMBA Industry-Standard between Firmware modules

AMBA stands for ARM Advanced Microcontroller Bus Architecture. AMBA Advanced eXtensible Interface (AXI) is the common interface used on majority of Xilinx IP cores. There are 3 types of AXI interfaces (AXI, AXI-Lite, AXI stream). AXI is a high performance memory interface and used to move data in/out of memory (example: writing data to the DRAM). AXI-Lite is a low performance, single transaction, 32-bit memory interface and used for all the firmware modules' slow controls and monitoring. AXI stream is a streaming data interfaces with flow control and is used to move stream data between the different firmware modules.

One of the most common issues that comes up when multiple firmware developers are all working on the same FPGA target is defining the interfaces between firmware modules. We plan to use only AXI type interfaces between different firmware modules in all our FPGA targets. This will prevent custom interface types between interfaces and force developers to only use well defined industry-standard interface types.

4.3 SLAC Ultimate RTL Framework (SURF)

SLAC Ultimate RTL Framework (SURF) is an open-source firmware library that is supported and maintained by the SLAC's TID-ID-ES division. This firmware library provides the base firmware and framework for many commonly used firmware module:

- Ethernet Library (1000BASE-KX, 10Gbase-KR, XAUI, IPv4, ARP, DHCP, ICMP, UDP)
- AXI4 Library (Crossbar, DMA, FIFO, etc.)
- AXI4-Lite Library (Crossbar, AXI4-to-AXI4-Lite bridge, etc.)
- AXI4-Lite Library (DMA, MUX, FIFO, etc)
- Device Library (ADI, Linear, Micron, TI, etc.)
- Synchronization Library (Synchronize bits, buses, vectors, resets, etc)
- Wrapped Xilinx Library (clock managers, SEM, DNA, IPROG)
- Serial Protocols Library (I2C, SPI, UART, G-Link, JESD204B, etc)
- All of SLAC custom Protocols
  - `https://confluence.slac.stanford.edu/display/ppareg/SLAC+Protocols`

This firmware library helps with rapid prototyping, and has been deployed in many experiments (e.g. LCLS-II, LSST, Heavy Photon Search, ATLAS CSC upgrade, proto-DUNE, etc). This firmware library is a public git repository and can be found on Github:

- `https://github.com/slaclab/surf`

4.4 Ethernet

SLAC has created a reliable communications protocol which resides on top of UDP for communications between FPGAs and software as well as between two FPGAs. This Reliable Slac Streaming Interface (RSSI)[5] is based upon existing RUDP standards[7],[6],[2] and enables reliable communication of framed data with up to 256 virtual channels. RSSI is used as the primary interface protocol for many of its FPGA applications and is core to the LCLS-2 control system. RSSI is provided in the SURF firmware library (refer to Section 4.3). RSSI + UDP create the Reliable UDP (RUDP) communication.

We plan is to use RUDP on all our FPGA Ethernet network nodes. We are expecting that 1 server node in the server farm will manage all the FPGA nodes in a tower (about 100 nodes).

4.5 Timing Receiver

Every FPGA node will have access to the timing network and have a local timing receiver. This common timing receiver will be used to synchronize the FPGA to the timing system. This timing receiver provides run control signals (start of run, stop of run, periodic strobes, etc) as well. The timing receiver will use a Gigabit Transceiver (GT) to deserialize the timing serial stream. The GT's integrated Clock and Data Recovery (CDR) will be used to recover the clock from the stream stream. This recover clock will have large jitter ($\sim$50 ps-RMS). To remove this jitter and generate other clock frequencies that are all phase aligned to the timing system, we will use a jitter cleaner Phase-Locked Loop (PLL) (e.g. SiLab Si5345).

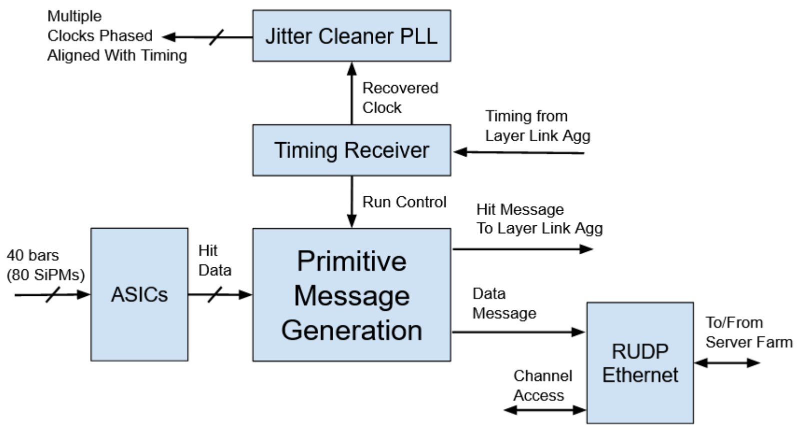

Figure 7: Front End Board (FEB) Firmware Block diagram

4.6 Front End Board (FEB)

An overview firmware block diagram is shown in Figure 7. The purpose of this firmware is to readout the timestamped hit coincidence data from the ASIC. This data is formatted into "hit primitives" for triggering and "data primitives" for data logging. There is a local timing receiver for synchronizing to the timing system. The Ethernet communication to this module is RUDP for channel access and streaming of the "data primitives" to the server farm.

4.6.1 ASIC Control/Monitoring

The ASIC will have a "memory map" like register access for channel access. Channel access is required for loading configurations into the ASIC and monitoring the ASIC's state of operation. We plan to implement this ASIC interface using SLAC ASIC Control Interface (SACI)[8]. SACI is a simple serial interface for configuring registers and sending commands to an ASIC. The firmware to interface with SACI is provided in the SURF firmware library (refer to Section 4.3).

4.6.2 Hit Primitive Messaging

We plan to transmit a hit primitive message periodically every 150 nsec. The message will compose of a header, payload and footer. The hit message will always be sent even if there is no hit data. In this case where there's no hit data, only the header and footer are sent. The header will be a 64-bit timestamp. This timestamp will be used to channel bond multiple hit messages together at the link aggregator. The footer likely will be small (like 8-bits) and used to forward metadata (e.g. too much data to fit within a 150 ns hit messaging window).

The message payload will contain "hit words". A "hit word" is 32-bit word that contains the following data fields:

- BIT(7:0): Average Hit time (8-bits)
  - Relative to timestamp header
  - In units of 1/1.28GHz
- BIT(14:8): Bar position (7-bits)
  - Position in the bar
  - In units of 1/1.28GHz
- BIT(24:16): Bar Index (9-bits)
  - 0 to 399
- BIT(30:28): Layer Index (3-bits)
  - 0 to 5

We plan to serialize the hit message using Pretty Good Protocol Version 4 (PGPv4)[4]. PGPv4 provides 64B66B encoding framing, CRC checking and useful diagnostics for debugging a flaky link. PGPv4 is provided in the SURF firmware library (refer to Section 4.3). We plan to run the PGPv4 links at 10 Gb/s.

A firmware simulation of a PGPv4 link running at 10 Gb/s has been performed. The result of this simulation shows that we can support up to 38 hit words in a given hit primitive message (sent every 150 nsec).

4.6.3 Data Primitive Messaging

Unlike the hit primitives that are packaged together and forwarded every 150ns, the data primitive will be sent on the private Ethernet network when ever they occur. The data primitives do not have the real-time and low latency requirements that the hit primitives have. This makes it possible to have the data primitives in a more "buffered-like" implementation.

A "data word" is 16-byte (128-bit) word that contains the following data fields:

- 64-bit absolute hit time
  - In units of 1/1.28GHz
- 7-bit delay line tap number
  - Position in the bar
  - In units of 1/1.28GHz
- 8-bit Time Over Threshold for SiPM pair 1 of 2
- 8-bit Time Over Threshold for SiPM pair 2 of 2
- 9-bit bar number (0 to 399)
- 4-bit layer number (0 to 9)
- 7-bit tower number (0 to 99)

To increase the efficiency of the Ethernet transport of the data primitives from the FEB to the server node, we plan to batch up multiple 12-byte data words into a single 1kB RUDP payload frame (up to 85 hit words). If the FEB hit rate is ∼3 kHz, the Ethernet frame rate would be ∼ 35 Hz and bandwidth of ∼288 kb/s. So the average bandwidth is much, much lower than the 1 Gb/s Ethernet links. However there will be cosmic shower events for a short period of time that might exceed the 1 Gb/s Ethernet link bandwidth. For these events, we plan to have a large FPGA buffer between the hit primitive generation and the RUDP firmware. If the FPGA buffer is not big enough to ride out the cosmic burst event while simultaneous streaming out on RUDP, then an overflow counter will increment for each data word that is dropped. This overflow counter would be monitored by the higher level software to gauge the health of the system and quality of the data taking.

4.6.4 Programmable Logic Resource Estimations

Table 5 shows the FPGA resource estimates. We are assuming an Artix Ultrascale+[1] (Part Number: XCAU25P) for these estimates.

4.7 Layer Link Aggregator

An overview firmware block diagram is shown in Figure 8. The purpose of this firmware is to aggregate and channel bond the 10 FEB hit primitive links into a singular hit primitive link towards the tower link aggregator. There is a local timing receiver for synchronizing to the timing system. The timing system repeaters are discrete circuits external of the FPGA but controlled by this firmware. The Ethernet communication to this module is RUDP for channel access only (no streaming data).

Table 5: FEB Programmable Logic Resource Estimations

Firmware Modules	LUT(K)	FF(K)	BRAM36	URAM	DSP48	GT	Estimate Confidence
RUDP+DHCP+MAC+PHY	16.5	28.5	17	0	6	1	HIGH
SRPv3	1.4	2.1	4	0	0	0	HIGH
SACI	0.2	0.2	0	0	0	0	HIGH
Timing Receiver	3	4	0	0	0	0.5	MEDIUM
PLL SPI w/ BOOT ROM	0.1	0.1	1	0	0	0	HIGH
Hit Primitive Generator	10	10	1	0	0	0	LOW
Data Primitive Generator	10	10	100	0	0	0	LOW
PGPv4 (TX only)	3.2	3.6	0	0	0	0.5	HIGH
Estimate Total	44.4	58.5	123.0	0.0	6.0	2.0	
Estimate Percentage	31	21	41	0	1	17	
Total Available	141	282	300	0	1200	12	

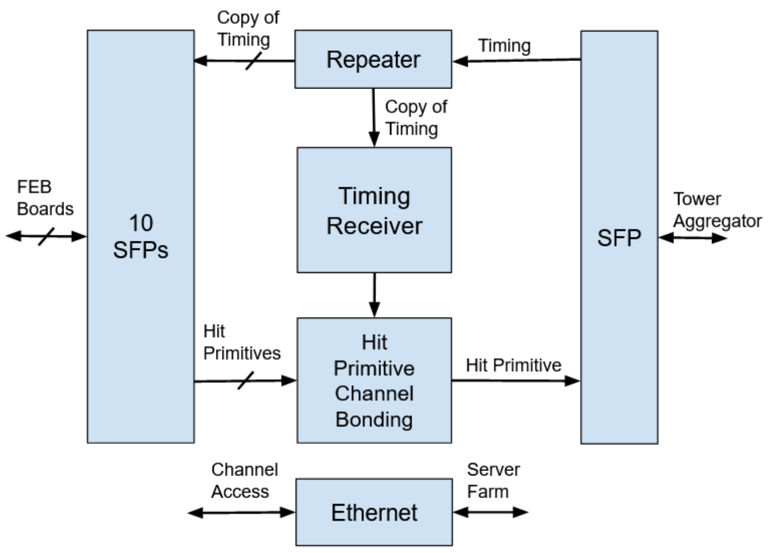

Figure 8: Layer's Link Aggregator Firmware Block diagram

4.7.1 Channel Bonding Hit Messaging

The primary function of this firmware module is to take multiple inbound streams and combine them together into a singular bound stream. It needs to be low latency and able to aggregate the inbound streams synchronously to the timing system. In addition it needs to support real-life situations where the messages is dropped due to bit error(s) on the transport or lost link connections (e.g. FEB is powered off). It will increment a status counter any time a message is dropped (one status counter per inbound stream). These status counters will be monitored by high level software to gauge the health of the system and quality of the data taking.

Hit primitives messages are sent from FEBs periodically at 150 ns. The FEBs use the timing system to synchronize the periodic update. This means the message header's timestamp will be the same for all FEBs at the time of publishing the message.

The local FPGA is also synchronize to the timing system and aware of the timestamp for each 150 ns update message. The local periodic timestamp will need a programmable calibrated offset to deskew its local timestamp used for channel bonding with respect to the latency from transporting the message from the source (FEB) to this destination. This local periodic timestamp is compared with all the inbound streams'. If all the inbound streams' timestamp match the local timestamp, then the channel bonding firmware knows all the inbound messages are synchronized and can aggregate all the inbound messages into a singular outbound message.

In real-life there will be situations where the inbound stream(s) are dropped due to bit error(s) detect by CRC check or lose of connection. The channel bonding firmware has to wait up to a programmable time (less than 1 periodic refresh cycle) for that inbound stream(s) to present its timestamp. If the inbound stream(s) does not publish a timestamp, then the channel bonding firmware will ignore that inbound stream(s) and form the outbound message with all the other inbound stream(s). If the inbound stream(s) has a timestamp that's ahead in time or behind in

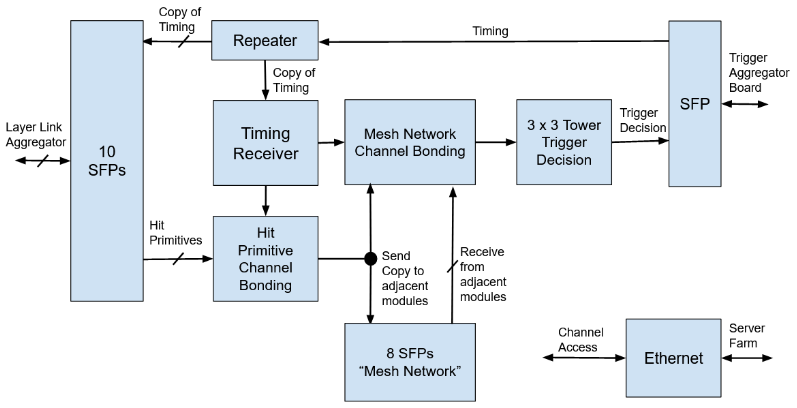

Figure 9: Tower's Link Aggregator Firmware Block diagram

time with respect to local timestamp, the channel bonding firmware will drop that out-of-sync inbound message(s) and will not be used outbound message.

The singular outbound message will have the same data format as the inbound message. The channel bonding forces the headers to all be the same and the inbound footers will be OR'd together to make the outbound footer (pass along the metadata). This means that it can only support up to 38 hit words in a given 150 ns periodic refresh. If there are more than 38 hits words, then some of the inbound hit words will need to get drop. When this happens, the channel bonding firmware will set a bit in the outbound footer to let the upstream module know that data was drop and increment a status counter for the higher level software to monitor.

4.7.2 Programmable Logic Resource Estimations

Table 6 shows the FPGA resource estimates. We are assuming an Artix Ultrascale+[1] (Part Number: XCAU25P) for these estimates.

Table 6: Layer Link Aggregator Programmable Logic Resource Estimations

Firmware Modules	LUT(K)	FF(K)	BRAM36	URAM	DSP48	GT	Estimate Confidence
RUDP+DHCP+MAC+PHY	16.5	28.5	17	0	6	1	HIGH
SRPv3	1.4	2.1	4	0	0	0	HIGH
Timing Receiver	3	4	0	0	0	0.5	MEDIUM
Hit Channel Bonding	10	10	12	0	20	0	LOW
10 x PGPv4 (RX only)	29	45	10	0	0	10	HIGH
1 x PGPv4 (TX only)	3.2	3.6	0	0	0	0.5	HIGH
Estimate Total	63.1	93.2	43.0	0.0	26.0	12.0	
Estimate Percentage	45	33	14	0	2	100	
Total Available	141	282	300	0	1200	12	

4.8 Tower Link Aggregator

An overview firmware block diagram is shown in Figure 9. The purpose of this firmware is to aggregate and channel bond the 10 hit primitive links from the layer link aggregators into a singular hit primitive link (A.K.A "local hit primitive link"). One copy of this local hit primitive link is sent to the mesh network channel bonding firmware module. Another copy is transmitted on all 8 of the mesh network SFPs. The received hit primitives links are sent to the mesh network channel bonding firmware module. This channel bonding module synchronizes the local hit primitive link with the 8 remote hit primitive links. The output of this channel bonding module is sent to the 3 x 3 tower trigger decision module. This trigger decision module looks for tracks across a 3 tower x 3 tower grid. If a trigger event is detected, the trigger decision module will form a trigger decision primitive.

There is a local timing receiver for synchronizing to the timing system. The timing system repeaters are discrete circuits external of the FPGA but controlled by this firmware. The Ethernet communication to this module is RUDP for channel access only (no streaming data).

4.8.1 Channel Bonding Hit Messaging

This firmware module is identical to layer Aggregator's channel bonding (refer to Section 4.7.1).

4.8.2 Channel Bonding Hit Messaging from Mesh Network

This firmware module is "similar" to layer Aggregator's channel bonding (refer to Section 4.7.1) but not "identical". The primary difference between the hit primitive channel bonding and the mesh network channel bonding is that the links do not get merge together into a singular stream. Instead they synchronized together and forwarded in 9 parallel streams to the 3x3 tower decision firmware module. The mesh network still has to handle drop messages or lost connections like the hit primitive channel bonding does. When a drop message condition happens, a "empty" hit primitive message (only a header and footer but no hit words) will be sent. The footer will have a bit set to inform the trigger decision module that it was a "filler" message and hit primitive information may have been lost.

4.8.3 3 x 3 Tower Trigger Decision

@JJ: Discuss the following:

0) Should we enforce some kind of ordering in the hit channel bonding. If yes, then how would you like it ordered to prevent your module from having to do a "sorting" process (increased latency). If no, describe your "sorting" process and gotchas for latency.

1) double caching to create 300 ns window

2) Receiving 9 parallel 64-bit AXI streams from mesh network

3) only trigger on upward going tracks (software for cosmic event triggering

4) how you will trigger on two or more upward going tracks that come from a common 4-dimensional vertex in the MATHUSLA volume (CMS LLP triggers).

5) how you will trigger on single upward going tracks (E.g. muon from CMS p-p collision) for calibration(local triggers and not sent on CMS trigger network)

6) Do you want to sent an trigger decision every 150 ns? Or only sent trigger decision when a trigger exists?

4.8.4 Trigger Decision Messaging

@JJ/Larry: Need to define the trigger primitive messaging format. Talk about how the message this be serialized and sent to trig agg.

4.8.5 Programmable Logic Resource Estimations

Table 7 shows the FPGA resource estimates. We are assuming an Kintex Ultrascale+[3] (Part Number: XCKU15P) for these estimates.

4.9 Trigger Link Aggregator

An overview firmware block diagram is shown in Figure 10. The purpose of this firmware is to aggregate 10 tower trigger decision primitive links into a singular trigger decision primitive link towards the system link aggregator. There is a local timing receiver for synchronizing to the timing system. The timing system repeaters are discrete circuits external of the FPGA but controlled by this firmware. The Ethernet communication to this module is RUDP for channel access only (no streaming data).

Table 7: Tower Link Aggregator Programmable Logic Resource Estimations

Firmware Modules	LUT(K)	FF(K)	BRAM36	URAM	DSP48	GT	Estimate Confidence
RUDP+DHCP+MAC+PHY	16.5	28.5	17	0	6	1	HIGH
SRPv3	1.4	2.1	4	0	0	0	HIGH
Timing Receiver	3	4	0	0	0	0.5	MEDIUM
10 x PGPv4 (RX only)	29	45	10	0	0	10	HIGH
Hit Channel Bonding	10	10	12	0	20	0	LOW
8 x PGPv4 (RX+TX)	38.4	47.2	8	0	0	8	HIGH
Mesh Channel Bonding	25	25	96	0	18	0	LOW
3x3 trigger decision							LOW
1 x PGPv4 (TX only)	3.2	3.6	0	0	0	0.5	HIGH
Estimate Total	126.5	165.4	147.0	0.0	44.0	20.0	
Estimate Percentage	24	16	15	0	2	71	
Total Available	523	1045	984	128	1968	28	

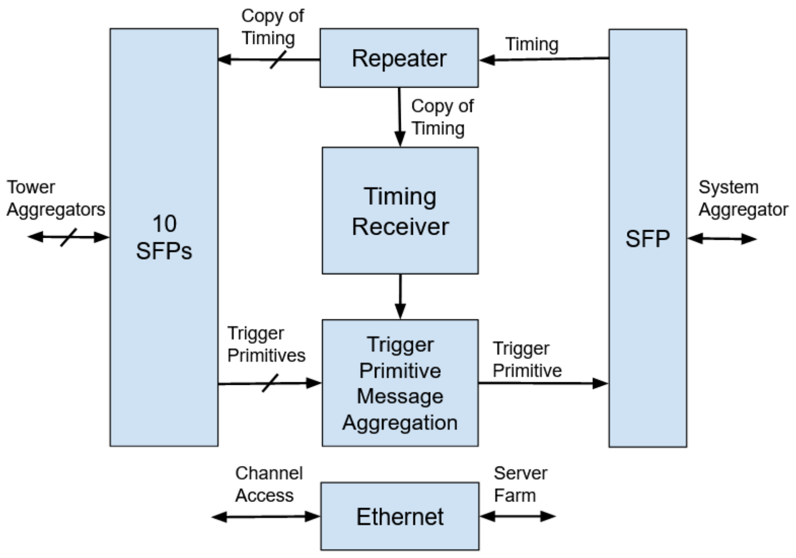

Figure 10: Trigger's Link Aggregator Firmware Block diagram

4.9.1 Trigger Primitive Message Aggregation

This firmware module is similar to hit primitive channel bonding (Refer to Section 4.7.1) except that it does not channel bond and forwards the trigger primitive upstream as soon as possible.

4.9.2 Programmable Logic Resource Estimations

Table 8 shows the FPGA resource estimates. We are assuming an Artix Ultrascale+[1] (Part Number: XCAU25P) for these estimates.

Table 8: Trigger Link Aggregator Programmable Logic Resource Estimations

Firmware Modules	LUT(K)	FF(K)	BRAM36	URAM	DSP48	GT	Estimate Confidence
RUDP+DHCP+MAC+PHY	16.5	28.5	17	0	6	1	HIGH
SRPv3	1.4	2.1	4	0	0	0	HIGH
Timing Receiver	3	4	0	0	0	0.5	MEDIUM
Trigger Decision Aggregation	10	10	12	0	20	0	LOW
10 x PGPv4 (RX only)	29	45	10	0	0	10	HIGH
1 x PGPv4 (TX only)	3.2	3.6	0	0	0	0.5	HIGH
Estimate Total	63.1	93.2	43.0	0.0	26.0	12.0	
Estimate Percentage	45	33	14	0	2	100	
Total Available	141	282	300	0	1200	12	

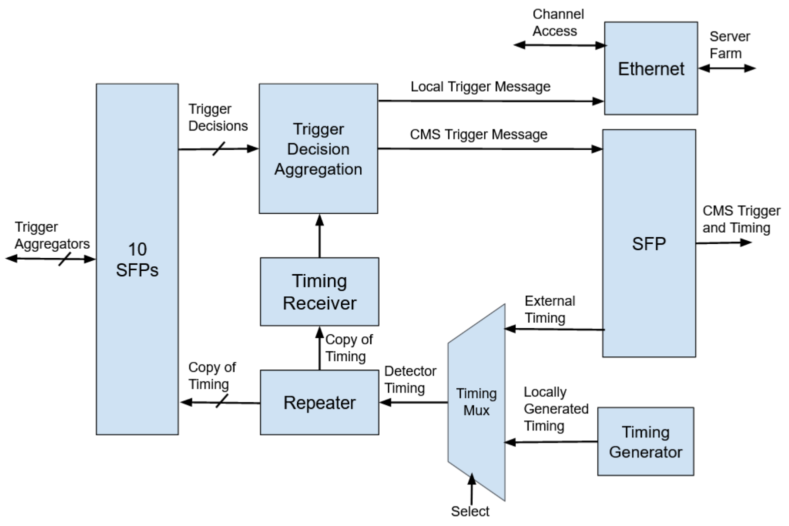

Figure 11: System's Link Aggregator Firmware Block diagram

4.10 System Link Aggregator

An overview firmware block diagram is shown in Figure 11. The purpose of this firmware is to aggregate the 10 trigger decision primitive links from the trigger link aggregators. This module will make the final trigger readout decision and publish both the CMS trigger message and the local trigger readout message. Standalone mode operations without external LHC timing network is achieved by emulating the timing system with a local timing generator firmware module. There is a local timing receiver for synchronizing to the timing system. The timing system repeaters are discrete circuits external of the FPGA but controlled by this firmware. The Ethernet communication to this module is RUDP for channel access and streaming of the "trigger readout message" to the server farm.

4.10.1 Timing Generator

The primary function of this firmware module is to provide a substitute timing system when the LHC timing network is not available. Example of this would be testing the electronics at a home university or home institution (offsite of CERN). The system link aggregator will have a multiplexer to select whether distributing the local timing generator or the external LHC timing network to all the downstream link aggregators and FEBs. This timing generator would need to emulate of all the run controls and periodic operation codes that the actual LHC timing network provides.

4.10.2 Trigger Decision Aggregation

The primary function of this firmware module is to aggregate the 10 trigger decision primitive links from the trigger link aggregators and make the final decision for the DAQ readout trigger. It will format the CMS trigger message and format the local trigger message sent to the server farm. This module will need to be capable to rate limited and prescale to protect against over triggering CMS. This module will have a programmable bunch crossing offset with respect CMS and will include the LHC bunch crossing number(s) that we want them to record. The exact message format for both the CMS trigger and local server node trigger is to be determined.

4.10.3 Programmable Logic Resource Estimations

Table 9 shows the FPGA resource estimates. We are assuming an Kintex Ultrascale+[3] (Part Number: XCKU15P) for these estimates. We have selected the same big FPGA that is used on the tower link aggregators because there is only 1 system link aggregator for the entire system and has highest risk of feature creep (new features require more resources).

Table 9: System Link Aggregator Programmable Logic Resource Estimations

Firmware Modules	LUT(K)	FF(K)	BRAM36	URAM	DSP48	GT	Estimate Confidence
RUDP+DHCP+MAC+PHY	16.5	28.5	17	0	6	1	HIGH
SRPv3	1.4	2.1	4	0	0	0	HIGH
Timing Receiver	3	4	0	0	0	0.5	MEDIUM
Trigger Decision Aggregation	10	10	12	0	20	0	LOW
10 x PGPv4 (RX only)	29	45	10	0	0	10	HIGH
Timing Generator	3	4	0	0	4	0.5	MEDIUM
CMS Trigger Message	10	10	0	0	0	1	LOW
Estimate Total	72.9	103.6	43.0	0.0	30.0	13.0	
Estimate Percentage	14	10	4	0	2	46	
Total Available	523	1045	984	128	1968	28	

5 DAQ Software

@JJ: We can finish writing this using FY22 funding.

Discuss the overall SW arch. Do we plan to use EPICS and use rogue underneath? If not EPICS, what DAQ SW arch? Discussion on real-time processing and offline processing. Where is the offline data storage located?

5.1 Revision Control

Similar to the firmware revision control (refer to Section 4.1), the software will also be revision controlled via git and hosted on Github. This software git repository will include all of the source code required to run the DAQ. It will not include any user analysis scripts, which should be in a different git repository. It will not include configuration management and configuration files, which should also be in a different git repository (or some other managed software environment). The reason for not including these items is to make the release process of the DAQ software more manageable and not dependent on frequently changing user code or configuration parameters.

5.2 Rogue

The SLAC TID-ID Electronics group has created a mixed Python/C++ platform to service as both a hardware abstraction layer and a lightweight Data Acquisition System call Rogue. Unlike many current similar platforms which wrap python around existing C++ software or are written entirely in Python, Rogue is architected to allow the appropriate mix of C++ and Python, allowing high performance multi-threaded processing in the C++ layer while still utilizing the flexibility and dynamic nature of Python.

Rogue allows the numerous modules and registers in a larger hardware system to be organized into trees using Python classes. This organization simplifies the interaction with hardware while allowing complex initialization and operation state machines to be defined at various levels. Registers in the hardware are abstracted as a set of variables and commands which are exposed to the user, as well as to numerous, simultaneous external managed systems including EPICS, Python scripts, custom GUIs and third party DAQ systems. The Rogue architecture allows the user to create data processing plug in modules written either in C++ or Python. This allows for quick prototyping of processing modules in Python with the ability to later move that processing to C++ if required for performance.

This software library is a public git repository and can be found on Github:

- `https://github.com/slaclab/rogue`

5.3 Remote access to external software

The text below is old and out-of-date. We no longer use Pyro4 and use ZeroMQ instead. Ryan is working on updating the documentation on the github repo. After his updates the documentation, we can copy the text from there into this section for most up-to-date description.

Rogue variables and Commands may be accessed directly in the local python instance, using the Python Remote Objects (Pyro4) protocol, or through a third party control layer using a plugin. Currently plugins exist for EPICS V3 channel access and EPICS V4 PV access protocols, as well as a Mysql gateway. A dynamic GUI is provided which can be attached directly to the server instance or attached remotely using Pyro4. A shared memory interface is also supported.

Rogue Variables and Commands may also be accessed through a C++ API which wraps around the Rogue Python structures, allowing the the Rogue platform to be integrated into any C++ based data acquisition system.

5.4 RUDP to FPGA Networked Nodes

Similar to SURF firmware library including RUDP IP core (refer to Section 4.4), rogue software library (refer to Section 5.2) includes an API for connecting and streaming data to/from an FPGA RUDP network node. This API is available in both a C++ or python. The SURF/rogue implementation of RUDP supports up to 256 virtual channels per connection.

However, we are planning to not use the virtual channel feature and use two separate RUDP connections per FPGA node. One RUDP connection for channel access only and the other connection for streaming FEB data primitives messages only. While it is possible to put both channel access and streaming data on the same RUDP connection via virtual channelization feature, we run the risk that channel access software process might back pressure the link and prevent streaming data from flowing to the streaming data software process. If the FEB runs out of buffering due to back pressure for too long of a duration on the RUDP link, then the FEB has to drop valuable physics data. This is why we are planning to use two separate RUDP connections per FPGA node.

5.5 Channel Access to FPGA Networked Nodes

The channel access to the FPGA nodes is achieved by adding a remote messaging protocol layer on top of the RUDP layer. We will use SLAC Register Protocol Version 3 (SRPv3) [9] for this remote messaging protocol layer. SRPv3 supports AXI4-Lite and AXI4 memory transactions in the FPGA firmware. It supports single read/write transactions of a single 32-bit FPGA register and supports bursts of up to 4kB read/write transactions of contiguous memory. SRPv3 is a fairly mature protocol (developed in 2016). The SRPv3 IP core firmware is provide by SURF (refer to 4.3). The SRPv3 software API is provide by rogue (refer to Section 5.2) and available in both a C++ or python.

5.6 Asynchronous Data Collection in Software Ring Buffer

Each FEB data primitive is 12 bytes (Section 4.6.3) for $\sim$3.5 Mbytes / sec, not a big deal nowadays. We can buffer $\sim$40 hours of hit data in a 500 GB disk. These numbers are not intended to say that we WILL buffer for such a long time. Triggers are written to disk in an independent stream. A High Level Trigger (HLT) process will decide which triggers will send the data in buffer to permanent storage. The algorithm will be something like:

- all upward going triggers
- Pre-scale of downward cosmic triggers
- possibly randoms

The HLT process uses the trigger time stamps to select ALL hits, meaning across the entire detector array and not just the triggering module/tower, within a $\pm$ 1 micro-sec window. This data is then written out to permanent storage.

The impact on buffer storage is that we need to support a read rate in addition to the write rate. The total cosmic ray rate is $\sim$2 MHz. I believe that 1 percent, i.e. 20 kHz, is plenty for calibration and alignment purposes. Single muons from LHC is 2x10^8 for 3000 fb^{-1} – see Section 4.2 in https://www.overleaf.com/project/5de13e248ea0320001de9d2c. Assuming this luminosity is integrated over 10 years of operations with 30 percent running time, the rate is 2x10^8 / (10 x 0.3 yr) = 2. Hz. This is sub-dominant by several orders of magnitude.

6 Prototyping

@All: We can finish writing this using FY22 funding

7 Installation and Commissioning

@All: We can finish writing this using FY22 funding

8 Costing

@All: We can finish writing this using FY22 funding

8.1 M&S

@All: We can finish writing this using FY22 funding

8.2 Labor

@All: We can finish writing this using FY22 funding

9 Schedule

@All: We can finish writing this using FY22 funding

10 Summary

@Charlie: We can finish writing this using FY22 funding

References

[1] *Artix Ultrascale+ Product Page.* URL: `https://www.xilinx.com/products/silicon-devices/fpga/artix-ultrascale-plus.html`.

[2] *Draft IEFT Sigtran Reliable UDP.* URL: `https://tools.ietf.org/html/draft-ietf-sigtran-reliable-udp-00`.

[3] *Kintex Ultrascale+ Product Page.* URL: `https://www.xilinx.com/products/silicon-devices/fpga/kintex-ultrascale-plus.html`.

[4] *Pretty Good Protocol Version 4 (PGPv4).* URL: `https://confluence.slac.stanford.edu/x/1dzgEQ`.

[5] *Reliable SLAC Streaming Protocol (RSSI).* URL: `https://confluence.slac.stanford.edu/x/1IyfD`.

[6] *RFC-1151 Version 2 of the Reliable Data Protocol.* URL: `https://tools.ietf.org/html/1151`.

[7] *RFC-908 Reliable Data Protocol.* URL: `https://tools.ietf.org/html/rfc908`.

[8] *SLAC ASIC Control Interface (SACI).* URL: `https://confluence.slac.stanford.edu/x/YYcRDQ`.

[9] *SLAC Register Protocol (SRP) Version 3.* URL: `https://confluence.slac.stanford.edu/x/cRmVD`.