Data Formatter Design Specification DRAFT version 0.1

Jamieson Olsen[1], Tiehui Ted Liu[1], Bjoern Penning[1], and Ho Ling Li[2]

[1]Fermi National Accelerator Laboratory, Batavia, Illinois 60510, USA
[2]University of Chicago, Chicago, Illinois 60637, USA

October 6, 2011

Abstract

Collisions in the LHC occur at the nominal rate of 40MHz with a design luminosity of 1×10^{34} and approximately 25 overlapping proton-proton interactions per crossing. The ATLAS detector trigger system must reject a vast majority of these events, and only 200 events per second can be stored for later analysis.

An upgrade to the LHC is in the planning stages. Instantaneous luminosity is expected to increase to 3×10^{34} with an average of 75 proton-proton interactions per crossing. Under these conditions the existing ATLAS trigger is strained and the need for a tracking trigger is clear. The Fast Tracker (FTK) proposal involves adding a hardware-based level-2 track trigger to the ATLAS DAQ system. The FTK proposal includes a Data Formatter system to remap the ATLAS inner detector geometry to match the FTK $\eta - \phi$ towers. The Data Formatter system also performs pixel clustering and data sharing in overlap regions.

This design specification describes the Data Formatter system in detail and chronicles the "bottom up" approach to hardware design. Based on the current design requirements and the need for future expansion capabilities, a full mesh backplane interconnect is a natural fit for the Data Formatter design. Our final design also works well as a general purpose FPGA-based processor board. The Data Formatter may prove useful in scalable systems where highly flexible, non-blocking, high bandwidth board to board communication is required.

Operated by Fermi Research Alliance, LLC under Contract No. De-AC02-07CH11359 with the United States Department of Energy.

Contents

List of Figures

List of Tables

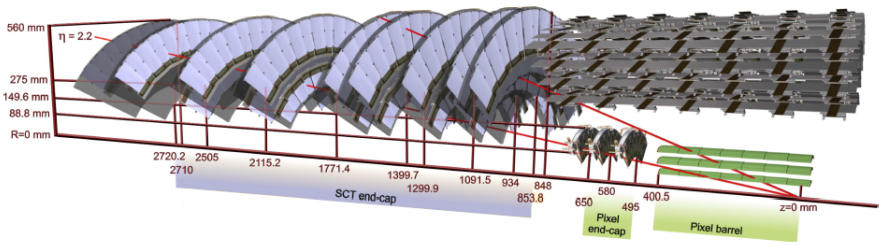

Figure 1: The ATLAS inner detector modules.

1 The LHC and ATLAS Detector

The Large Hadron Collider (LHC) at CERN will extend the frontiers of particle physics with its unprecedented high energy and luminosity. Inside the LHC, bunches of up to 10^{11} protons will collide 40 million times per second to provide 14 TeV proton-proton collisions at a design luminosity of 1×10^{34} cm^{2}s^{-1}.

Two general purpose detectors, ATLAS (A Toroidal LHC ApparatuS) and CMS (Compact Muon Solenoid) have been built for probing proton-proton and heavy ion collisions. Currently the ATLAS DAQ system has no low level hardware-based track trigger.

An upgrade to the LHC is in the planning stages. Instantaneous luminosity is expected to increase to 3×10^{34} with an average of 75 proton-proton interactions per crossing. At these higher luminosity levels the need for a low level track trigger becomes apparent.

1.1 Inner-Detector Sensors and Modules

The ATLAS inner detector is shown in Figure 1. Silicon sensors are used to construct the Pixel and SCT detectors. The Pixel detector is located close to the interaction region and has the highest resolution and highest "hit" occupancy. The SCT detector covers a larger area and has a lower resolution, utilizing silicon micro-strips instead of square pixels.

The Pixel detector is composed of three barrel layers (radius 50 to 123mm) and six end-cap disks (at z = 495 to 650mm). All of the 1,744 pixel modules (external dimensions 19×63mm) are identical and consist of 47,232 pixels per module for a total of 82,372,608 pixels.

The SCT detector is composed of four barrel layers (radius 299 to 514mm) and 18 end-cap disks (at z = 954 to 2720mm). SCT modules consist of two stereo layers. The SCT barrel sensors measure 63×63mm and consist of 768 80μm strips. Four SCT barrels are constructed from 2,112 sensors mounted to "stave" support structures. SCT end-cap sensors are trapezoidal shaped and come in three varieties: inner modules measure $45 \times 55 \times 61$mm (inner width, outer width, length); middle modules measure $55 \times 75 \times 119$mm; and outer modules measure $56 \times 72 \times 123$mm. A total of 1,976 modules are used to construct the SCT end-cap disks.

1.2 Readout Drivers

Inner detector front end electronics are implemented in radiation-hardened ASIC chips which are mounted to the modules. These front end ASICs interface to the silicon sensors and incorporate analog circuitry to amplify the signals and compare the signal level against a programmable threshold. Digital logic in the ASIC stores the "hit" pixel coordinates, as well as a time stamp and amplitude (time over threshold) in a buffer, which is read out following a L1 trigger.

Subdetector	Partition	Modules	ROLs
Pixel	Barrel 0	286	44
	Barrel 1	494	38
	Barrel 2	676	26
	End-Cap A	144	12
	End-Cap C	144	12
SCT	Barrels A	1056	22
	Barrels C	1056	22
	End-Cap A	988	23
	End-Cap C	988	23

Table 1: Data Formatter input readout links

Chains of front end ASICs are connected over fiber optic links to the Readout Driver (ROD) electronics, which are located off-detector. RODs receive serialized data from the detector after a L1 trigger and are responsible for de-serializing the data, error checking, local event building and data monitoring tasks.

Each ROD services up to 48 SCT modules or between 6 and 26 Pixel modules (determined by occupancy). Table 1 shows the number of modules and readout links for the Pixel and SCT detectors.

Each ROD board forms an event packet which consists of a variable length list of hit pixels or strips, along with time stamp and other associated data. This event packet is sent from the ROD board over a high speed optical readout link (ROL) which conforms to the CERN SLINK specification[6]. SLINK transmitters are mezzanine cards located on the ROD transition boards.

1.3 Future Expansion

Plans are currently underway to install an "insertable B-layer" (IBL) pixel detector in 2013. The IBL consists of additional pixel modules arraigned in a barrel near the beam pipe, at a radius of approximately 34mm. A total of 224 planar modules or 448 3D modules will be mounted to 14 stave structures. As in the existing Pixel detector the ratio of modules to RODs is dependent on hit occupancy; modules closest to the interaction region will have more hits and thus require more ROD resources and result in more ROLs.

2 The Fast Tracker

The ATLAS detector and readout electronics were not originally designed to trigger on tracks at the hardware level. As the LHC instantaneous luminosity increases from 1×10^{34} to 3×10^{34} $\mathrm{cm}^2\mathrm{s}^{-1}$ triggering on tracks will become necessary to reduce background events.

The Fast Tracker (FTK) system will find and fit tracks using the inner detector silicon layers for every event that passes the level-1 trigger. It receives the Pixel and SCT data at full speed from an duplicate output added to the ROD optical transmitter mezzanine cards[7]. The FTK system is a scalable, highly parallel processor which uses an associative memory approach to quickly find track candidates in coarse resolution roads[9]. Roads which match the selection criteria are then analyzed using full resolution silicon hits and the track parameters are reported to the level-2 trigger.

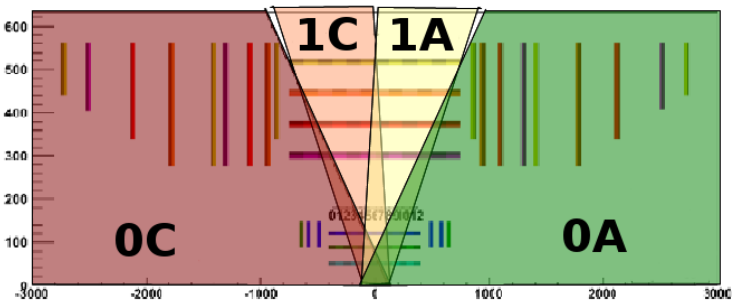

Figure 2: The Four FTK η regions. Note the significant overlap in the high occupancy central barrel regions.

2.1 FTK Towers

The FTK system partitions the ATLAS inner detector into four η regions as shown in Figure 2. The inner detectors are further partitioned into eight ϕ regions. The ϕ towers are split again into 16 regions of roughly 22.5° with approximately 10° overlap. Thus there are 64 $\eta - \phi$ towers. The Data Formatter system remaps the ATLAS inner detector modules to line up with the FTK $\eta - \phi$ towers. As we will describe in later sections, the module and ROD organization is less than ideal and will ultimately drive the overall design of the Data Formatter hardware.

3 Data Formatter Preliminary Design

The initial Data Formatter design process is described in this section. It is important to note that the Data Formatter hardware design is driven by input and output requirements, while also maintaining flexibility needed to accommodate future expansion and allowing for changes in the number of ROLs and module-ROD assignments.

3.1 Inputs from RODs

The FTK system is "grafted" onto the existing ATLAS DAQ system. A new dual-output HOLA SLINK mezzanine card has been developed and approved for use on the ROD transition boards. This mezzanine card "taps" into the ROD output data stream. Ideally, the FTK system should have no impact on upstream ATLAS DAQ hardware. This means that the Data Formatter must have sufficient memory to buffer or process the entire input record without activating the flow control features of the data link. This also means that the Data Formatter must remain flexible with regard to input changes. New ROLs will be added to the system. The mapping between inner detector modules and ROLs may change over time. The Data Formatter must be able to compensate for these changes without imposing any limitations on the ATLAS front end electronics.

The Data Formatter receives 222 ROLs from the ATLAS RODs shown in Table 1. Chains of inner detector modules are processed by the RODs and following a L1 trigger a variable length list of pixel (or strip) hits are sent over the ROLs to the Data Formatter. Therefore to begin the Data Formatter design process it is necessary to understand the mapping between inner detector modules and the ROD boards. The module-ROD map has been extracted from the ATLAS CORACOOL database and is assumed to be complete and up to date.

Module names reflect their physical location on the detector. For example, module `L1_B15_S1_M1A` is located in Pixel barrel 1 and is located on the first half (S1) of bi-stave

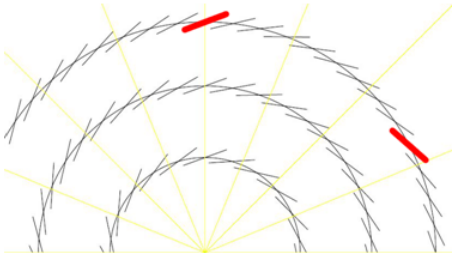

Figure 3: A Pixel Barrel ROL which contains modules with significant ϕ separation.

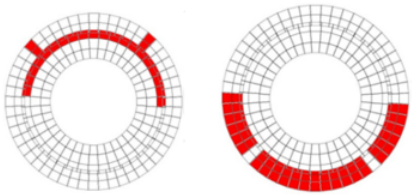

Figure 4: An SCT end-cap ROL spanning the entire ϕ range. Portions of this ROL will go to many Data Formatter boards.

number 15. In the Z direction it located in position M1 (closest to the center) on the "A" half of the barrel. (Refer to Figure 2 for more information about "A" and "C" halves of the detector.)

The FTK system partitions the inner detector into $\eta - \phi$ towers, so naturally the Data Formatter design would benefit from a similar organization of RODs and modules. In most cases modules are indeed organized in ϕ, which is beneficial since this will help to reduce data sharing between DF boards. However, some RODs modules have significant ϕ separation, as in Figure 3. In this example a Pixel barrel ROD `ROD_L2_S10` contains 13 modules: seven modules are from stave `L1_B10_S2` and six are from stave `L1_B15_S1`. This means that a single ROL will include modules from physically distant areas of the inner detector. Likewise, there are some ROLs that include end-cap modules from wide ϕ region, shown in Figure 4.

3.2 Outputs to FTK

The FTK core crates expect Pixel and SCT clusters organized in $\eta - \phi$ towers and delivered over fiber optic bundles, one bundle per $\eta - \phi$ tower. The details of the data transmission format for each fiber in this bundle are a work in progress. For the time being we will take a more abstract view, and insure that the DF system is capable of routing each module's data to the correct $\eta - \phi$ tower.

As previously mentioned, the FTK core crates define 64 $\eta - \phi$ towers, and each of these tower boundaries coincide with the module edges. We are still in the process of determining the exact $\eta - \phi$ tower boundaries used by the FTK system. In the meantime the DF simulation tool we have developed uses a conservative estimate, where modules that touch the tower boundary are included.

Reflex Photonic's "SNAP12" optical drivers have twelve parallel uni-directional fibers each rated for up to 3Gbps [10]. It is assumed that a return channel is needed to implement flow control between the DF and FTK core crates; individual fiber optic transceivers will be used for the return channel communications.

3.3 Data Formatter Partitioning

The FTK core crates expect the Data Formatter output arranged in 64 $\eta - \phi$ towers, one SNAP-12 fiber optic bundle per tower. If the Data Formatter hardware follows the same $\eta - \phi$ tower partitioning then the output fiber routing to FTK AUX cards is a simple one-to-one mapping. Any other Data Formatter organization would require external fiber optic splitter and re-combiner hardware. For this reason our design simulations will assume that the data Formatter hardware is partitioned into 64 "tower formatters" following the FTK organization.

In the innermost barrel layers the overlap between η regions is significant (refer to Figure 2). This overlap occurs in a high occupancy region of the inner detector, thus resulting in large data transfers between η regions within a ϕ tower. Data transfers across the crate backplane are minimized if adjacent η towers are located on the same Data Formatter board. Placing four $\eta - \phi$ towers on a single Data Formatter board is not feasible due to physical space and I/O constraints; however, two towers per board appears to be a good fit. In our simulation we will assume each Data Formatter board includes two towers, adjacent in η and from the same ϕ tower (e.g. towers 0A and 1A, 0C and 1C). Thus our baseline design consists of 32 Data Formatter boards, which we label `01A` to `16A` and `01C` to `16C`.

4 Data Formatter Simulation

At this point we have a clear understanding of relationship between RODs and inner detector modules. We also have some basic assumptions about the number Data Formatter boards and how they map into the FTK $\eta - \phi$ towers. Each of the 246 ROLs must plug into a single Data Formatter board, how this assignment is done will have a direct impact on the data sharing between boards. What ROL-DF map will minimize sharing between boards? How many connections are there between Data Formatter boards? How much data is transferred across each of these connections? Clearly, we need a new tool to simulate the system at a high level.

Early attempts to simulate the Data Formatter used a simple spreadsheet and assumed that the ROLs were arraigned in regular, symmetric regions in ϕ. When it was discovered that some ROLs were asymmetric and non-contiguous (see Figure 3) the spreadsheet effort was quickly abandoned. A new simulator tool is needed that can deal with module-ROD idiosyncrasies.

The first step is to explicitly define the ROL-module map and store the results in a database. ROL and Module names were extracted from the ATLAS CORACOOL database and is stored in a simple ascii text list, an excerpt of which is shown below:

```
ROD_NAME   MODULE_NAME
---------  -------------
ROD_D1_S6  D1A_B01_S1_M1
ROD_D1_S6  D1A_B01_S1_M2
ROD_D1_S6  D1A_B01_S1_M3
ROD_D1_S6  D1A_B01_S1_M4
ROD_D1_S6  D1A_B01_S1_M5
ROD_D1_S6  D1A_B01_S1_M6
```

The next step is to define the mapping between the Data Formatter towers and the modules. In most cases a module will belong to more than one tower. In this table wildcard characters (*) and comments (#) are supported. A excerpt of this file is shown below:

```
# pixel layer 0 DF A01
A01 L0_B01_S*_M0
A01 L0_B01_S*_M*A
A01 L0_B01_S*_M1C
A01 L0_B01_S*_M2C
```

In the above example the Pixel modules located on Layer 0, stave 1, bi-staves 1 and 2, "A" side and part of the "C" side map to Data Formatter board `A01`.

4.1 Assigning ROLs to Data Formatter Boards

Our high level Data Formatter simulator tool is written in Python and is called ROLMAP. After reading in the ROL-module and DF-module data files, ROLMAP performs a *join* on the module names. The resulting table represents the *intersection* between the ROLs and Data Formatter boards:

```
                    A  A  A  A  A  A  A  A  A  A  A  A  A  A  A  A
                    0  0  0  0  0  0  0  0  0  1  1  1  1  1  1  1
                    1  2  3  4  5  6  7  8  9  0  1  2  3  4  5  6

ROD_D1_S6   12      .  .  .  .  .  .  .  .  .  .  .  .  .  4  10 6
ROD_D2_S10  12      .  .  .  .  .  .  .  .  .  .  .  2  5  5  5  3
ROD_D1_S21  12      10 6  .  .  .  .  .  .  .  .  .  .  .  .  .  4
ROD_D1_S19  12      .  4  10 6  .  .  .  .  .  .  .  .  .  .  .  .
```

In the above output fragment we see four ROLs and how they intersect with Data Formatter boards `A01` through `A16` (the 16 "C boards" are not shown here due to space limitations). Let's consider the first ROL, `ROD_D1_S6`, which contains 12 modules. Four of the 12 modules are needed by board `A14`; 10 of the 12 modules are needed by board `A15`; and six of the 12 modules are needed by board `A16`. Since "most" of this ROL is needed by board `A15`, it makes sense to assign it to board `A15`. Thus board `A14` must import 4 modules from board `A15`. Likewise, board `A16` must import six modules from board `A15`.

4.2 Balancing and Optimization

Data sharing between boards is minimized when the ROL is plugged into the board which "needs it the most". Initially this is how ROLMAP assigns ROLs to boards. This simple algorithm does not however take into account balancing the number of ROLs per board. After the initial assignments some Data Formatter boards have 4 ROLs while others have 15. Clearly more optimization is needed. ROLMAP determines the number of ROLs per Data Formatter board and calculates the standard deviation on this list. ROLMAP then makes successive passes over the data structures, choosing the "next best" board to assign each ROL. After a few passes the minimum standard deviation value is found. The result is that each Data Formatter board has 6 or 7 input ROLs.

4.3 Data Sharing Between Boards

After the ROLMAP program optimizes the ROL assignments it calculates the number of modules that must be transferred between Data Formatter boards. The result of this calculation is displayed as a square matrix, shown in Figure 5. Note that the main diagonal has been zeroed out (because boards do not need to share data with themselves). Rows describe the number of modules which must be exported by the specified board. For example, the first row describes board `A01`. This board exports 31 modules to board `A02`, 3 modules to `A02`, and so on. Likewise, columns in this matrix describe the number of modules which must be imported by the specified board.

Matrix elements have been colored to help visualize the number of modules transferred between boards. The highest module counts are along the main diagonal, which is good news because it indicates that Data Formatter boards (which mirror the FTK $\eta - \phi$ towers) share data with neighboring boards in ϕ. A pair of smaller diagonals, offset from the main diagonal, shows the data sharing that occurs in η: boards need to transfer barrel modules cross the half-barrel boundary (at $z = 0$, see Figure 2).

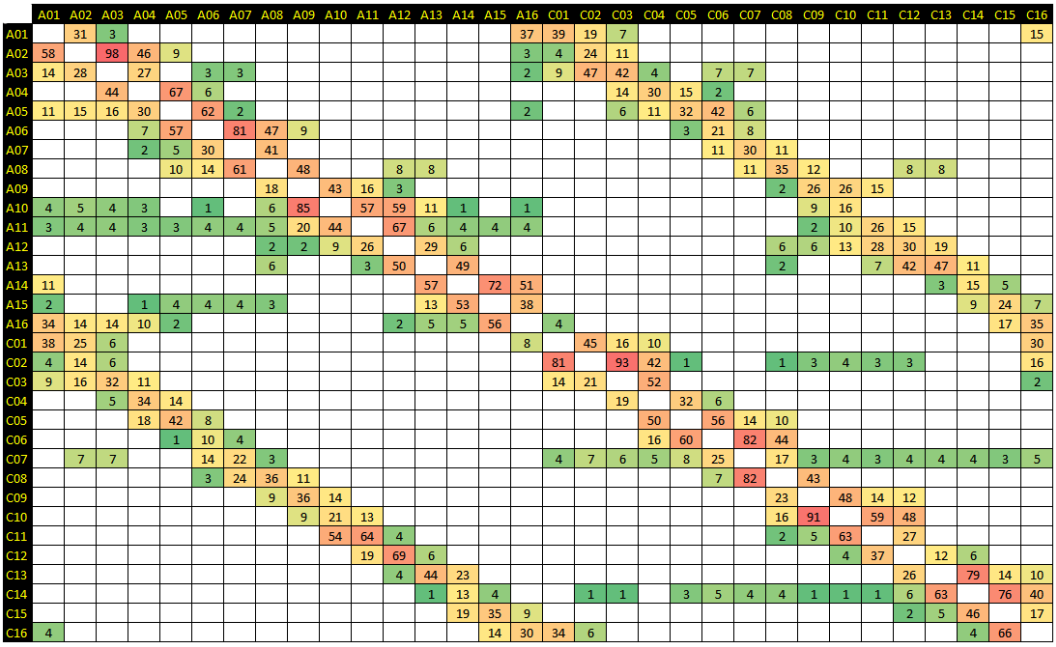

Figure 5: Data Formatter board module sharing matrix.

Outside the main and two offset diagonals in the matrix we find some non-zero elements, which means that Data Formatter boards must share data with boards outside their neighbors in η and ϕ. This phenomenon is of great importance because a successful and robust hardware design must not simply accommodate the ideal conditions, but it must handle the exceptions as well. Closer inspection reveals that these "outlier" cells are caused by ROL-module mappings, as illustrated in Figure 3 and Figure 4. What we interpret graphically as a horizontal "spreading" in the sharing matrix corresponds to ROLs which come from RODs where the modules span a large ϕ range. For example, ROL `SCT_C6_R12` covers the entire ϕ range of an SCT end-cap, shown in Figure 4. Unfortunately the Data Formatter board associated with this ROL must export to all other boards on that side of the detector.

Ideally each Data Formatter board would share only with its two ϕ neighbors and η neighbor across the half barrel boundary. The ROLMAP results suggest that we come fairly close to this ideal, given some quirks module-ROD mapping. These quirks, however, do have a significant impact on the number of connections between Data Formatter boards. While these connections are well understood now, there are no guarantees that the ROD-module mapping is fixed, or that the number of ROLs will remain constant. The Data Formatter must be flexible enough to handle such changes to the system.

4.4 Crate Partitioning and Backplane Selection

The ROLMAP program results graphically illustrate the challenges faced when grouping boards into crates. At this stage our goals are to: 1) minimize the number of crates; 2) minimize the data transferred between crates; and 3) leave room in the crates for future expansion. Conventional electronics crates typically have fewer than 20 slots available, so that sets an upper bound. Two crates of 16 boards seems like a reasonable fit, but leaves only four slots available per crate for future expansion. Partitioning into four crates of eight boards leaves lots of extra slots available, so this configuration was selected as a starting off point.

The ROLMAP sharing matrix (Figure 5) was imported into a spreadsheet and groups of

	A01	A02	A03	A04	C01	C02	C03	C04	A05	A06	A07	A08	C05	C06	C07	C08	A09	A10	A11	A12	C09	C10	C11	C12	A13	A14	A15	A16	C13	C14	C15	C16
A01	0	31	3		39	19	7																					37				15
A02	58	0	98	46	4	24	11		9																			3				
A03	14	28	0	27	9	47	42	4		3	3			7	7													2				
A04			44	0			14	30	67	6			15	2																		
C01	38	25	6		0	45	16	10																				8				30
C02	4	14	6		81	0	93	42					1			1					3	4	3	3								16
C03	9	16	32	11	14	21	0	52																								2
C04			5	34			19	0	14				32	6																		
A05	11	15	16	30			6	11	0	62	2		32	42	6													2				
A06				7					57	0	81	47	3	21	8		9															
A07				2					5	30	0	41		11	30	11																
A08									10	14	61	0			11	35	48			8	12			8	8				8			
C05				18				50	42	8			0	56	14	10																
C06								16	1	10	4		60	0	82	44																
C07		7	7		4	7	6	5		14	22	3	8	25	0	17					3	4	3	4					4	4	3	5
C08										3	24	36		7	82	0	11				43											
A09												18				2	0	43	16	3	26	26	15									
A10	4	5	4	3						1		6					85	0	57	59	9	16			11	1		1				
A11	3	4	4	3					3	4	4	5					20	44	0	67	2	10	26	15	6	4	4	4				
A12												2				6	2	9	26	0	6	13	28	30	29	6			19			
C09												9				23	36	14			0	48	14	12								
C10																16	9	21	13		91	0	59	48								
C11																2		54	64	4	5	63	0	27								
C12																			19	69		4	37	0	6				12	6		
A13												6				2			3	50			7	42	0	49			47	11		
A14	11																								57	0	72	51	3	15	5	
A15	2			1					4	4	4	3													13	53	0	38		9	24	7
A16	34	14	14	10	4				2											2					5	5	56	0			17	35
C13																				4				26	44	23			0	79	14	10
C14						1	1						3	5	4	4					1	1	1	6	1	13	4		63	0	76	40
C15																								2		19	35	9	5	46	0	17
C16	4				34	6																					14	30		4	66	0

Figure 6: Partitioning the Data Formatter boards into four crates. Colored boxes represent crate boundaries.

eight boards were manipulated by hand to see if any patterns emerged. Since the boards naturally share data with their η and ϕ neighbors, groupings that follow this arrangement were tried first. The result is shown in Figure 6.

Minimizing data sharing between crates means maximizing sharing within each crate. We discovered that the ideal configuration closely follows the FTK tower partitions. Selecting groups of four boards adjacent in ϕ and their corresponding η neighbors concentrates most of the heavy module sharing within each crate backplane. Module sharing between crates appears relatively modest: crates adjacent in ϕ need to share the highest numbers of modules (100 to 200) while the non-adjacent crates share fewer (under 50) modules. Although more detailed analysis is needed, transferring on the order of 200 modules through several high-speed links appears manageable.

Now we focus on the data transfers within each crate. Here the connections between boards are quite dense. On average, each board connects with 6 other boards in the crate. There does not appear to be any repeating pattern within each crate, which strongly suggests that *every board needs the ability to communicate with every other board in the crate.* While an address/data bus would support board to board DMA data transfers, it is inherently blocking and will be unsuitable for high bandwidth applications such as this. Higher performance can only be archived through non-blocking direct *full mesh* point to point links between boards. What current backplane technologies support such an architecture?

- VME crates support a wide address-data bus which would not be usable for high speed board to board transfers. A very dense custom J3 backplane would need to be designed to implement the full mesh connectivity between boards. The custom J3 backplane would also need to provide a data path to the back of crate card.

- A full custom monolithic backplane could be fabricated using backplane traces or jumper cables to implement the board to board sharing. As with the VME J3 backplane, this is a significant design effort and an "off the shelf" solution is preferred.

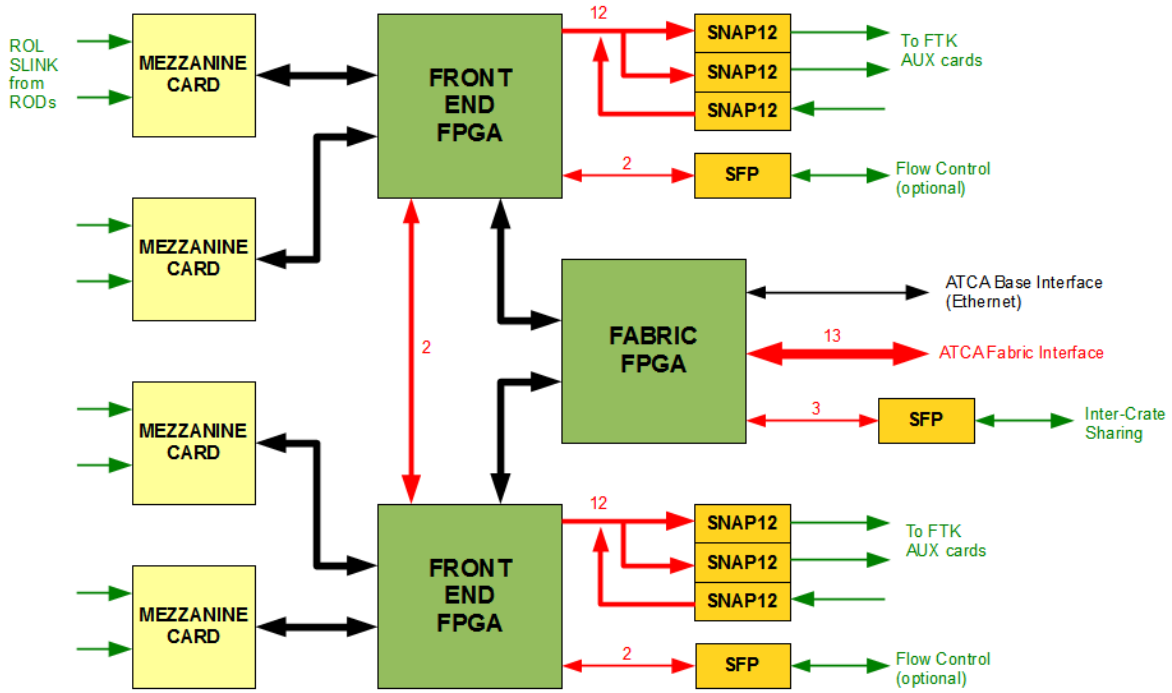

Figure 7: The Data Formatter board block diagram. High speed serial links are shown in red. Optical connections are shown in green. SNAP12 and SFP optical transceivers are located on the rear transition board (RTM).

- A "full mesh" backplane exists in the Advanced Telecommunications Computing Architecture (ATCA) standard. ATCA crates support up to 14 slots and feature dedicated high speed serial connections between all slots.

Early in our discussions it became clear the ATCA full mesh backplane is a quite a good fit for our design requirements. Other options would require significant engineering efforts to design a backplane that would essentially do what the ATCA solution already provides. A more in-depth discussion of the ATCA hardware is presented in Section A.

5 Data Formatter Board

Before introducing the Data Formatter board block diagram, let's summarize our assumptions and simulation results:

- There are 32 Data Formatter boards with two FTK $\eta - \phi$ towers per board. The two towers on the board are adjacent in η, located on the same side of the half barrel, and belong to the same ϕ tower.
- Eight ROL inputs per Data Formatter board. ROLMAP simulation indicates 6 to 7 ROLs per board are typical figures.
- Eight Data Formatter boards per crate. These eight boards are grouped in η and ϕ (A_n, ... A_{n+3}, C_n, ... C_{n+3}).
- Four crates, with several high speed links for inter-crate data sharing.

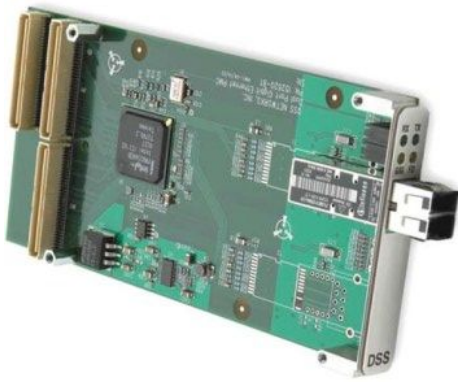

Figure 8: A Compact Mezzanine Card (CMC).

- Within a crate each Data Formatter needs to communicate with all other slots. The ATCA "Full Mesh" backplane as been selected.
- The ATCA crate has extra slots available for future expansion.

Minimizing latency through the Data Formatter latency is important as it directly effects the overall FTK system processing time. Furthermore, latency through the Data Formatter should be as consistent and deterministic as possible (despite the variable length lists of module hits provided by the RODs). For this reason, FPGAs are preferred for critical data paths within the Data Formatter system. Network switch components are less desirable because they employ elastic buffers and thus cannot guarantee consistent latency (nor can they guarantee in-order packet transmission).

Essentially the Data Formatter has three main tasks: 1) find clusters in the Pixel modules and receive hits from the SCT modules; 2) route these clusters (and hits) to their destination Data Formatter boards; and 3) concatenate and format the clusters and hits before transmission to the downstream FTK hardware. In the following sections will we show how these tasks will be performed by the Data Formatter hardware.

5.1 Cluster Finder Mezzanine Cards

The Data Formatter first stage unpacks ROD data and finds SCT hits and Pixel clusters. The cluster finder module is a mezzanine card which consists of fiber optic transceivers, FPGAs, and memory. The baseline design consists of four mezzanine cards per Data Formatter board. Each mezzanine card supports up to two ROLs which can be any combination of SCT or Pixel sub-detectors.

Processing SCT data is straightforward as the FPGA performs a simple one dimensional search for hits. Clustering pixel hits is a two dimensional problem and requires considerable logic to implement. The cluster finder algorithms and firmware are being developed at INFN[3]. On the prototype mezzanine card a Xilinx Spartan-6 FPGA (`XC6S150T`) is used to process one Pixel ROL and one SCT ROL.

Various mezzanine card specifications exist, however after considerable research we believe that the industry standard Common Mezzanine Card (CMC) format shown in Figure 8 is a good fit. A single-width CMC measures 74mm × 149mm and has up to 256 pins on four connectors. The CMC standard calls for up to 80 I/O pins on the JN1 and JN2 connectors, with an additional 100 I/O pins on the optional JN3/JN4 connectors. With so many general purpose I/O pins available it is possible to implement wide single-ended buses to and from the CMC card. When two SLINKs are plugged into the CMC card the total input bandwidth

is 2 × 32 bits × 40MHz = 2560Gbps. CMC output buses can easily transfer this much data with reasonable clock rates and without resorting to dual-edge clocking (DDR) techniques.

5.2 Data Formatter Main Logic

As shown in Figure 7 the Data Formatter design uses three FPGAs to implement the core functionality. The three FPGAs used on the Data Formatter are low cost ($300) Artix 7 devices.

Combining three FPGAs into a larger, single FPGA was also explored. In a single FPGA, the Gigabit serial transceiver count alone would require a large, relatively expensive ($2000) Virtex-7 FPGA. The wide parallel data buses from the four mezzanine cards pushed the upper limits on the I/O pin count, even in largest BGA packages. Consequently it was discovered that a FPGA "sweet spot" exists for this design: when three FPGAs are used a good balance between the transceiver count (16), I/O pin count (600), logic cells (350k) is archived.

In this section we will describe the functions of the three FPGAs since the logic partitioning is a good fit to Data Formatter operation.

5.2.1 Front End FPGA Operation

Two Front End FPGAs accept Pixel clusters and SCT hits from the CMC cards. For each cluster the Front End FPGA must do the following:

- Determine if the cluster is needed for the home tower. If so, save it in memory for later transmission downstream to FTK.
- Determine if the cluster is needed by another tower. If so, send it to the destination board over the full mesh fabric or dedicated fiber optic link.

Front End FPGAs also accept clusters from the Fabric FPGA. These clusters are stored in the Front End FPGAs (or external RAM chips) for later transmission to FTK.

The Front End FPGAs are connected by a pair of dedicated high speed serial links. Each link is full duplex and runs at speeds up to 5Gbps, which is more than sufficient to support the worst case situation where the Front End FPGAs must exchange all of their cluster data.

Once the Front End FPGAs have collected the full list of clusters and hits these are sorted and formatted into packets and transmitted over the appropriate SNAP-12 channels to the FTK AUX cards. It is assumed that the FTK system will utilize flow control and fiber optic transceivers are included for this purpose.

5.2.2 Fabric FPGA Operation

The Fabric FPGA is the gateway by which the Front End FPGAs send and receive clusters with other Front End FPGAs in the system. Both Front End FPGAs send clusters to the Fabric FPGA. For each of these clusters the Fabric FPGA determines the following:

- Determine if the cluster needs to go to another DF in the same crate. If so, send it to the appropriate DF board over the full mesh fabric.
- Determine if the cluster needs to go to a DF in another crate. If so, send it over the dedicated fiber link to the proper crate.

The Fabric FPGA also receives clusters and hits from the full mesh fabric, or directly from other crates via dedicated fiber links. Each of these clusters is sent to one or both Front End

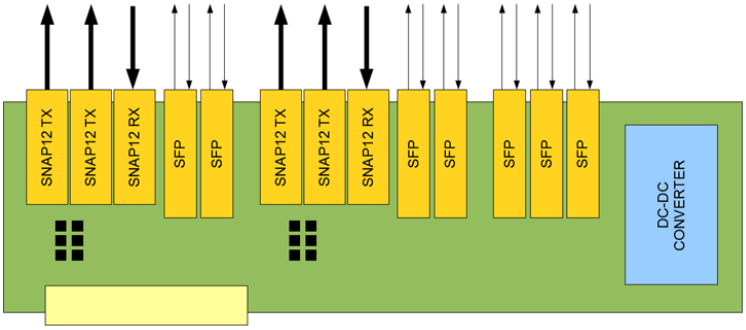

Figure 9: Data Formatter Rear Transition Module (RTM). The RTM measures 8U X 90mm.

FPGAs. Fabric FPGAs also have the ability to re-route incoming clusters and hits to other boards.

Ethernet connectivity to the backplane occurs through a PHY chip attached to the Fabric FPGA. This interface is intended to be used for board control and monitoring, as well as diagnostics and FPGA programming. The ATCA management interface (IMPI, described in Section A.3) is also available for slower speed control and monitoring.

5.3 Rear Transition Module

The Rear Transition Module (RTM) is shown in Figure 9. The upper four Small form-factor pluggable (SFP) optical transceivers are used for FTK flow control and the lower three SFP units are used for inter-crate data sharing. SNAP12 Receiver units are not required for Data Formatter operation but may be used for loopback testing during commissioning and board check-out.

The Zone-3 connector contains 62 signal pairs. Each Front End FPGA drives a duplicate SNAP-12 transmitters, which is a requirement of the FTK system. High speed PECL splitter buffer/driver chips are used to cleanly drive both SNAP-12 transmitters. These splitter chips are available from multiple vendors and are rated for up to 3Gbps.

The RTM contains no other active circuitry and is powered through the Zone-3 connector. It is anticipated that +3.3V power is all that will be required on the RTM.

6 Firmware Simulation and Diagnostics

The Data Formatter boards consist of three large FPGAs each with a well-defined set of tasks. Since there are many FPGAs in the Data Formatter system care should be taken to avoid unique hard-coded parameters in the source code. Moving board parameters into configuration registers reduces the number of unique firmware builds. In some cases the boards can determine operating parameters automatically with no user input. Automatic calculation of routing table data is one example of this technicque and it is described in more detail in Appendix C.

6.1 Simulation Techniques

FPGAs are programmed in VHDL and simulated separately and then together as a “test bench” in a dedicated HDL simulator tool such as ModelSim or ActiveHDL. Large firmware designs benefit greatly from a carefully designed and through test bench simulation because the simulator tool offers the best visibility into internal registers, buses and memory elements.

While tools (such as "ChipScope") exist to probe internal nodes on the physical device, this "burn and learn" technique is quite limited, tedious, and often requires running the design back through the "place and route" tools, thereby changing the device timing.

The VHDL testbench is most successful when paired with a corresponding physics simulation, often written in C++ or some other high level language. If the testbench and physics simulator agree on the input and output data format then they can operate in parallel and perform cross checks on large numbers of events. This is an excellent way to insure that the Engineer's firmware design behaves exactly the same way the Physicist believes it should operate.

6.2 On Board Diagnostics

Simulation and verification does not have to end after installation and commissioning is completed. Buffers on the input and output of each board should be able to capture the input and output records. These buffers can then be read out and run through the testbench and physics simulation tool at any time. Test vectors can also be loaded into the buffers and injected into the design under test, or sent to downstream hardware as well.

7 Current Status

Currently we are working on finalizing the details on the mezzanine card connectors, investigating the IMPC interface, and simulating various firmware modules dealing with data routing.

Appendix A AdvancedTCA Hardware

Figure 10: An ATCA board and 14-slot shelf unit.

Virtually every component in the ATCA shelf is a "field replaceable unit" (FRU) which means it may be replaced without powering down the shelf. Boards, fans, power entry modules, and shelf manager boards are *hot-swappable* and redundant. From the ground up ATCA has been designed for high availability operation.

A.1 Shelf

Boards are inserted into the ATCA shelf slots. Large shelf units contain 14 slots in a vertical configuration; smaller shelf units generally orient the blades horizontally. A typical 14 slot ATCA shelf is shown in Figure 10. For our application a 14 slot shelf will be used.

Each board is 8U (322.25mm) by 280mm deep. The width of each slot is considerably wider than VME at 30mm, which allows for taller components such as connectors, mezzanine cards, and power converters.

A.2 Backplane

The PICMC specification specifies three backplane connector zones. Zone-1 is near the bottom of the board and this connector is used for redundant 48VDC power and Intelligent Platform Management Controller (IPMC) management signals.

High speed data communication between boards occurs on the Zone-2 connectors. A few clocks and other synchronization signals are bussed to all slots in the shelf, however a vast majority of the Zone-2 connections are point to point high speed serial links. ATCA is often described as "protocol agnostic" which means that the PICMC specification simply describes the physical and electrical characteristics of these connections. The high speed serial data protocol is user defined. Zone-2 is comprised of two type of connections: the *Base Interface* and *Fabric Interface*.

The *Base Interface* is wired as dual star topology. There are two redundant hub slots in the center of shelf; these hub slots are referred to as logical slots 1 and 2. Each hub slot has a direct connection to every other slot in the shelf. The Base Interface protocol is TCP/IP over Gigabit Ethernet (1000BASE-T) and is intended for out of band management operations such as board control and monitoring.

Figure 11: A Shelf Manager board.

High speed data transfers take place on the Zone-2 *Fabric Interface*. The Fabric Interface is available in full-mesh, dual-star, dual-dual-star and replicated-mesh topologies. The Data Formatter will use the full-mesh configuration, which features four 100Ω differential signal pairs between each slot. Each differential signal pair is rated for speeds up to 10Gbps.

The final backplane region is the user-defined Zone-3 at top of the board. Connectors in the Zone-3 area are intended for passing data from the front board to the rear transition module (RTM). There is no backplane in this zone; rather the front board connectors mate directly with the connectors on the RTM.

A.3 Intelligent Platform Management Interface

ATCA hardware incorporates an Intelligent Platform Management Interface (IPMI), which is required on all shelf components. Through this interface the Shelf Manager card can query sensors and control shelf components. For example, if the Shelf Manager detects an over-temperature condition on a board then the fan speed may be increased or the board could be powered down.

High availability operation is archived through redundancy built into the IMPI specification. Each shelf has dual redundant Shelf Manager cards, one is shown in Figure 11. If the master Shelf Manager fails then control automatically transferred to the slave unit. Like other ATCA components the Shelf Manager boards support hot swap operation. The heart of the Shelf Manager card is a single board computer running Linux. It is possible to log into the Shelf Manager through telnet, SSH, or a serial terminal; however the user will typically interact with Shelf Manager through the web interface. An Ethernet port is located on the front panel of the Shelf Manager, and there is also an option to connect the Shelf Manager to the backplane *Base Fabric* network as well.

Shelf Manager cards communicate over the dual redundant Intelligent Platform Management Bus (IPMB), which uses the I^2C protocol [13] for the base layer. Typically the following sensors are monitored: temperature, fan speed, voltage and current, and board handle switch status. The board or FRU must also report back a description, serial number, manufacturer name, part number, and various hardware, firmware, and software version numbers. The IPMI protocols are fairly complex and a microcontroller (Intelligent Platform Management Controller, or IPMC) must be used.

Hot swap operation is implemented by monitoring the status of a microswitch in the board handle and controlling the DC-DC converters on the board. Removing a component from an ATCA shelf requires following a simple procedure which involves opening the ejector handle slightly, then watching the blue “HS” LED until it indicates that the board has completed the shutdown procedure, then the board may be removed safely from the system.

In system device programming is supported by IPMI commands. Firmware devices such

as FPGA configuration PROMs, Flash memories and microcontrollers may be programmed and verified through the IPMC. The IPMC microcontroller firmware itself may be updated remotely as well.

IPMC reference designs are available commercially available [11]. The reference designs fully implement the latest IPMI specification and have been debugged and technical support is provided. However the commercial reference designs are strictly licensed, closed source, and discourage collaboration by requiring non-disclosure agreements. As an alternative, several physics laboratories have produced open-source designs [12] for IPMC controllers. We are currently investigating which route to take.

A.4 Network Connectivity

As previously mentioned, the Ethernet *Base Interface* is available for high speed board management communication, such as downloading firmware and reading and writing board parameters. Since the *Base Interface* is based on Ethernet, each board must incorporate a PHY chip and support one of the following protocols: 10BASE-T, 100BASE-T, or 1000BASE-T. Low-cost and low-power PHY chips which support all three speeds are readily available. The PHY signals are routed to an FPGA which implements a MAC interface and TCP/IP stack and allows the board to communicate on the *Base Interface* private network. Network protocols above the TCP/IP layer is user defined; boards can support WWW, telnet, ssh, etc. Soft-core processors and Ethernet MAC cores are available through the Xilinx COREgen tool.

It should be noted that if the Data Formatter board uses the *Base Interface* an additional "hub board" must be added to the crate to implement the central Ethernet switch. Hub boards may simply be a switch, or it may be coupled with a high speed general purpose processor that is capable of running user code.

Alternatively, the IPMI management interface may be used for slow monitoring, control, and diagnostics. However the IMPI interface is considerably slower than the Ethernet interface.

A.5 Power Supply

ATCA evolved out of the telecommunications industry, which has historically used a -48VDC power distribution. The shelf incorporates dual redundant power entry modules, each of which has a connection for the -48V supply and return lines. ATCA hardware supports up to 200W per slot.

Power supplies are also redundant. A common configuration is a 1U rackmount chassis with three power supplies which operate in an "N+1" redundant mode. Output diodes and special circuitry is employed to implement dynamic load sharing and hot swap capability. Therefore, a failed supply can be shutdown or replaced without interrupting crate operation.

Our experience with 48VDC "N+1" redundant power supplies has been extremely positive. For instance, when a power supply fails it is shut down and the other supplies automatically take up the load without interruption. Then, during a normally scheduled controlled access the faulty supply is simply replaced. Local voltage regulation on the board (with isolated DC-DC converters) is reliable and eliminates the need for remote sensing which is common on low-voltage high-current power supplies. Compared to a large low-voltage high-current power supply a board mounted DC-DC converter can simply react faster to the highly dynamic load often associated with high performance FPGAs, resulting in improved voltage regulation.

Appendix B Data Formatter Board Picture

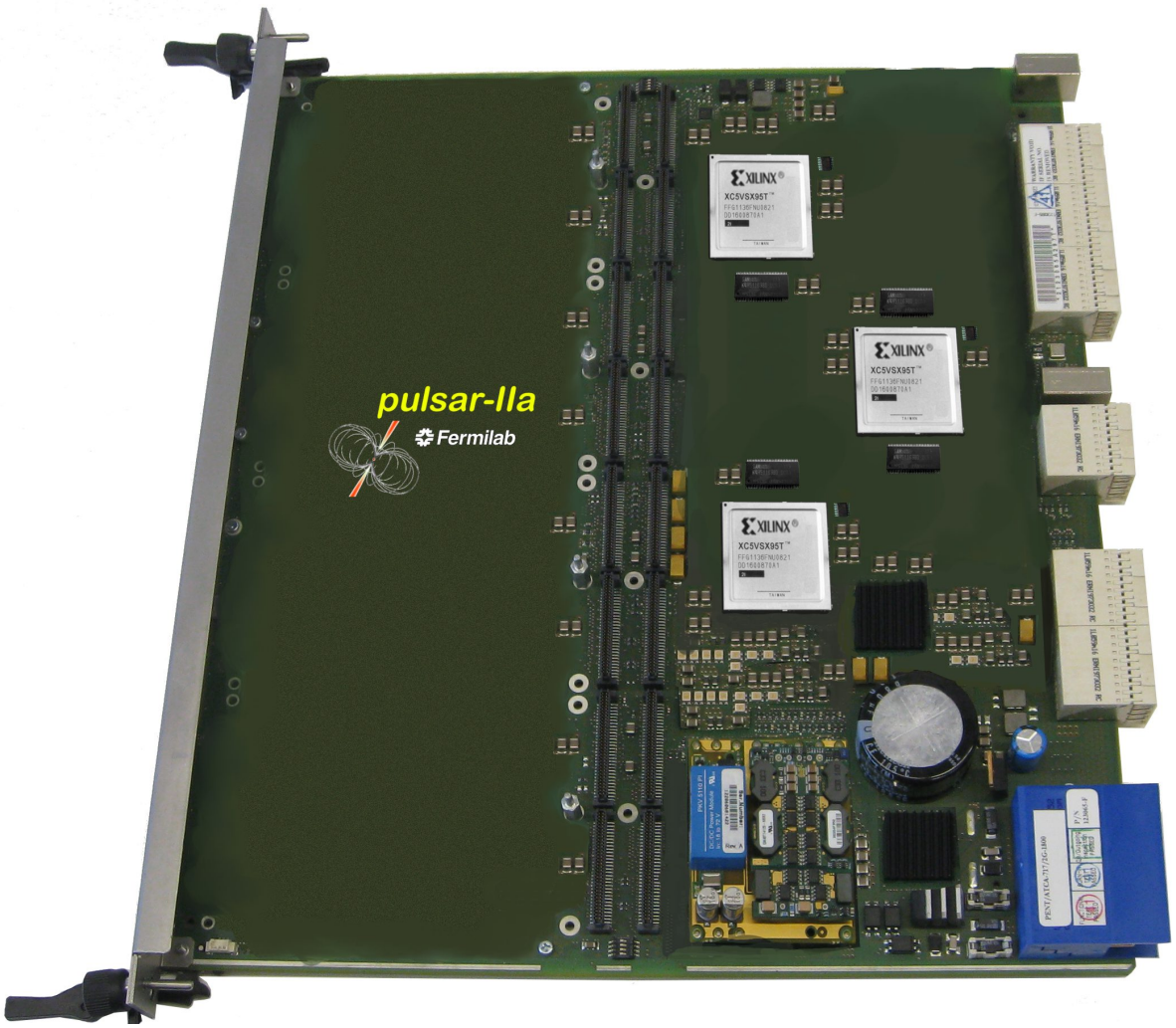

Figure 12: Artistic rendering of the DF board.

Appendix C Dynamic Routing Algorithm

By using a simple "discovery protocol" messaging scheme the Data Formatter boards can automatically determine the optimal routing paths. The algorithm described here has been simulated and it has been determined that after a few iterations every board properly calculates its routing table.

Figure 13 shows four crates, each with 8 Data Formatter boards. (Note that the empty slots are not shown for clarity. All 14 slots in the crate are connected in the full mesh.) Each line represents a full duplex communication link.

All boards in the system function as a router nodes, receiving data packets and forwarding packets on to their destination. Note that the boards do not know the entire path to the destination; rather, boards use their routing tables to determine which output to use. For example, if board 8 has data that must get to board 23 it sends the data to board 14; board 14 sends to board 18; board 18 sends to board 23 for a total of 3 hops.

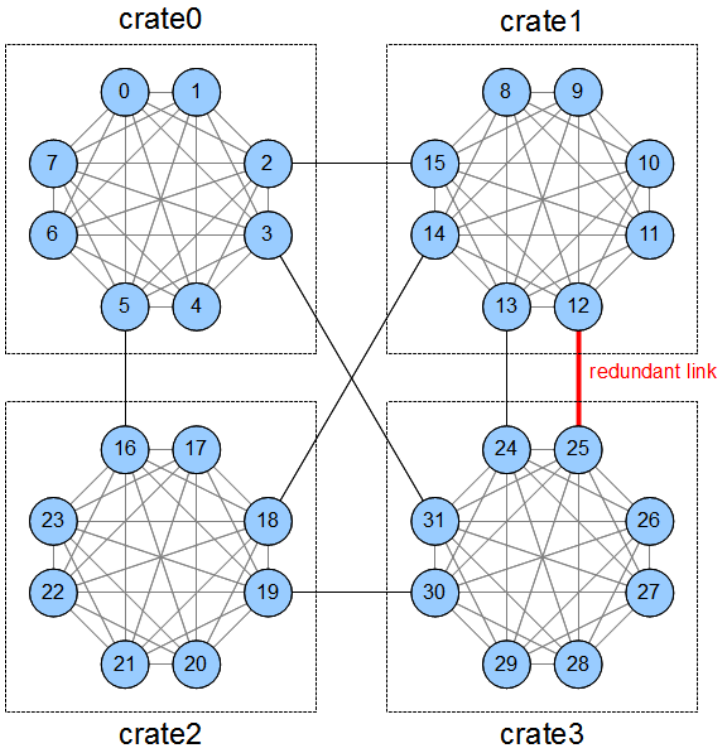

Figure 13: An example configuration of Data Formatter boards.

Here's how the algorithm works. Each board maintains a routing table, which consists of 32 entries, one entry for every board. The entry fields are DF id, valid_flag, hops, and outport. Initially, the routing tables are empty: each board knows only its unique ID number and marks the home entry in its routing table as valid and zero hops.

All boards loop through their routing tables and for each valid entry in the table they transmit a routing message ("I am X hops away from board Y") on all output ports. Boards listen on their input ports for messages coming from other boards. When a new message arrives the board notes the input port it arrived on, increments the number of hops and compares the message against the appropriate entry in it's routing table. If the number of hops in the message is less than the table entry then the table is updated with the new number of hops and port information. After a few iterations all routing tables are filled. An example routing table for board 8 is shown in Table 2.

DFid	V	Hops	Port	DFid	V	Hops	Port
0	1	3	6	16	1	3	5
1	1	3	6	17	1	3	5
2	1	2	6	18	1	2	5
3	1	3	6	19	1	3	5
4	1	3	6	20	1	3	5
5	1	3	6	21	1	3	5
6	1	3	6	22	1	3	5
7	1	3	6	23	1	3	5
8	1	0	X	24	1	2	4
9	1	1	0	25	1	2	3
10	1	1	1	26	1	3	4
11	1	1	2	27	1	3	4
12	1	1	3	28	1	3	4
13	1	1	4	29	1	3	3
14	1	1	5	30	1	3	3
15	1	1	6	31	1	3	3

Table 2: Example routing table for board number 8.

Redundant links, shown in red in Figure 13, provide an alternate, equivalent data path between crates, thereby reducing or eliminating data flow bottlenecks. The routing algorithm as described above must be modified slightly so that it properly balances traffic across redundant links. Link balancing is accomplished when new routing messages arrive. For example, board 8 receives messages from boards 12 and 13, on ports 3 and 4, respectively. Because boards 12 and 13 have a direct connections to crate3, they will inform board 8 of their connections to boards 24 through 31. The routing algorithm as described in the preceeding paragraph will work, however our simulation shows that it ignores the redundant link and channels all traffic between crate1 and crate3 through the link connecting boards 13 and 24. The solution to link balancing turns out to be straightforward. Whenever a new routing message has the same number of hops as the corresponding table entry, then count the number of times the new and old port numbers appears in the table. If the new port number occurs less frequently than old port, then update the table with the new port number. The result can be seen in the routing table for board number 8 (Table 2): the path to boards 24 through 31 is evenly split on ports 3 (board 12) and port 4 (board 13).

References

[1] ATLAS Experiment at CERN
Geneva, Switzerland
http://atlas.ch

[2] Fermi National Accelerator Laboratory
Batavia, Illinois 60510 USA
http://www.fnal.gov

[3] A Fast General-Purpose Clustering Algorithm
Based on FPGAs for High-Throughput Data Processing
A. Annoiv and M. Beretta
INFN - Laboratori Nazionali di Frascati, via E. Fermi 40, Frascati

[4] PICMC AdvancedTCA Core Short Form Specification
http://www.picmg.org/pdf/PICMG_3_0_Shortform.pdf

[5] PICMC Advanced Mezzanine Card Short Form Specification
www.picmg.org/pdf/AMC.0_R2.0_Short_Form.pdf

[6] CERN SLINK Homepage
http://hsi.web.cern.ch/hsi/s-link/

[7] Dual output SLINK card

[8] Fast Tracker Technical Proposal

[9] Associative Memory

[10] Reflex Photonics http://reflexphotonics.com/products_2.html

[11] Pigeon Point Systems, Inc. http://www.pigeonpoint.com/products.html

[12] CERN xTCA Resources Wiki https://twiki.cern.ch/twiki/bin/view/XTCA/WebHome

[13] Introduction to the I2C Bus http://www.i2c-bus.org/