June 22, 2012

FTK: a hardware track finder for the ATLAS trigger

Technical Proposal

A. Annovi[1], M. Beretta[1], M. Bogdan[2], E. Bossini[3a, 3b], A. Boveia[2], F. Canelli[2,4], F. Crescioli[3a, 3b], M. Dell'Orso[3a, 3b], S. Donati[3a, 3b], G. Drake[5], M. Dunford[2], J.F. Genat[2], P. Giannetti[3a], F. Giorgi[6a, 6b], J. Hoff[4], A. Kapliy[2], M. Kasten[7], Y.K. Kim[2,4], N. Kimura[8], T. Liu[4], A. McCarn[7], C. Melachrinos[2], A. Negri[9a, 9b], M.S. Neubauer[7], B. Penning[4], M. Piendibene[3a, 3b], J. Proudfoot[5], C. Roda[3a, 3b], M. Shochet[2], F. Tang[2], J. Tuggle[2], V. Vercesi[9a], M. Verzocchi[4], M. Villa[6a, 6b], G. Volpi[1], J.Y. Wu[4], K. Yorita[8], J. Zhang[5], A. Zoccoli[6a, 6b]

[1] *INFN of Frascati*
[2] *University of Chicago*
[3] *INFN of Pisa(a), University of Pisa(b)*
[4] *Fermilab*
[5] *Argonne National Laboratory*
[6] *INFN of Bologna(a), University of Bologna(b)*
[7] *University of Illinois*
[8] *Waseda University*
[9] *INFN of Pavia(a), University of Pavia(b)*

1 Introduction

Triggering is an enormous challenge in a hadron collider experiment. With the nominal LHC operating parameters, there are 40M bunch crossings per second. Since only about 200 events per second can be stored for later analysis, the trigger must select on average the most important event per 10^5 bunch crossings. That task is complicated by the ~ 25 overlapping *pp* collisions per crossing at the design luminosity of 1×10^{34}. The existing ATLAS trigger system was designed to work well under these conditions.

The first stage of the SLHC accelerator upgrade is now being planned. The expected instantaneous luminosity will increase to 3×10^{34}, meaning the average number of collisions per crossing will rise to 75. This makes the trigger task even more difficult since signal rates increase linearly with luminosity and backgrounds go up even more quickly. The load on the level-2 trigger system will significantly increase due to both the need for more sophisticated algorithms to suppress background and the larger event sizes. It is not obvious how best to deal with this challenge since the new physics processes that will be seen at the LHC are unknown. Until they are discovered, we don't know whether the important triggers will involve leptons or jets, moderate or large transverse momentum. Some triggers can take much greater than the allowed average level-2 event execution time as long as that class represents a small fraction of the total level-1 trigger rate. But we don't know *a priori* what the optimal level-1 trigger mix will be. The needed trigger menu flexibility can be enhanced by increasing the available information. Early track reconstruction can be an important element of the SLHC trigger strategy.

There are numerous examples of the importance of tracking in the trigger. The source of electroweak symmetry breaking couples in proportion to mass. Thus heavy fermions are likely in the final state, in particular *b* quarks and τ leptons. High trigger efficiency for these processes requires sensitivity to the generic hadronic decays of the heavy fermions. The challenge comes from the enormous background from QCD produced light quark and gluon jets, which can be suppressed using tracking. Tracks coming from a secondary vertex or not pointing to the beamline identify *b* quark jets, while τ jets can be separated from background using the number of tracks within a narrow cone and the number in a larger isolation region.

Electron and muon triggers can also be improved at high luminosity using track information. Traditionally background is suppressed by applying an isolation requirement using the calorimeters. At SLHC luminosity, the energy added by the 75 additional collisions results in either decreased lepton efficiency or increased background contamination. The effect can be greatly ameliorated with a track-based isolation only using tracks pointing to the lepton candidate at the beamline.

Global rather than Region-of-Interest tracking can further expand triggering possibilities. Among numerous examples are a lepton plus isolated track trigger for extending new physics searches and improved rapid primary vertex determination.

The Fast TracKer (FTK) will enable early rejection of background events and thus leave more level-2 execution time for sophisticated algorithms by moving track reconstruction into a hardware system with massively parallel processing that produces global track reconstruction with near offline resolution shortly after the start of level-2 processing. FTK is based on the very successful CDF Silicon Vertex Trigger (SVT).

The system is described in section 2. FTK performance at 3×10^{34} luminosity for both individual tracks and physics objects is presented in section 3. The path to even higher luminosities is discussed in section 4. Needed personnel resources as well as cost and schedule are described in section 5, with staging possibilities presented in section 6. The summary and conclusions are in section 7.

2 System Description

2.1 Functional overview

2.1.1 Introduction

FTK is an electronics system that rapidly finds and fits tracks in the inner detector silicon layers for every event that passes the level-1 trigger. It uses all 11 silicon layers over the full rapidity range covered by the barrel and the disks. It receives the pixel and SCT data at full speed as it moves from the RODs to the ROSs following a level-1 trigger, and after processing it fills a ROBIN with the helix parameters of all tracks with P_T above a minimum value, typically 1 GeV/c. The level-2 processors can request the track information in a Region of Interest or in the entire detector. A dual-output HOLA board provides FTK with an identical copy of the silicon data being sent to the DAQ ROBINs. That board was designed and tested, passed an ATLAS electronic board review, and has since been used in the CDF level-2 trigger. Dual-output HOLAs would replace the existing HOLAs in the pixel and SCT RODs.

FTK is a scalable processor. The system presented below will work well at the SLHC Phase I luminosity of 3×10^{34} $cm^{-2}s^{-1}$. We describe how it could be expanded to operate at the higher Phase II luminosities. We also present staging options that would work at luminosities up to 1×10^{34} $cm^{-2}s^{-1}$ and serve as an R&D prototype for the full Phase I system.

We have modified the architecture of the FTK since the ATLAS upgrade R&D proposal was approved. At that time, data flowed serially through the boards within each FTK crate. All of the silicon hits passed through all of the pattern recognition boards, and track candidates passed through a single track fitter board. That scheme worked well at the luminosities for which ATLAS digitization worked at the time. We now have simulation at 3×10^{34} $cm^{-2}s^{-1}$ where we

find the data flow problem to be much more challenging. Figure 2.1 shows the number of hits in each silicon layer as a function of luminosity. Note that the *y* intercept, which is barely visible above the origin, represents all of the hits from the hard scattering process!

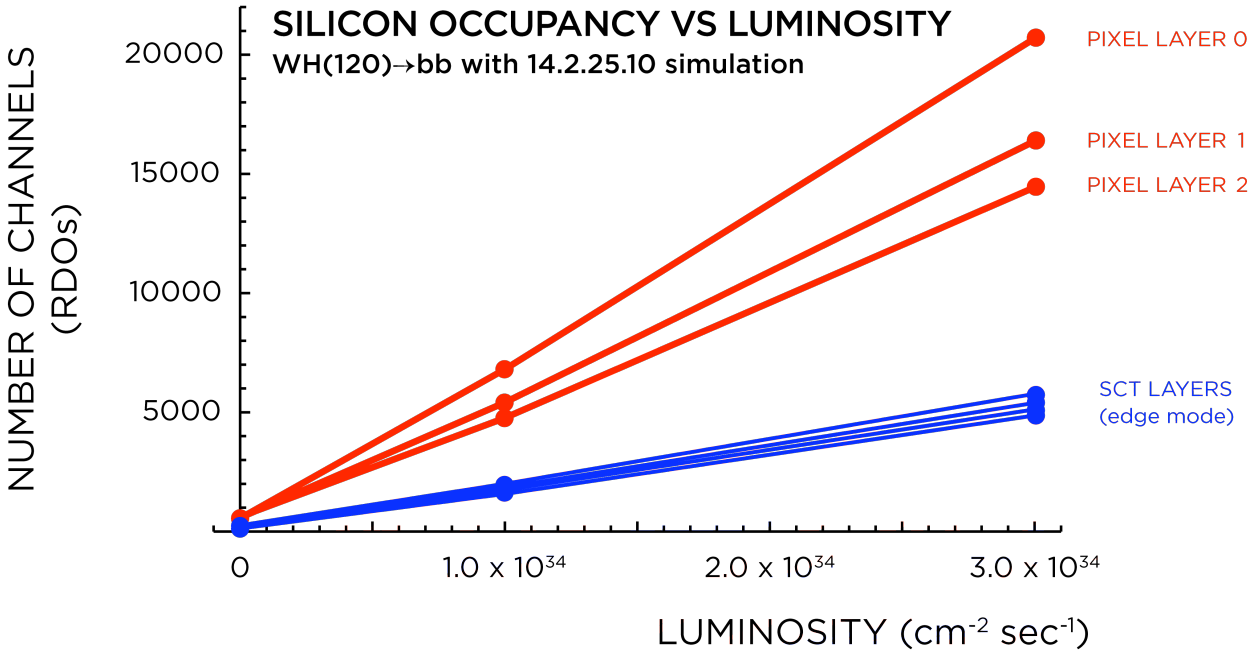

Figure 2.1: The number of ROD output hits from each silicon layer in the barrel vs. luminosity.

To deal with the large input data rate at 3×10^{34} $cm^{-2}s^{-1}$ as well as the greatly increased number of track candidates found at that luminosity, we have made the FTK much more highly parallel, segmenting a crate into η-ϕ towers, each with its own pattern recognition hardware and track fitter. A tower receives many fewer silicon hits, and the track fitter has many fewer candidates to process.

2.1.2 The FTK architecture

The FTK algorithm consists of two sequential steps of increasing resolution. In step 1, pattern recognition is carried out by a dedicated device called the Associative Memory (AM) [1] which finds track candidates in coarse resolution roads. When a road has silicon hits on all layers or all except one, step 2 is carried out in which the full resolution hits within the road are fit to determine the track helix parameters and a goodness of fit. Tracks that pass a χ^2 cut are kept. If there are hits in all layers and the χ^2 fails the cut but is not extremely large, the track is refit a number of times with the hit on one of the layers dropped each time. This "majority recovery" allows for the loss of a single hit due to detector inefficiency with a random hit picked up instead.

The first step rapidly carries out what is usually the most CPU intensive aspect of tracking by massive parallelism – processing hundreds of millions of roads nearly simultaneously as the silicon data pass through FTK. The road width must be optimized. If it is too narrow, the needed size of the AM and hence the cost is too large. If roads are too wide, the load on the track fitters can become excessive due to the large number of uncorrelated hits within a road. This increases both the number of roads the track fitter must process and the number of fits within the road due to the hit combinatorics.

Figure 2.2 shows a sketch of FTK, which is FPGA based with the exception of one specially designed chip for the associative memory.

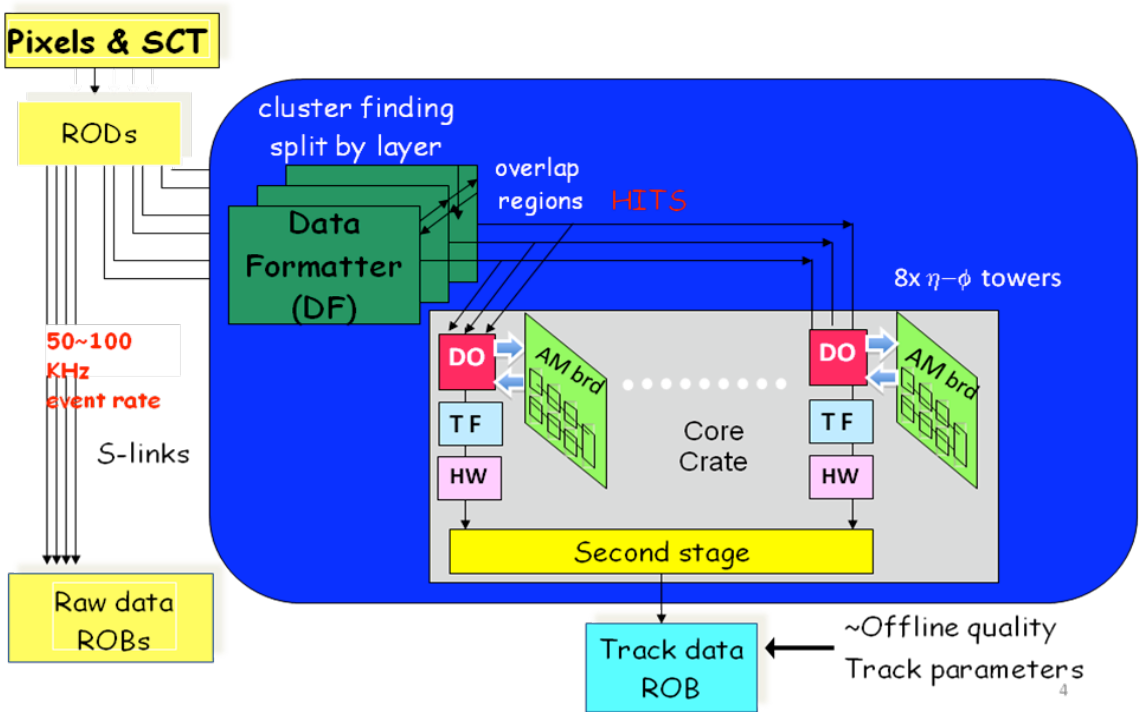

Figure 2.2: Functional sketch of an FTK core crate plus its data connections.

The FTK input bandwidth sets an upper limit on the product of the level-1 trigger rate and the number of silicon hits per event. We minimize this limitation by running the silicon hits on multiple 100 MHz buses within FTK. Nevertheless, in order to sustain a 100 kHz level-1 trigger rate, it is necessary to organize FTK as a set of independent engines, each working on a different region of the silicon tracker. The first step is to divide the detector into azimuthal regions. This segmentation generates some inefficiency at region boundaries that can be removed by allowing a small overlap region at the boundaries. At luminosities up to 3×10^{34} $cm^{-2}s^{-1}$, we plan to use 8 ϕ regions (45° wide) with an overlap of 10° to guarantee high efficiency for tracks with P_T of 1 GeV/c and above. Each region will have its own "core processor" contained in a 9U VME crate ("core crate"), for a total of 8 engines working independently. Each region and its processor are further subdivided into sub-regions, corresponding to η-ϕ towers, again largely independent and

with enough overlap to maintain high efficiency. The η range is divided into four intervals, and the region's ϕ range is divided again by two (22.5° plus 10° overlap). The overlap in η takes into account the size of the beam's luminous region in z. With this detector segmentation, we can distribute the pixel and SCT data on the 8 parallel buses at the full 100 kHz level-1 trigger rate with the detector occupancy expected for 3×10^{34} $cm^{-2}s^{-1}$ (see section 3.3).

The pixel and strip data are transmitted from the RODs on S-LINK fibers and received by the Data Formatters (DF) which perform cluster finding. The barrel layers and the forward disks are grouped into logical layers so that there are 11 layers over the full rapidity range (see figure 2.3). The cluster centroids in each logical layer are sent to the Data Organizers (DO). The DFs are not partitioned into regions (they are not in the "core crates"). They organize the detector data into the FTK η-ϕ tower structure for output to the core crates taking the needed overlap into account.

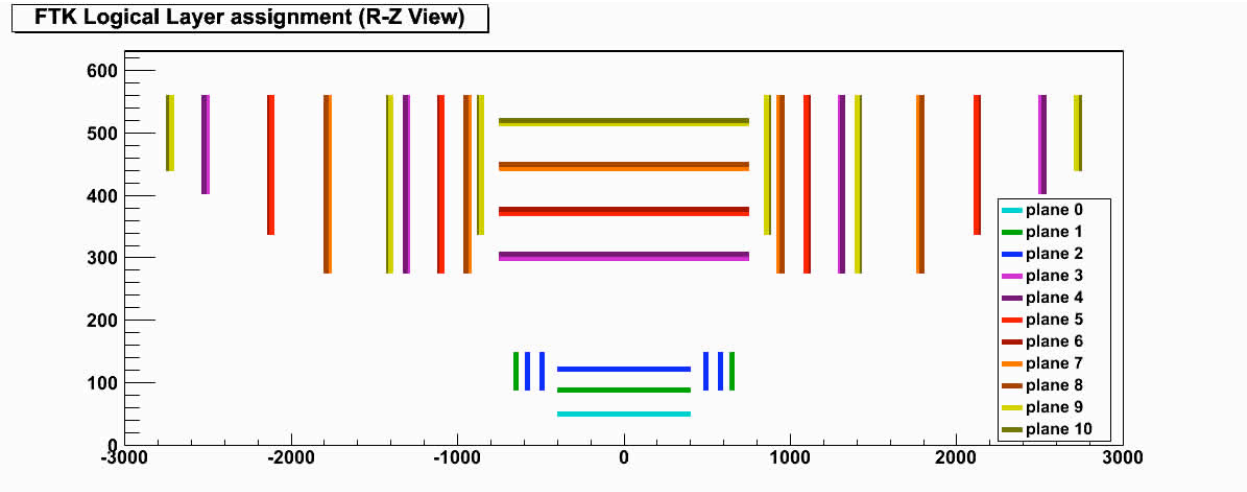

Figure 2.3: The assignment of barrel layers and forward disks to 11 logical silicon layers.

The DO boards are smart databases where full resolution hits are stored in a format that allows rapid access based on the pattern recognition road ID and then retrieved when the AM finds roads with the requisite number of hits. In addition to storing hits at full resolution, the DO also converts them to a coarser resolution (SuperStrips, SS) appropriate for pattern recognition in the AM.

The AM boards contain a very large number of preloaded patterns or roads corresponding to the possible combinations for real tracks of a superstrip in each silicon layer. These are determined in advance from a full ATLAS simulation of single tracks using detector alignment extracted from real data. The AM is a massively parallel system in that all roads see each silicon hit nearly simultaneously. When a road has found the requisite number of hit layers, the AM sends that road back to the DOs. They immediately fetch the associated full resolution hits and send them and the road number to the Track Fitter (TF). Because each road is quite narrow, the TF can provide high resolution helix parameters using the values for the center of the road and applying

corrections that are linear in the actual hit position in each layer. Fitting a track is thus extremely fast since it consists of a series of integer multiply-and-accumulate steps. In a modern FPGA, approximately 10^9 tracks can be fit per second.

Both the pattern matching and track fitting functions produce duplicates or "ghosts", duplicate roads in the former and duplicate tracks in the latter. Duplicate roads occur because we use majority logic in the pattern matching stage. FTK can require *N* fired layers among the total of *M* layers (*N*/*M*) to declare a road successfully matched. We plan to allow one missed layer in order to keep track-finding efficiency high. The use of the (*M*-1)/*M* matching criterion generates duplicate roads. For each real track, it is possible to find a single *M*/*M* road (a hit found on each layer) and/or a large number of (*M*-1)/*M* roads (a hit is missing on one layer). For the latter, all of the roads will have identical superstrips fired in the *M*-1 non-empty layers. If all of these roads are sent to the DO and TF, there will be a great deal of time wasted repeatedly fitting the same silicon hits. The Road Warrior function (RW), which was implemented in the CDF SVT [2], identifies and removes all but one road containing identical hit superstrips.

Duplicate tracks share most but not all of their hits. They occur when a track has non-associated nearby hits, either noise or coming from another track. If the replacement of real track hits by other hits still produces a satisfactory χ^2, then there will be duplicate tracks. The Hit Warrior function (HW) is applied after track fitting and reduces the duplicate track rate by keeping only the best χ^2 track among those that share at least a specified number of hits.

Now that we have high luminosity simulation, we find that the probability that a superstrip is empty is small enough to make the RW reduction in the number of roads a minor effect, while the importance of the HW is increased. Consequently we have removed the RW function from our baseline design and implemented the HW in each η-ϕ tower.

An FTK R&D program has been carried out for a number of years, and prototypes have been built. Details of their design and performance are given in [3] and [4] and in the references in those papers. Important R&D for FTK was also carried out in the context of the CDF SVT upgrade. A published paper [5] describes the upgrade and its relevance to future applications.

2.1.3 Summary of the FTK architecture evolution

As we noted above, the major change since the 3×10^{34} $cm^{-2}s^{-1}$ simulation became available is to replace the serial data flow in each core crate by data flowing in parallel through η-ϕ towers. Only those silicon hits within the tower pass through it. A tower consists of a DO, AM, TF, and HW. Only the tracks exiting the HW are sent out of the tower hardware.

Since the single DO, TF, and HW in the old architecture are now duplicated in each tower, the natural question arises whether the cost and physical size of the system is significantly increased. The answer is no. The rapid advancement in FPGA technology allows the DO, TF, and HW functions to each fit into a single chip. Section 2.3.2 describes how a tower, including the DO,

AM, TF, and HW functions, fits into a single core crate slot, with a main board and an auxiliary card on the back of the crate. We call the logic in a single slot a "Processor Unit". As a result, the number of crates in the system does not increase. The ability to place multiple functions in a single slot provides a significant advantage for the data flow between functions. These high rate transfers now occur on short PCB lines rather than across a complex high-speed custom backplane as needed with the old architecture. The cost also does not increase very much because the distributed functions can often be implemented on inexpensive FPGAs (Xilinx Spartan or Altera Cyclone families) or smaller sections of the very expensive top-of-the-line devices we needed in the old architecture (Virtex and Stratix respectively).

There is one potential disadvantage of the new architecture. Since the hardware is now partitioned among the η-ϕ towers, we could be more sensitive to large fluctuations in the numbers of hits, roads, and tracks in a tower. Previously the DO, TF, and HW were resources that were shared across the crate; now they are dedicated within each tower. However this is not a problem at high luminosity because the processing is dominated by pile-up hits rather than those from the hard scattering process (see figure 2.1). The computing power within each tower is large enough to handle the fluctuations at high luminosity and of course is much larger than needed at lower luminosity.

There is one other important change in the architecture. Previously pattern matching and track fitting were done with all 11 silicon layers in one step. The 3×10^{34} $cm^{-2}s^{-1}$ simulation shows this strategy to be untenable at very high luminosity. The number of matched roads, largely dominated by fakes and thus sensitive to detector occupancy, strongly depends on the width of superstrips. Fakes can be reduced by having very narrow roads, but that would require an AM size that today seems unaffordable for the 11-layer option (the AM bank size increases rapidly with the number of layers). In addition, since at high luminosity there are often multiple hits within a superstrip, the number of fits to be done within a road increases as a power of the number of layers. On the other hand, we must use all 11 layers to control the fake track rate at high luminosity. For these reasons, we now start track finding using a subset of the silicon layers but employ all 11 layers in the final fitting. This scheme allows the use of narrower SSs, reducing the number of matched roads. It also breaks the power law behavior of the number of fits because only good tracks are kept after the first step for use in the second step. We are studying two options, either of which is consistent with the overall architecture we described above.

A. **"7 layer" architecture** – We perform AM pattern matching using the 3 pixel layers and the 4 axial SCT layers. Those 7-layer tracks that pass a χ^2 cut are then extrapolated into the SCT stereo layers. This can be done very precisely because of the proximity of each stereo layer to the adjacent axial layer. The track location in a stereo layer can be localized to 3 silicon strips. We subdivide each stereo layer into very narrow superstrips (4 strips wide) and fetch 3 superstrips, the one found

by the extrapolation and the two neighboring SSs. Finally we perform the 11-layer fits using the 7 hits on the original track plus the stereo hits within the recovered SSs.

B. **"SCT 1st – Pixel 2nd" architecture** – Again there are two stages. Pattern matching and track fitting are first done in the 8 SCT layers. For each track that passes the χ^2 cut, a pseudohit is created for the second stage. The curvature, azimuth, and polar angle are each binned, and a single "pseudolayer" is created with these three parameters encoded into a SS. That pseudolayer and the 3 pixel layers go through a 4-layer AM, with the matched patterns then going into full 11-layer track fitting.

These two options have a very similar first stage, but differ in the second stage. That is why figure 2.2 shows only the first stage in detail. We will continue to study the two schemes with the high luminosity simulation until it is clear which is best. Since the differences in hardware and data flow are not large, our overall schedule will not be significantly affected.

2.1.4 Data flow inside the FTK pipelines

A uniform communication protocol is used for all data transfers in FTK. The data flow through unidirectional links connecting one source to one destination. The protocol is a simple pipeline transfer driven by a write enable (WEN) that indicates the clock cycles with a valid data word. To maximize speed, no handshake is implemented on a word-by-word basis. A hold signal (HOLD) is used instead as a loose handshake to prevent loss of data when the destination is busy. Data words are sent on the cable by the source and are strobed in the destination on the positive clock edge if WEN is active. Input data are pushed into a FIFO buffer. The FIFO is popped by whatever processor sits in the destination device. If the destination processor does not keep up with the incoming data, the FIFO produces an Almost Full signal that is sent back to the source on the HOLD line. The source responds to the HOLD signal by suspending data flow. Using Almost Full instead of Full gives the source plenty of time to stop. Since the source is not required to wait for an acknowledge from the destination device before sending the next data word, data can flow at the maximum rate compatible with the link bandwidth even when transit times are long. The standard FTK frequency is 100 MHz.

On each link there is a bit-field that is dependent on the kind of information being transferred. There is also a bit for WEN and an End Event bit (EE). The associated HOLD signal travels in the opposite direction, from destination to source. Sometimes there is also an End Packet bit (EP) if data are sent as packets of words. The EP bit marks the last word of the packet. The EE bit is used to mark the end of the data stream for the current event. The complete sequence of words in a data stream is called an Event. Each board will assert EE in its output stream after it has received an EE in each input stream and it has no more data to output. The last word in each Event (the EndEvent word) has a special format. It has EE=1, with the remainder of the word used for the event tag, parity, and error flags.

2.2 Physical description of the system

2.2.1 Crates and their contents

FTK as configured for 3×10^{34} $cm^{-2}s^{-1}$ consists of 13 crates. The Data Formatter boards, which receive the data from the pixel and SCT RODs and transmit cluster centroids to the core crates, occupy 4 crates. There are 8 core crates, each covering 45° in azimuth, and a final crate containing the final Hit Warriors and RODs.

Figure 2.4 shows the organization of a core crate for options A and B. In both cases, most of the slots are occupied by Processor Units which contain AM, DO, TF, and HW in a single slot. For option B, some of the processor units (blue in the figure) are used in the first stage (SCT tracking), with the others (in pink) used for the second stage (adding pixels to tracks). In option A, the Processor Units are used only at the first stage and they occupy most of the space; 16 slots are used to build a very large AM bank since the use of SCT and pixel layers together produces large banks. For option B, the Processor Units are almost equally divided between the first and second stages.

The output tracks from the many Processor Units for both options are collected, distributed, and refined by the boards in the yellow slots. In the case of option A, there are four second stage boards. They receive the 7-layer fit results from 4 Processor Units and the related stereo SCT hits from the back of the crate, perform the 11-layer fits, apply track cleanup, and send the tracks to the final crate. For option B, the two boards receive the final fits from the second stage, perform track cleanup, and send the tracks to the final crate.

The hits from the DFs are received on the back of the crate. For option A, the Processor Units receive hits from the SCT axial layers and pixel layers, while the "11-layer fit" boards receive hits from the stereo SCT layers. For option B, the blue Processor Units receive hits from all SCT layers and the pink Processor Units receive the pixel hits.

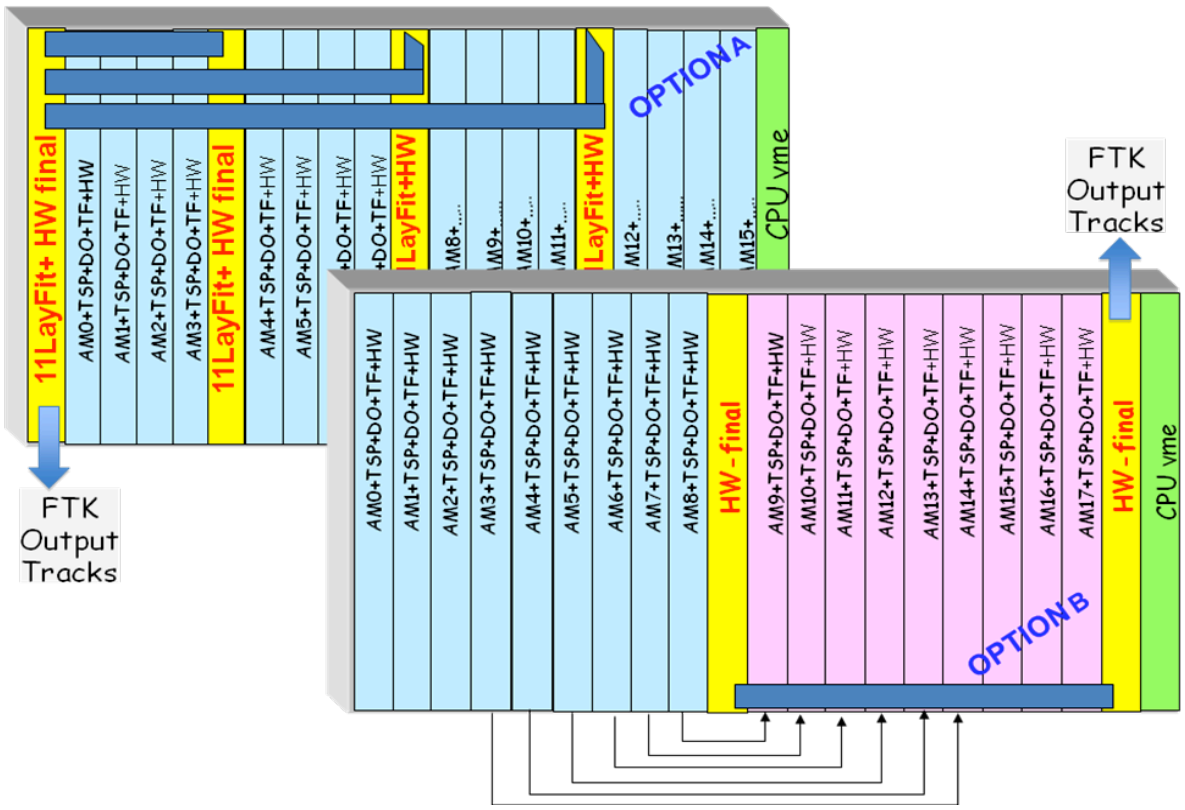

Figure 2.4: Layout of a FTK core crate for options A and B.

2.2.2 Data into and out of FTK

2.2.2.1 Data from the RODs to the Data Formatters

The pixel and SCT data are sent to the DFs from the RODs on standard ATLAS S-LINK optical fibers. The HOLA mezzanine cards in the RODs will be replaced by dual-output HOLAs which have two optical outputs that transmit identical data to the normal DAQ ROBINs and to the DFs. This is accomplished by sending the output of the serializer to two drivers, each of which feeds an optical transmitter. Unlike the primary path to the DAQ, the fiber to FTK does not have flow control enabled. Thus FTK cannot adversely affect the DAQ data transfer. To minimize the loss of FTK data, the DFs will have deep input buffers. If on very rare occasions some input data are lost, FTK will turn on an error bit to signify that the analysis of the event is incomplete.

Some years ago, we took the CERN HOLA design and modified it to have two output fibers. Prototypes were built and tested, and the board passed an ATLAS board review. The dual-output HOLA has been used in the CDF level-2 trigger system since that time. An unstuffed board is shown in figure 2.5 with the location of the second optical transmitter noted.

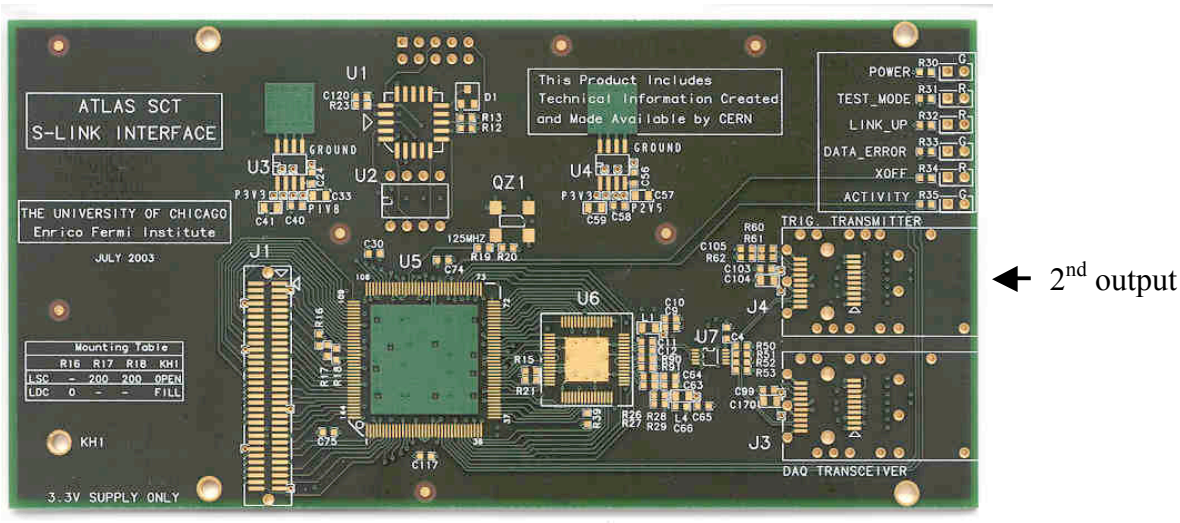

Figure 2.5: An unstuffed dual-output HOLA board showing the location of the 2nd output.

2.2.2.2 Data from the final crate to the ROSs

The event output from FTK consists of tracks, each containing 5 helix parameters, a goodness of fit, and 4 bits to indicate which layer, if any, does not have a hit on the track (we require no fewer than 10 layers with a hit). The latter is particularly important for *b*-tagging because the absence of a hit on the inner pixel layer affects the impact parameter resolution. Our simulation shows that there are on average ~300 reconstructed tracks with $P_T > 1$ GeV/c in events at 3×10^{34} $cm^{-2}s^{-1}$. The information per track will be 8 bytes for a total event size of approximately 2.5 KB.

Getting the FTK data into the level-2 processors presents a problem. It is important that the track information be available for all triggers, since at high luminosity tracking will be needed for e, μ, τ, and b-jet triggers, as well as whenever rapid determination of the high-P_T primary vertex is desired. That means FTK data will be requested at a 100 kHz rate, the expected level-1 limit for SLHC Phase I. All of the FTK data for an event fits easily in a single ROBIN, but the 100 kHz request rate has not been possible to date. We are discussing two options with ATLAS ROS experts.

One option is straightforward but requires more hardware. Multiple ROSs can be used with a lower data request rate for each. If there were 4 ROSs, FTK would send event data using the two least significant bits of the L1ID as the ROS ID number. A level-2 processor would know which ROS contains the data based on the L1ID.

The other option is to use a single ROS containing a single ROBIN fed by only one ROL. In a test carried out recently with ROS experts, it was found that the request rate for this configuration could exceed 100 kHz. There are some caveats however. The test was done with a new PC with the latest architecture. More serious is the fact that in the test there were 20 L2PU applications, so there were 20 connections to the ROS. In reality, there will be

approximately 4000 L2PU applications. It is not believed that a ROS PC can handle 4000 connections at a 100 kHz rate with the existing software. However there are efforts underway to optimize the software to improve the network traffic. Thus this option might work in the future.

For diagnostic purposes, we will want to transfer more data per event. This includes the road ID, the η-ϕ tower, and the local hit coordinates within the road's superstrips. This would amount to an additional ~5 kB at 3×10^{34} $cm^{-2}s^{-1}$. For a sampling of events, this can be sent from the final FTK crate CPU to a monitoring PC.

2.2.3 Need for Services

2.2.3.1 Racks and Power

With 2 crates per rack, FTK will need 7 racks. We won't know the total power and cooling needs per crate until the engineering design of each board is completed, but at this stage we believe the power demands in the DF and final crates will not exceed that of existing ATLAS electronic systems. The 8 core crates however will have larger power and cooling requirements primarily due to the power consumed by the large number of Associative Memory chips. Our current estimate is 5 kW per core crate and our cost estimate in section 5.4 includes core crates that provide power and cooling up to 5.5 kW. However our ASIC chip engineer has an idea for reducing the power consumption in the next AM chip by as much as 70%. Since there will be FPGAs on the Processor Unit AUX cards, it is important that there be good air flow through the AUX card region of the crates.

2.2.3.2 Connection to DCS

We will use the standard ATLAS crate with its connection to DCS for environmental monitoring.

2.2.3.3 Error handling

If an extremely large event causes an internal FTK buffer overflow, if the FTK execution time exceeds an allowed maximum, or if there is an internal parity error, FTK will set an error flag in the output stream so the level-2 processors know that FTK processing for that event was corrupted or incomplete. In the very rare case of a serious error, such as loss of synchronization in an SLINK data stream from a ROD, FTK will inform Run Control and then recover from the error.

2.3 Functional description of each board type

2.3.1 Data Formatter

FTK interfaces to the pixel and SCT detectors through the Data Formatter boards. The data from the dual-output HOLAs are received by the DFs, which cluster the silicon hits and distribute the cluster centroids to the Processor Units. Each DF will typically have 5 input fibers from the

current inner detector. There is space for one additional fiber per DF to allow for an increased number of Read-Out Links (ROLs) from the replacement B pixel layer compared with the current B layer. The DF distributes its silicon layer data streams to the η-ϕ towers it feeds. This includes exchanging data with neighboring DFs to replicate hits that are needed in multiple towers.

Figure 2.6 shows the structure of the Data Formatter boards and their data connections. Each DF receives data from the SCT and pixel detectors and forms projective towers that span all layers from the B-layer to the outer SCT layer. This data is distributed to 1 or 2 η-ϕ towers. Since there isn't an exact match between the ROD segmentation and the FTK η-ϕ tower segmentation, and because η-ϕ towers have silicon modules in common because of the length of the LHC luminous region, the DF boards will exchange data among nearby boards to ensure that each DF has all of the silicon data for its η-ϕ tower. The data is exchanged with the nearest neighbor in η or ϕ over the P3 backplane or other high-speed link.

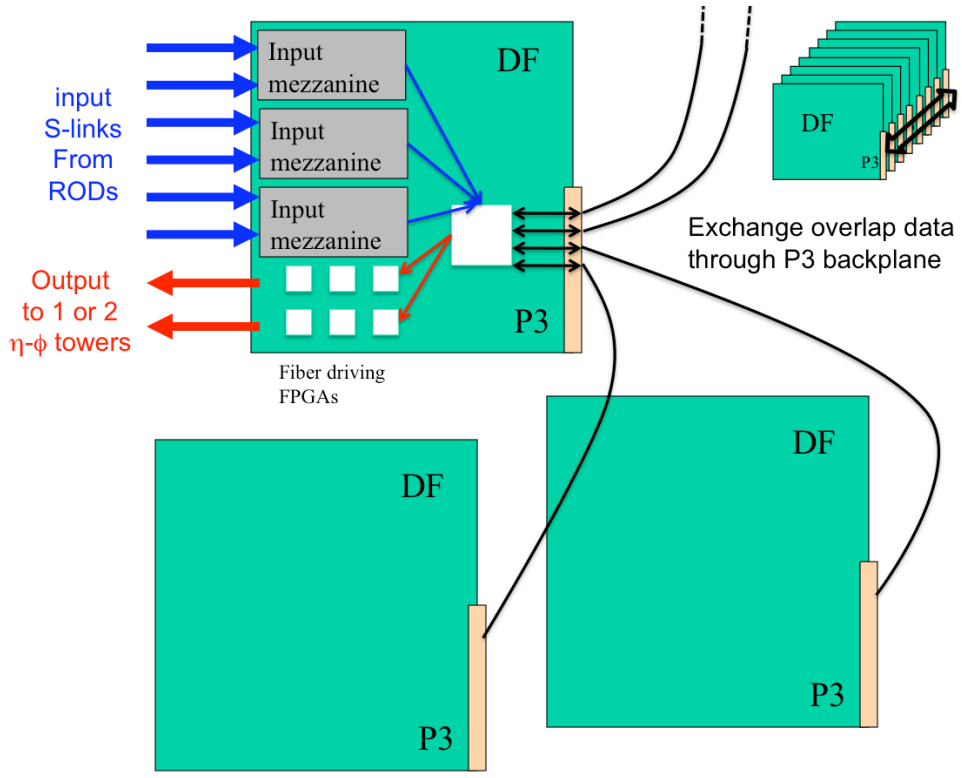

Figure 2.6: The Data Formatter boards with their data connections and their mezzanine cards for SCT and pixel input data.

The organization of the DFs is based on the ROD organization for the pixel and SCT detectors. The number of ROLs (one per ROD), taken from the ATLAS detector paper [6], is shown in table 2.1.

Subdetector	Partition	# of ROLs
Pixel	B layer	44
	Disks	24
	Layers 1 & 2	38 + 26
SCT	Barrel A	22
	Barrel C	22
	Endcap A	24
	Endcap C	24

Table 2.1: The number of ROLs coming from the silicon detectors.

Figure 2.7 shows how FTK towers are divided in η, taking into account the overlap needed to ensure full coverage. In order to efficiently distribute tower data, the DFs should match as closely as possible the η-ϕ towers in the core crates. For the SCT data, this is easy since the RODs provide hits in four η towers that match our tower structure quite well. For the pixels, the match is not as good because our forward η-ϕ towers include a large fraction of the barrel due to the length of the LHC luminous region. Consequently more pixel data is transferred between DF boards.

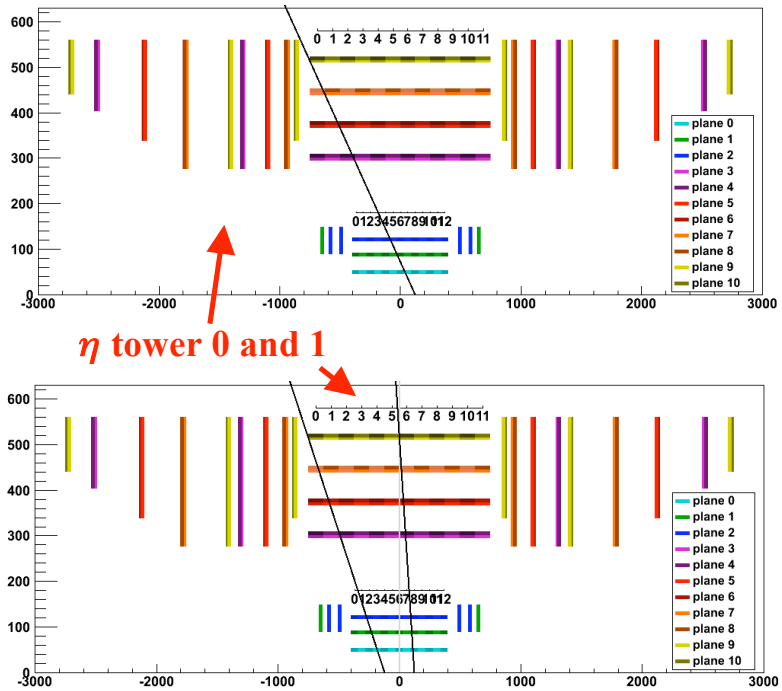

Figure 2.7: The η range for two of the four FTK towers in a ϕ slice. The other two are the mirror images of those shown.

FTK will have a total of 48 DFs, 12 for each of the 4 η regions. Since within each η region there are 16 η-ϕ towers, 2 for each of the 8 core crates, each DF will feed data to either 1 or 2 η-ϕ towers. Table 2.2 shows the number of input S-Link fibers for each of the 4 η regions. In the assignment of ROLs to DFs, the primary goal is to minimize the amount of data to be transferred between DFs.

# input ROLs	Endcap A	Barrel A	Barrel C	Endcap C
SCT endcaps	24			24
SCT barrel		22	22	
Pixel endcaps	12			12
Pixel Layer 2	1	12	12	1
Pixel Layer 1	7	12	12	7
Pixel B-Layer	11	11	11	11
# of DFs	12	12	12	12
Max # inputs/DF	5	5	5	5

Table 2.2: The number of input ROLs for the four η regions.

This table needs some explanation. For the SCT and pixel endcaps, the ROL to DF assignment is easy. On each side, there are 12 DFs each with 2 SCT links and 1 pixel link. For the SCT barrel, there are 22 ROLs per half-barrel. We assign 2 to each of 10 DFs and 1 each to the other 2 DFs. Pixel layer 2 has 13 ROLs per half-barrel. One each is assigned to the 12 half-barrel DFs, and the 13th is assigned to an endcap DF. The 19 ROLs per half-barrel in pixel layer 1 go one each to the 12 barrel DFs and 7 endcap DFs. Because of the large overlap among η regions in the B layer, its 44 ROLs are distributed uniformly among the four η regions. This association of ROLs to DFs is still under study and thus is preliminary.

For the overlap and output links, possible technologies are described in section 2.3.2.1. The SNAP12 transceiver will be used to transfer all data for one η-ϕ tower to a Processor Unit using a single link. For the overlap data transmission between DF boards, a set of serial LVDS connections can be used. A conservative approximation of the I/O needed for the overlap hits is between 50% and 100% of the total input hit bandwidth, which is 5 Gbps for the 4 S-Links entering a DF. The LVDS link described in section 2.3.2.1 has a 1.2 Gbps bandwidth. We will use four such pairs leaving a DF and four entering it. One pair each will communicate with each neighboring DF board.

2.3.1.1 Silicon hit clustering

Each DF will contain three clustering mezzanine cards, one for each pair of input ROLs. The mezzanine clusters the hit data and transmits the result to the DF motherboard.

Clustering for the SCT data is already performed in the ABCD chips. The DF will only have to identify clusters that cross chip boundaries in order to merge them.

For the pixels, a two-dimensional clustering algorithm capable of processing the 40 MHz hit rate from an ROL has been developed and simulated using Xilinx xc5vlx155 FPGAs. It associates consecutive hits into a cluster with a speed that is not affected by high hit density due to pile-up or collimated jets. After the hits are associated, the cluster center is computed with simple functions that account for the time over threshold. A detailed description of the algorithm is given in reference [7].

2.3.2 Processor Unit

The Processor Unit consists of a 9U VME board, the associative memory or AMboard, along with an AUX card on the back of the crate. The associative memory board has a long development history while the use of the AUX card containing multiple high-level functions is a new idea and needs prototyping. Figure 2.8 shows the last AMboard prototype [8] with a 9U AUX board developed in CDF added to the figure to show the relative sizes.

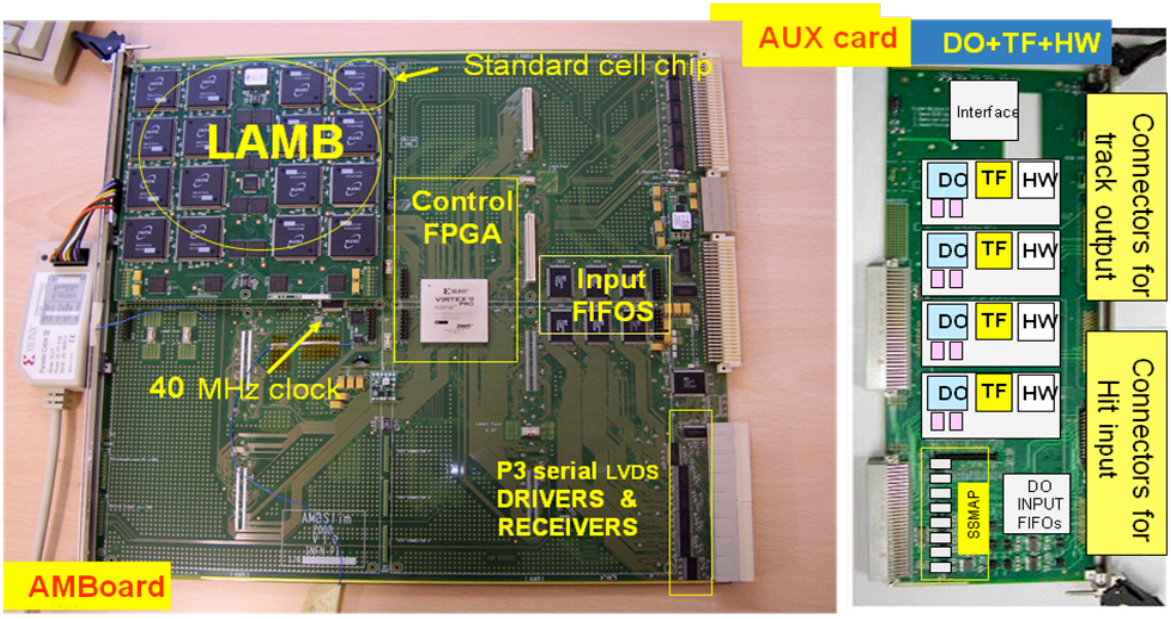

Figure 2.8: A previous generation AMboard prototype and an associated AUX card.

The major chips that will be on the AUX card have been sketched in to show that there is space for multiple DOs, TFs, and HWs. We will build them on small mezzanine cards so that we can mount between 1 and 4 of them. This will allow FTK's processing capability to grow as the

luminosity and thus detector occupancy increases. Each square on a mezzanine has the dimensions of an XC6SLX150, the top of the Spartan-6 series in a CSG484 package (19×19 mm, 330 pins). There will also be two interface chips on each AUX card, one controlling the input to the mezzanines and the other handling the output. These are the same Spartan-6 chips but with a larger number of pins (500) to allow point-to-point connections with each mezzanine. In addition to FPGAs, we need memory chips. We plan to use very dense DDR3 SDRAMs. The Spartan-6 has Integrated Memory Controller blocks that support DDR, DDR2, DDR3, and LPDDR. Data rates at present are up to 800 Mb/s with 12.8 Gb/s peak bandwidth [9]. The DDR3 SDRAMs are available today as 2 GB RAMs organized into 16×128 M-locations and they satisfy our needs perfectly (see section 2.3.2.2). They have a small package 96-ball FBGA (9×13.3 mm, only 1.1 mm thick) and can be placed on both sides of the mezzanine card. In the figure, they are shown as small pink rectangles around the FPGAs.

2.3.2.1 The AUX board and data received from the DFs

Silicon hits will be transferred from the Data Formatters to the Processor Unit AUX cards on cables operating at 100 MHz. Thus for a level-1 trigger rate of 100 kHz, the average number of hits per event per η-ϕ tower per cable cannot exceed 1000. We have partitioned each core crate into 8 η-ϕ towers so that the numbers of hits received by towers are approximately equal. Because of the variation in curvature for tracks of $P_T > 1\text{GeV/c}$, the length of the LHC luminous region, and multiple scattering, each of the 8 towers must receive more than 1/8 of the hits sent to the crate. For the SCT layers, the ratio of the number of hits in a tower to hits in the crate varies from 0.17 to 0.21. In the pixel layers, where the effect of the length of the luminous region is much greater, the ratio varies from 0.28 to 0.45.

The average numbers of clusters per event to be sent to each core crate are shown in Table 2.3. They were obtained from WH events at 3×10^{34} $\text{cm}^{-2}\text{s}^{-1}$. We find that the numbers are unchanged when we use dijet events, confirming that the occupancy is dominated by pile-up.

Pixel 0:	HITS = 2184±26
Pixel 1:	HITS = 1992±23
Pixel 2:	HITS = 2356±28
SCT 0 axial:	HITS = 1877±22
SCT 0 stereo:	HITS = 1879±22
SCT 1 axial:	HITS = 2024±24
SCT 1 stereo:	HITS = 2019±25
SCT 2 axial:	HITS = 2069±24
SCT 2 stereo:	HITS = 2069±24
SCT 3 axial:	HITS = 2128±25
SCT 3 stereo:	HITS = 2113±25

Table 2.3: The average number of hits per logical silicon layer per core crate for WH events at 3×10^{34}.

For the SCT layers there is not a problem even if we transmit the data from each layer on a single 100 MHz cable, since the partitioning into 8 towers reduces the number of hits on an individual cable by approximately a factor of 5. This brings the hit rate more than a factor of two below the maximum allowed. For the pixel layers, we maintain the same margin by using two 100 MHz links per layer.

Option B has the easier data connection problem. The SCT Processor Units (blue in figure 2.4) would receive 8 links, one each from the 8 SCT layers, while the pixel Processor Units (pink in figure 2.4) would receive 6 links, two each from the 3 pixel layers. Option A is more challenging since its Processor Units each receive 10 links, one each from the 4 SCT axial layers and two each from the 3 pixel layers. We have not yet selected a technology for the data transfer, but there are a number of viable solutions. Here we present two possibilities.

1. Serial LVDS is quite compact. We could use National Semiconductor SDS92LV16 serializer/deserializers which send 16 TTL signals on a single pair at a 16-bit word transmission rate up to 80 MHz (see figure 2.9). We expect that the rate will exceed 100 MHz by the time we produce the final FTK boards. The chip is an 80-pin LPFP package, 14×14 mm, 1.4 mm thick. A full resolution hit plus the EE and Write Enable bits will fit in a 24-bit word. The SCT will not need quite so many, and the pixels will need a few more. However an SCT hit plus a pixel hit will fit in 48 bits. So 3 LVDS pairs and 3 chips will handle an SCT layer plus a pixel layer. We have 4 of these SCT-pixel pairs (the 4 SCT axial layers used in option A and 2 links each for pixel layers 1 and 2). The two links for pixel layer 0 require an additional 3 LVDS pairs, for a total of 15. These can fit into two 14526-EZHB-XXX-0QC .050 Mini D Ribbon (MDR) Cable Assemblies for High Speed Digital Data Transmission, each of which contains 10 pairs (see figure 2.9). In option A each of the 8 η-ϕ towers is processed by 2 Processor Units, so the data received by the deserializer are immediately fed back into the serializer to be sent to the adjacent board.

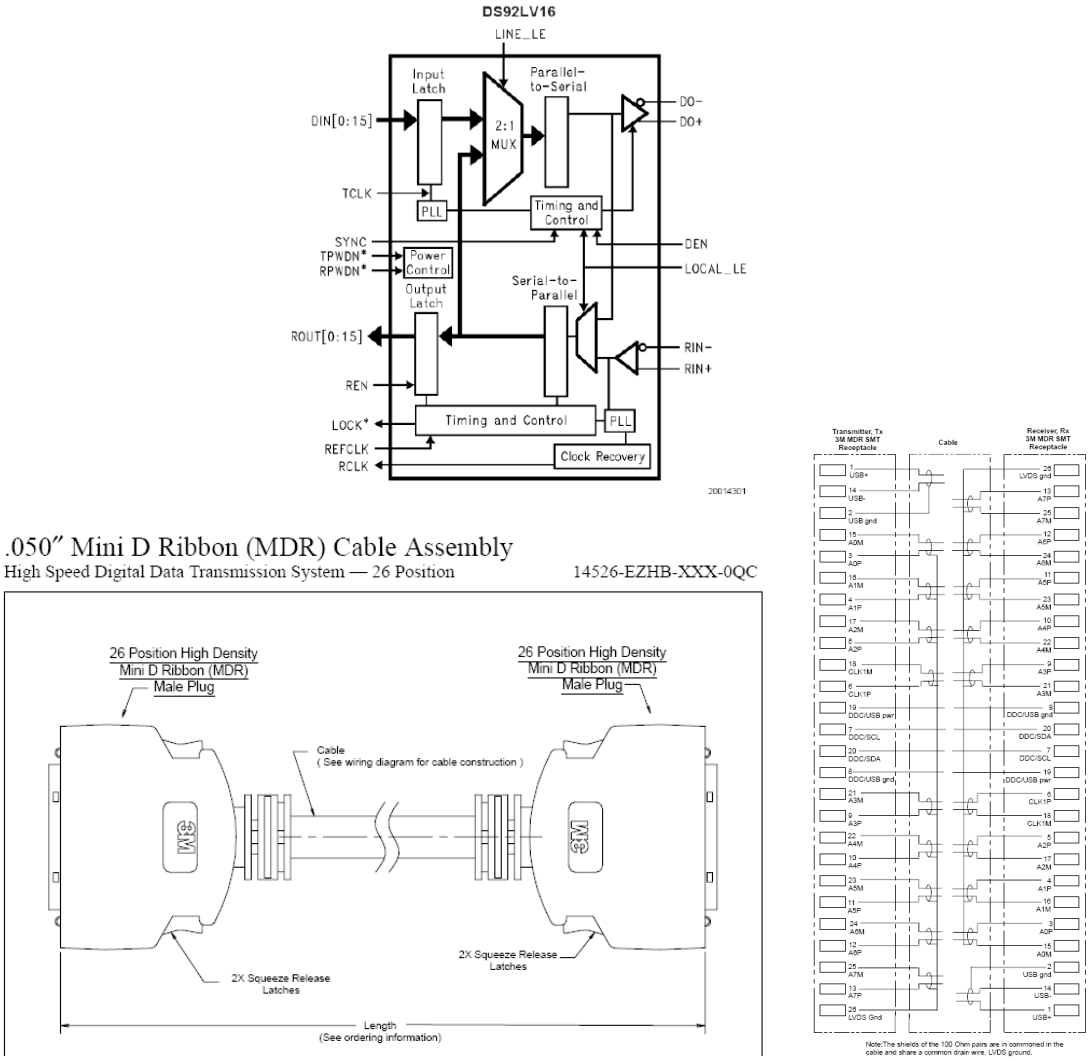

Figure 2.9: The LVDS serializer/deserializer chip (top) and high speed cable (bottom).

The cables are available in lengths up to 10 m. The only problem is their rigidity. They have been successfully used in the level-1 100 MHz trigger in the MEG experiment with similar serializers. That collaboration has tested the cables up to 600 MHz, while we want to use them at higher frequency (1.2-1.6 GHz). Figure 2.10 shows the 9U VME board where 11 such connectors are installed.

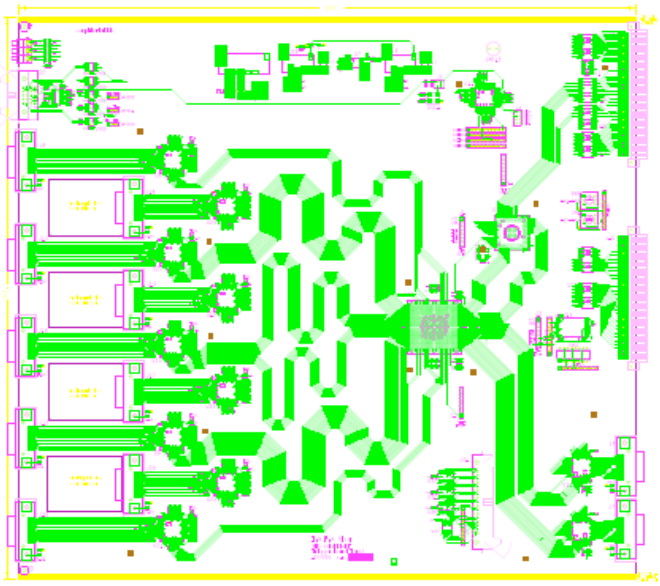

Figure 2.10: A 9U VME board from the MEG experiment with 11 connectors for high speed cables.

2. Fiber optic transmission is currently the best option. We can use the SNAP12 Transmitter and Receiver modules to provide high performance multi-channel optical links. They are designed for very short reach (VSR) high-speed data communication and computing applications where bandwidth bottlenecks are inherent. In terms of data transfer rates, they offer the lowest cost solution and the highest packing density. Common devices contain 12 independent optical channels, each capable of transmitting 3.5 Gbps up to a distance of 300 m on 50 micron multimode optical fiber operating at a wavelength of 850 nm. These modules have an aggregate link bandwidth in excess of 40 Gbps. They are optimized for applications that require line rates of 2.5, 2.7, or 3.3 Gbps, and they are perfect for our application.

   We have 10 buses (4 SCT + 6 pixels using 2 buses per pixel layer). Even using 25 or 27 bits per bus, more than we specified for the LVDS case, we would only need the lower available line rates (2.5 or 2.7 Gbps) for a 100 MHz word transmission rate. There is also space for 2 additional buses when the IBL becomes operational. A single connection, as shown in figure 2.11, will be enough for the entire transfer. The transmitter and receiver mechanical outline is shown in figure 2.12. It is compact and fits well on our AUX card since it is only 17 mm wide.

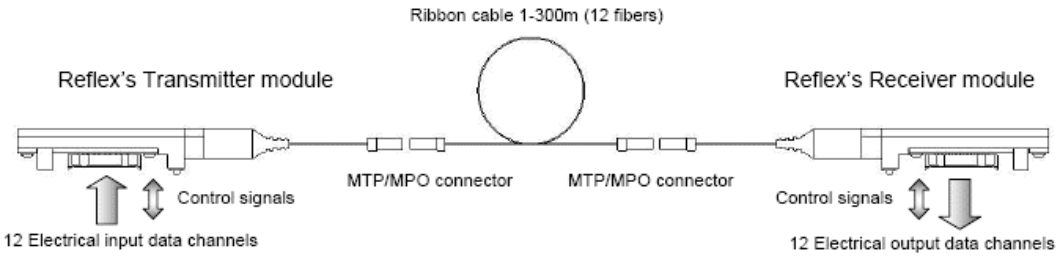

Figure 2.11: A complete point-to-point 12-channel optical link.

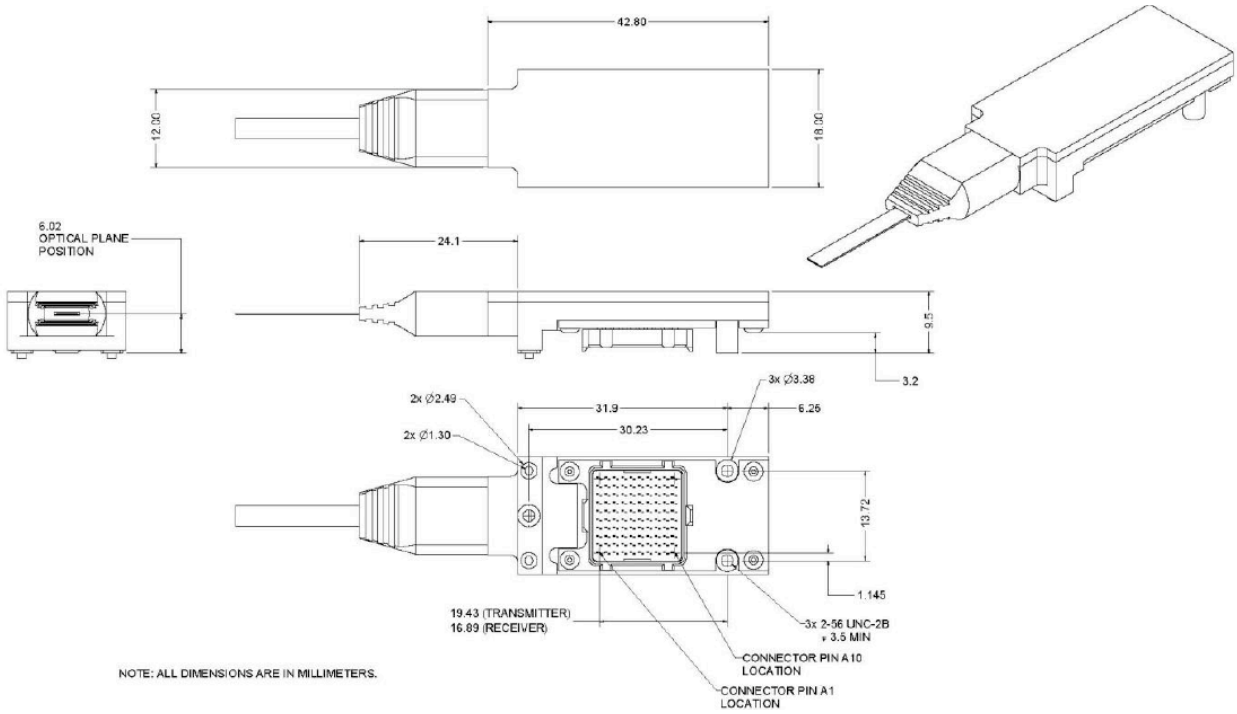

Figure 2.12: The optical transmitter and receiver mechanical outline.

The transmitter module converts parallel electrical inputs into parallel optical signals using a laser driver and a vertical cavity surface emitting laser (VCSEL). The transmitter module accepts both low voltage positive emitter coupled logic (LVPECL) and current mode logic (CML) input. All input data signals are differential and internally terminated. Figure 2.13 shows the functional block diagram of the transmitter module. The control block ensures proper operation of the device and provides individual channel settings and monitoring.

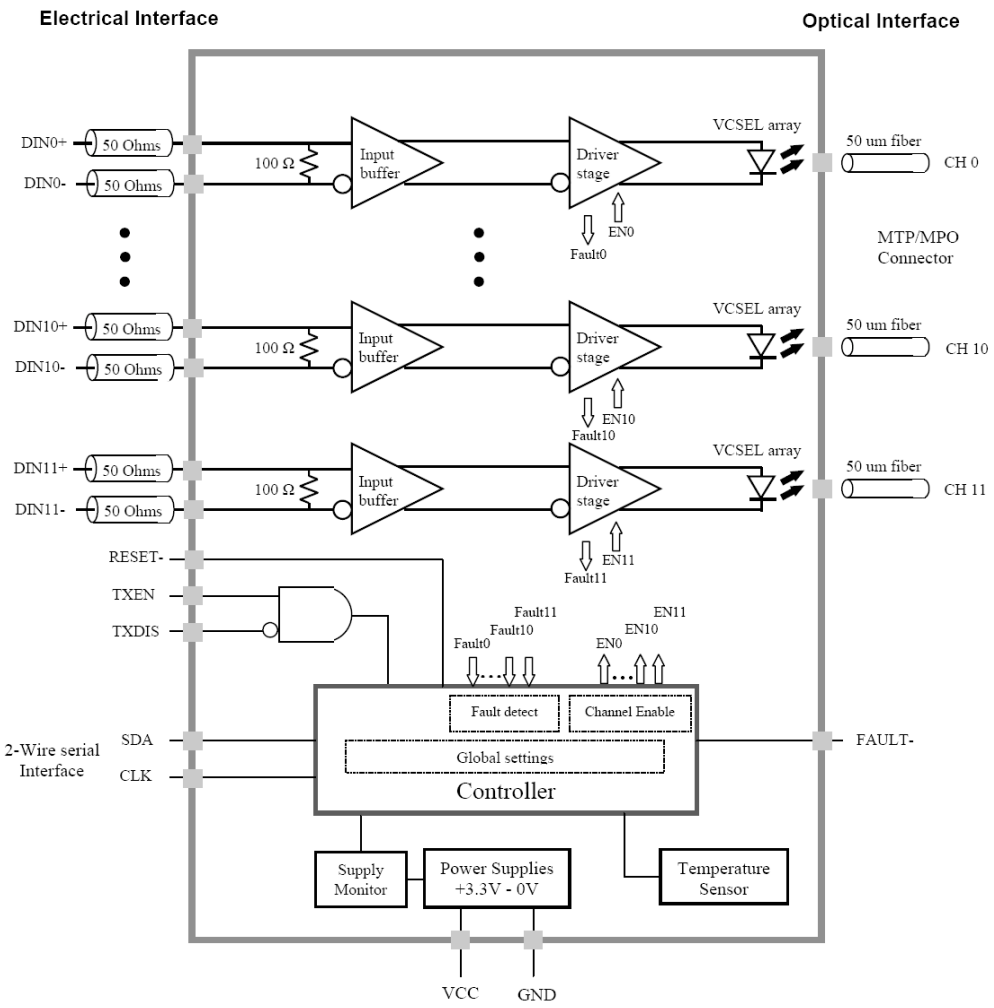

Figure 2.13: The functional block diagram of the transmitter module.

The receiver module converts parallel optical inputs into parallel electrical outputs using a photodiode array. The output signals are differential with voltage compatible with current mode logic (CML) levels.

Inexpensive FPGAs (Spartan or Cyclone) can be connected directly to the transmitter or receiver module using their serial inputs/outputs. For example, the small Spartan-6 chip (15×15 mm) is available with 4 such serial links, GTP transceivers that are highly configurable and tightly integrated with the programmable logic resources of the FPGA. It provides a number of features important for our application: (a) current mode logic (CML) serial driver/buffers with configurable termination and voltage swing; (b) support for line rates from 2.45 Gbps to 3.125 Gbps; and (c) built-in optical PCS features such as 8B/10B encoding, comma alignment, channel bonding, and clock correction. Surrounding the optical module with 3 small FPGAs with 4 GTPs each will allow us to transmit/receive data on the 12 different lines.

Such optical links could also be used to transfer input data between the two AM boards of the same η-ϕ tower in option A or transfer data between the two stages in option B.

2.3.2.2 The Data Organizer: a smart traffic node

The Data Organizer receives the hits from the Data Formatters and stores them in a database organized for rapid retrieval. It is analogous to a library where books arrive in arbitrary order and are quickly organized in bookshelves by subject. When a subject is later requested by a user, all the books in that section are quickly found and delivered. In our case, the "subjects" are track roads. Hit coordinates are mapped into superstrips, coarser resolution hit locations appropriate for the pattern matching stage. These SSs are sent to the Associative Memory. Each matched road is sent by the AM back to the DO, which rapidly extracts all of the full resolution hits in the road's SSs and transmits them to the Track Fitter.

Within the DO, hits are stored in the Hit List Memory (HLM). A structured database is built on the fly so that a road number can then be used as a key to directly access lists of hits. Incoming hits are sorted by superstrip with each superstrip having its own Hit List. Figure 2.14 shows the logic controlling the Hit List Memory. Data Organizer operation is divided into two phases: the Write Mode (the red lines in the figure show the data path in this phase) and the Read Mode (blue lines). During the Write Mode, hits are copied from the input into the Hit List Memory based on SS number. Since a road consists of one SS in each silicon layer, the HLM stores data separately for up to 8 layers. This is sufficient for Option A or each of the two stages in Option B. During the Read Mode, each road received by the DO causes the data in the Hit Lists for those superstrips to be transmitted on the output stream.

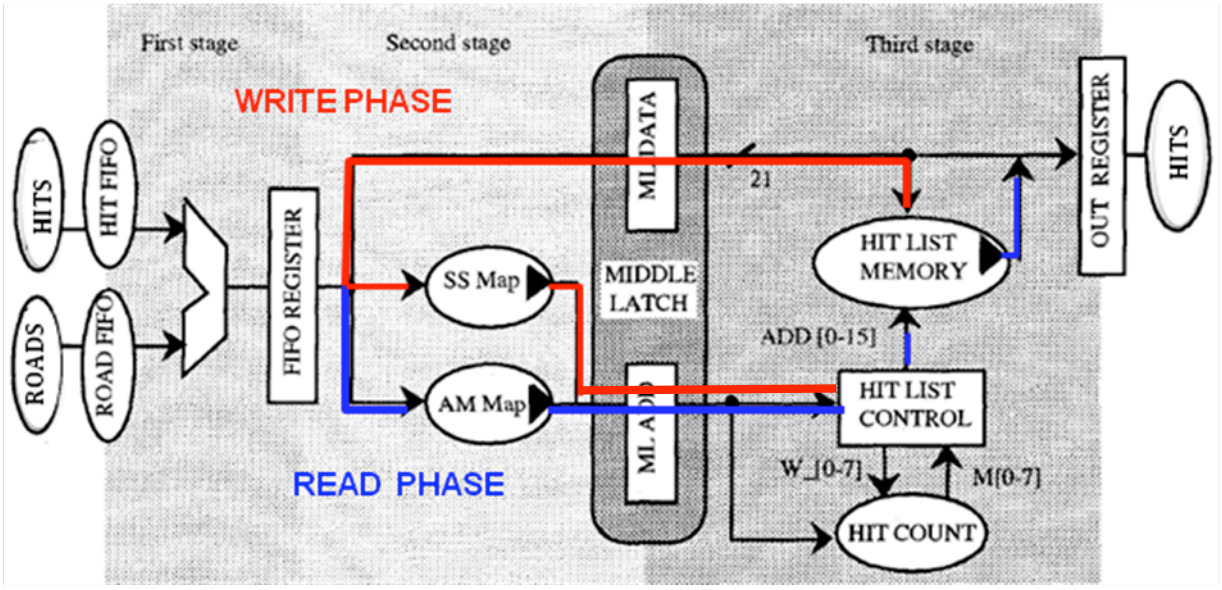

Figure 2.14: The Hit List Memory addressing and control in the write (red) and read (blue) modes.

To provide very fast access to the Hit List Memory, the DO has large look-up tables where pointers into the HLM are stored. The SuperStrip Map (SS Map) provides the starting address of the Hit List where hits in a superstrip will be written in Write Mode, while the Associative Memory Map (AM Map) provides the starting addresses of the Hit Lists to be sent on output for each road in Read Mode. The information in the SS Map and the AM Map are essentially the same, but the two maps implement different database structures with different addressing schemes.

An overview of the DO logic as used in the past in CDF [16] and in FTK R&D [17] is shown in figure 2.14. Input data are asynchronously pushed into two sets of FIFO buffers, the Hit FIFOs and Road FIFOs, by their upstream modules, DFs and AMs respectively. These are read via FIFO Controllers that convert the FIFO protocol to the DO internal synchronous pipeline protocol. Data flow from the two FIFOs to the output is coordinated by Finite State Machines (FSM). Data from the input FIFOs are latched in the Middle Latch (ML) which separates the look-up tables (SS Map and AM Map) from the Hit List Memory. Middle Latch is subdivided into two sections: ML DATA and ML ADD. When in Write Mode, ML DATA contains the hit to be written in the Hit List Memory and ML ADD contains the pointer to the Hit List where it will be written (this information is fetched from SS Map). When in Read Mode, ML DATA contains the road number and ML ADD contains the pointer to the Hit List word to be fetched and sent to the output stream. During normal operation, the DO starts each event in Write Mode, switches to Read Mode after all input data have been stored, and stays there until the end of the event when it switches to Write Mode again.

We plan to duplicate the HLM and completely separate the Write and Read paths so that we can execute the Write and Read paths in parallel on successive events. In figure 2.8, the input FIFOs are shown in the lower right of the AUX card along with the SS Map chips, one per layer, which are the grey chips between the FIFOs and the P3 connector. The Hit List Memories are contained inside each DO chip. Thus there are 4 HLMs per AUX card, one for each LAMB card on the AM board (see section 2.3.2.4). An HLM needs less than 3.5 Mb and fits in the XC6SLX150 which has 4.8 Mb in block RAM. However doubling the HLM in each DO chip would require chips with more memory like the smallest devices in the Virtex-6 family.

The AM Map fits in a single DDR3 chip (128M×16 bits) where the SSs are organized into 8 contiguous 16-bit words. Each LAMB contains fewer than 15M roads, so an AM Map for a LAMB requires 16 bits x 8 SSs × 15M roads = 120M × 16 bits. Multiple AM Maps are provided for each DO (small pink chips adjacent to the DOs in figure 2.8 and on the other side of the mezzanine cards) so that, even if fetching the 8 SS words from the DDR3 takes 20-25 ns, successive roads can be handled in parallel using the other AM Maps.

2.3.2.3 The AUX card and data distribution to the AM board

The AM Board prototype we have now receives 6 hit buses of 18 bits each. The width of the word exceeds our needs, so we will use narrower buses in the future. Without yet taking into

account the partitioning of a crate into towers, we need 13-bit SS numbers for Pixel 0 and each of the SCT layers. Pixel 1 and Pixel 2 each need 14 bits.

We will need a maximum of 8 buses for either option A or B. Here again we focus on option A since it is the most difficult; if its needs can be satisfied, there will be no problem with option B. We also note that since the occupancy in the inner pixel layer is expected to double when the IBL is installed, we will assign 2 buses to Pixel 0.

The SS number is immediately found from a hit via the SS Map. We will pass to the AMBoard a superstrip number only once per event, the first time there is a hit found within it. In table 2.4, we list the average number of SSs found in each region for WH events at 3×10^{34} $cm^{-2}s^{-1}$. Since the partitioning of the region into 8 towers produces a gain of at least a factor of 2 for the inner pixel layer and a factor near 5 in the external SCT layers, the number of SSs per layer will be below 1000 per event. This means we will be able to handle an event rate of 100 kHz using 100 MHz links. In fact, the chips we plan to use (Texas Instruments SN65LVDS93A) work up to 135 MHz. They serialize 28 TTL signals into 4 LVDS pairs (plus a pair for the clock). They are functionally equivalent and pin compatible with the serializer/deserializer we have been using (http://www.national/com/pf/DS/DS90CR288A.html). They pass the hits from the AUX card to the AM Board through the P3 connector. Each word to be transferred consists of the SS bit field and the EE and WEN bits as described in section 2.1.4.

Pixel 0:	HITS=[2184±26]	SS=[1629±16]
Pixel 1:	HITS=[1992±23]	SS=[1591±17]
Pixel 2:	HITS=[2356±28]	SS=[1903±21]
SCT 0 axial:	HITS=[1877±22]	SS=[1659±18]
SCT 1 axial:	HITS=[2024±24]	SS=[1796±20]
SCT 2 axial:	HITS=[2069±24]	SS=[1842±20]
SCT 3 axial:	HITS=[2128±25]	SS=[1888±21]

Table 2.4: The number of hits and superstrips per logical layer for WH events at 3×10^{34} $cm^{-2}s^{-1}$.

We have 6 buses (4 for SCT and 2 for Pixel 0) of 16 bits (14 SS bits + EE + WEN) and 2 buses (Pixel 1 and Pixel 2) with 17 bits (15 SS bits + EE + WEN) for a total of 130 bits which fit into 5 or more conveniently 6 chips. Six is the number we currently have on the prototype board. Figure 2.15 shows how the 5 LVDS pairs (4 data pairs plus a clock pair) from a chip are organized into a 4-column section of the P3 connector. The 6 chips would take a total of 20 columns. The P3 connector will be built with 5-row 2mm Hard Metric PCB Mount Connectors (Z-PACK 2mm HM). The prototype we have now has an 8-row connector but we prefer to compact the signals and use a 5-row connector in the future.

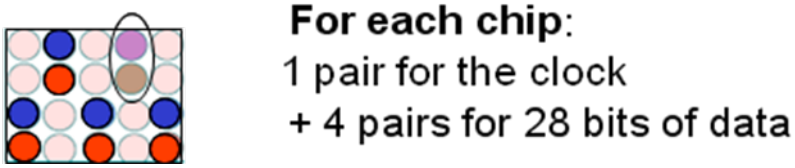

Figure 2.15: The data and clock pairs on the AM P3 connector. The pink pins are ground.

2.3.2.4 The Associative Memory Board

Figure 2.16 is a functional sketch of the existing AM Board. It has a modular structure consisting of 4 smaller boards, the Local Associative Memory Banks (LAMB), one of which is shown in the top left quarter of the figure. We are using the LAMB developed during FTK R&D [10]. Each LAMB contains 32 Associative Memory (AM) chips, 16 per face. The AM chips are standard cell devices developed for CDF [11] with FTK in mind. They are in PQ208 packages and contain the stored roads with readout logic. They are connected into four 8-chip pipelines on each LAMB. Roads with the required number of hit layers flow down in the 4 pipelines and are collected and merged into a single stream by the GLUE chip shown on the top of the LAMB. The motherboard has flexible control logic placed inside a single very powerful FPGA chip (Virtex II pro xc2vp100 with a 1696-pin package [9]) visible in the center of the figure.

An End Event word separates hits and roads belonging to different events. Different inputs have to be synchronized at the input of the AM Board. The board input is provided with deep FIFOs for this purpose. If, occasionally, a FIFO becomes “Almost Full”, a HOLD signal is sent to the upstream board, which suspends the data flow until more FIFO locations become available. The Almost Full threshold is set to give the upstream board plenty of time to react.

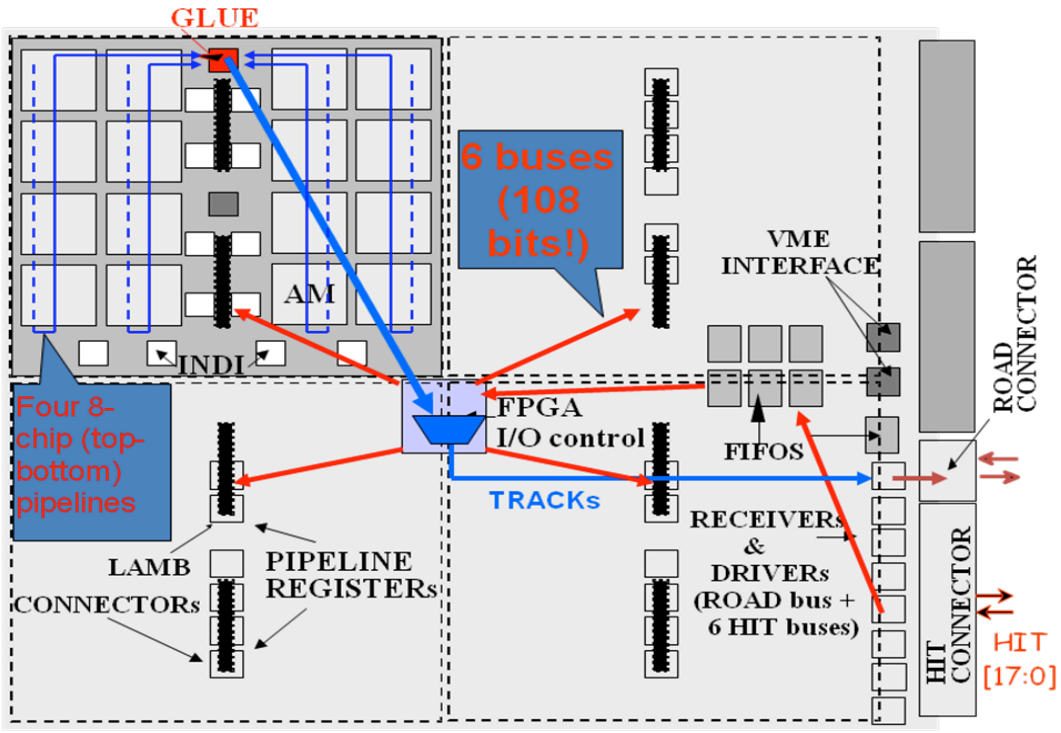

Figure 2.16: Functional sketch of the Associative Memory Board

When an AM board starts to process an event, the SSs are popped in parallel from the six hit input FIFOs. The popped SSs are simultaneously sent to the four LAMBs (the red arrows in the figure show the incoming hit path). Data from different input streams are checked for consistency. Upon detection of different event sequences, a severe error is signaled and the whole system must be resynchronized. As soon as hits are downloaded into a LAMB, locally matched roads set a request to be read out from the LAMB (Output Ready). When the End Event word is received on a hit stream, no more words are popped from that FIFO until the End Event word is received on all hit streams. Once the event is completely read out from the hit FIFOs, the LAMBs make the last matched roads available within a few clock cycles. The control chip collects matched road numbers from the LAMBs and sends them to the output (the blue arrow on the figure shows the output road path).

We plan several changes to this prototype. (1) The single large and very expensive FPGA will be replaced by 3 cheap Spartan-6 chips. One of them will replace the 6 input FIFOs and will be placed near the P3 connector; a second one will distribute the input buses to the LAMBs and will remain in the board center; the third one will handle the road output stream and be where the FIFOs are now. (2) We will place Spartan-6 chips with DDR3 RAMs below the LAMBs to process the AM output before sending roads to the Data Organizer (see section 2.3.2.6). (3) We will provide up to 4 parallel serialized links between the LAMB GLUE chip and the motherboard to increase the maximum LAMB output rate to 400 MHz (each LAMB will be able to output on average 4000 roads per event at a 100 kHz event rate, using 100 MHz links). This

powerful connection through the motherboard connector will be necessary at the highest luminosities, when AM total cost limits our ability to continue to decrease the road size as occupancy increases. We will allocate 20 TTL bits in the LAMB connector to the road output to implement the multiple serial links working at 100 MHz. This implies also a change to the LAMB mezzanine where the SPLD GLUE chip will have to be replaced by a Spartan-6 to provide the serial links (GTPs) or the smallest Virtex-6 (XC6VLX75T-2FFG484CES) for GTX use.

2.3.2.5 The LAMB and the Associative Memory chips

The hit buses are fed in parallel to the four LAMBs and distributed to the 32 AM chips on each LAMB through 12 fanout chips called INput DIstributers (INDI), each chip on the prototype sending one bus to 16 AM chips. For reading out the matched roads, the AM chips on each LAMB are connected into four 8-chip pipelines. A pipeline consists of 4 chips on the top side and 4 chips on the bottom side as shown in figure 2.17. Each AM chip receives an input road bus from its upstream chip, and it multiplexes the input roads with its internally matched roads onto a single output bus to its downstream chip. Signals propagate from chip to chip on the board surfaces through very short pin-to-pin lines using no vias. The outputs from the 4 AM pipelines are then multiplexed into a single road bus by specially-designed GLUE chip on each LAMB. Patterns are downloaded into the AM chips by programming them through the VME controlled JTAG port. Chains of 4 AMs each are downloaded in parallel to reduce programming time. The VME 32-bit wide data transfer allows 32 chains to be programmed in parallel, *i.e.* all of the LAMBs on an AM Board. Downloading time is measured to be a few seconds.

We will replace the CPLD GLUE chip (xc2c512-7FT256, 17×17 mm) with the biggest Spartan-6 or the smallest Virtex-6 chip (XC6VLX75T-2FFG484CES, 23×23 mm) to allow GTP or GTX serial data transfer and to transform the four 8-chip pipelines into eight 4-chip shorter pipelines.

The AM chip stores a large bank of pre-calculated roads or patterns. In principle the pattern bank should contain all possible tracks that go through the detector (a 100% efficient bank). Of course one must consider effects that make particles deviate from the ideal trajectory, such as detector resolution smearing, multiple scattering, etc. These effects generate a huge number of extremely improbable patterns, which blow up the bank size. For this reason, we will use a bank that is partially inefficient. For details about the generation of the bank, see section 3.1. We use simulation to generate tracks in the detector and then we convert them into patterns. The pattern bank is always generated to take advantage of all the pattern space available in the hardware. The pattern recognition resolution, *i.e.* geometrical width of the patterns, is tuned to get a reasonably high pattern bank efficiency (~97%) using all patterns.

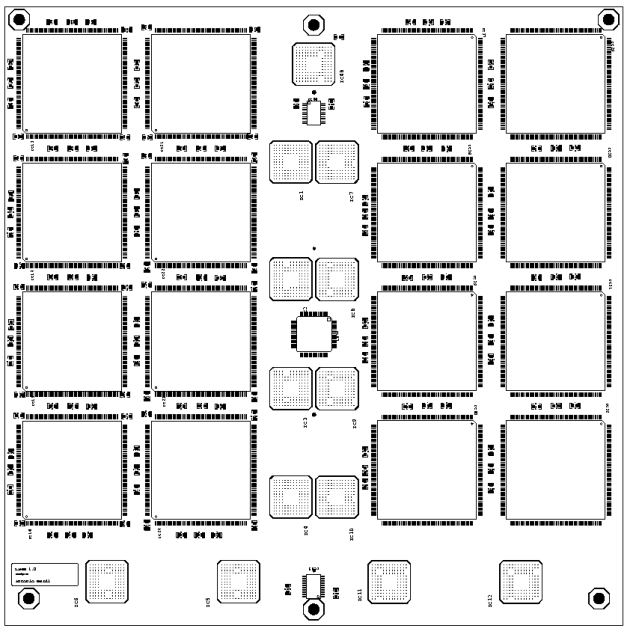

Figure 2.17: Top view of a LAMB board. Sixteen AM chips are arranged into a 4×4 array. On the bottom side, not visible in the figure, sixteen additional chips are arranged in the same positions. Each array column, with the corresponding column on the other board side, constitutes an 8-stage pipeline for reading out matched roads. The small squares full of tiny pads are the locations of additional chips.

A track candidate is found whenever all the hits in a road, or a majority of them, are present in the event. The minimum number of layers that have to be hit for a road to be declared matched is a free parameter and is set in a control register as a single programmable threshold common to all roads in the chip. Threshold comparison is introduced to account for silicon detector inefficiency. The CAM technology, which allows this search to be performed in just a few clock cycles, is obviously the key component of the AMchip03.

It must be emphasized that with respect to commercially available CAMs, the AMchip03 has the unique ability to search for correlations among input words received at different clock cycles. This is essential for tracking applications since the input words are the detector hits arriving from different layers. They arrive at the chip, serialized on six buses, without any specific timing correlation. Each pattern has to store each fired layer until the pattern is matched or the event is fully processed and thus patterns can be reset.

The Associative Memory allows the simultaneous comparison of all stored patterns with the current event. The existing AM chip performs pattern recognition in a detector of up to 6 layers. The 0.18 micron CMOS process (with 1 poly and 6 metal layers), available from the silicon foundry UMC, was chosen. Figure 2.18 gives a block diagram of the complete device (see [11] for details).

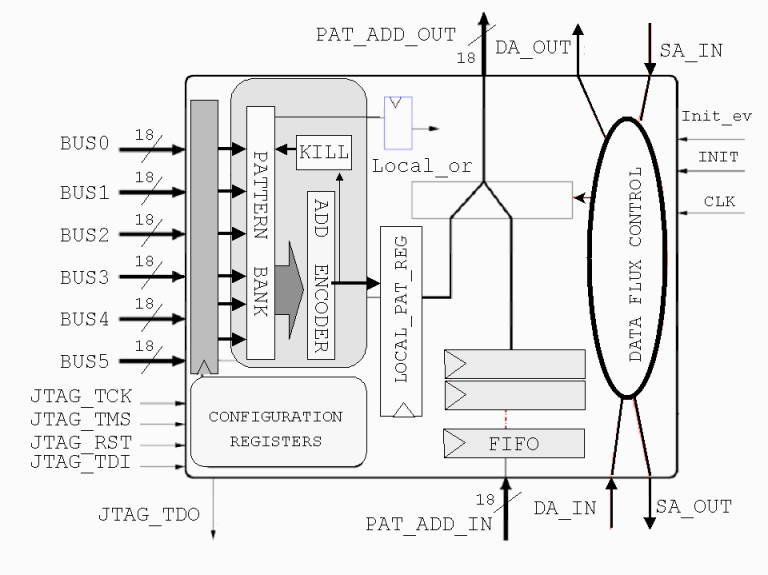

Figure 2.18: Block diagram of AMchip03. The pattern bank, readout, and control logic are emphasized.

The pattern bank uses approximately 80% of the silicon resources in the device and contains 5120 patterns, corresponding to a total of approximately 500,000 content-addressable memory bits. In addition to the default 6-layer mode, the chip can be configured to perform pattern recognition for a detector of up to 12 layers, combining pairs of 6-word patterns into a 12-word pattern. In this case the 18th bit of each data bus is used to distinguish hits coming from two layers, multiplexed on a single bus. With the 12-layer configuration, the number of available patterns is 2560.

Using this existing chip we can build a pattern bank of 2560 patterns/chip × 32 chips/LAMB × 4 LAMBs/AM board × 16 AM boards/crate = 5.2 million patterns per crate for option A and 2.6 million patterns for each stage of option B. We show in section 4 that this would be enough for an FTK working at instantaneous luminosities no larger than 1×10^{34}.

We plan to produce a denser chip for luminosities above 10^{34}. There are a number of ways to increase the memory density:

- Since 11-layer matching would require too large a pattern bank, we will build patterns of 8 layers instead of 12. This change will provide an increase of 12/8 = 1.5 in the number of patterns per chip.
- We will use 90 nm technology instead of 180 nm, gaining another factor of 4.
- We will develop a full custom chip that will reduce the size of a pattern, gaining at least a further factor of 2. We have an ongoing R&D project to produce a 90 nm mini-ASIC to test the full custom cell.

- A factor of 2.2 can be gained by enlarging the silicon area that is used. The package can house a 1.5×1.5 cm^2 chip instead of the 1 cm^2 used now.
- A factor of 2 or more can be gained with the use of 3D technology which would extend the existing silicon in the vertical direction; two or more chips could be placed in the same package. This is an option that we are just starting to consider. Fermilab has developed expertise in 3D silicon chip technology.

The expected total gain, taking into account all the contributions listed above, is over 50. A new chip should contain 135k patterns for a total of ~4 million patterns per LAMB and 256 million patterns per crate for option A (half of that for each stage in option B). To succeed in this expansion, we must be mindful of power and cooling considerations. Some gain will come naturally, for example a reduction of the chip power by a factor of 4 when the chip voltage decreases by a factor of 2

There is also a potential to reduce the needed size of the AM bank. We have been using 1 GeV/c as the FTK minimum track transverse momentum. We will assess the change in physics performance as the minimum P_T is increased to 1.5 or 2.0 GeV/c. If a higher value still gives good results, the reduction in the needed AM bank size would be significant since it scales with curvature.

2.3.2.6 The Tree Search Processor

We show in section 3 that at 3×10^{34} $cm^{-2}s^{-1}$ the average number of matched roads coming out of an affordable Associative Memory might be too large to directly download into the Data Organizer. For this reason, we are considering adding a Tree Search Processor (TSP) to reduce the number of fake roads before transmission to the AUX card. If this is needed, which we will know after further study, TSPs could be designed into the new AM chip using 3D techniques, they could be mounted on the LAMB cards, or they could be placed on the AMBoard near the output of the LAMB, whichever is the most cost effective solution. As an example, in this section we consider the latter possibility (see figure 2.21).

We first explain how a general TSP works and then describe the simplified version to be used in FTK. The TSP was proposed long ago as a second stage in pattern recognition [12] [13] following a first stage of associative memory. While the AM operates at the full detector readout rate, the TSP can work at the much slower rate of matched roads coming out of the AM. The task of the TSP is to complete the pattern recognition by narrowing the width of the road from that used in the AM. Call the width of the AM roads generically "fat roads". For each event, the TSP receives fat roads and higher resolution hit data (finer superstrips) and then searches a large bank for patterns matching the hits. The TSP pattern bank, organized for high speed pattern matching, can be stored in high density commercial RAMs, thus reducing size and cost.

The TSP employs a successive approximation strategy, doing pattern matching on the same event with ever increasing spatial resolution. Coarser data resolution is obtained by simply ORing adjacent superstrips. Figure 2.19 demonstrates how the TSP works for the case of straight line tracks passing through a 4-layer tracker. Figure 2.19a shows the eight possible patterns when the fat road in each layer is divided into two smaller superstrips. Pattern 3 matches the track's actual trajectory, and it becomes the "parent pattern" for the next stage in which the width of the superstrips is further reduced by a factor of two. Figure 2.19b shows that there are 4 possible finer resolution patterns, with pattern 3 matching the input track. Since there still is a track candidate, we further reduce the superstrip size by another factor of two. This process is iterated until either we reach the final TSP resolution (success) or we are left with no track candidate (failure).

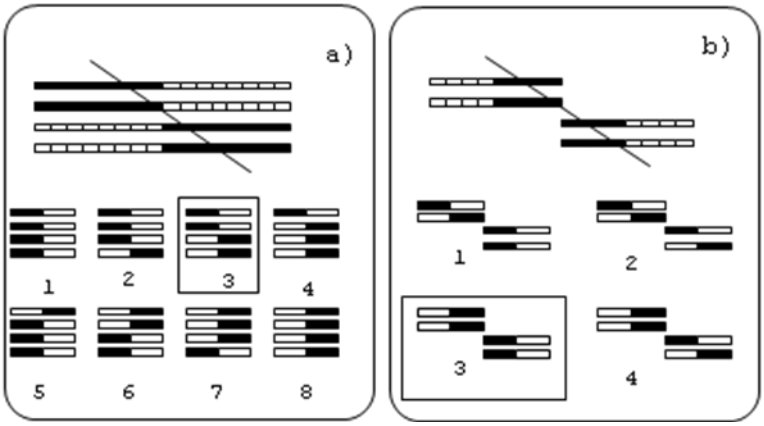

Figure 2.19: (a) Eight patterns are compatible with a straight line when each layer is divided into two bins. Pattern number 3 matches the input track. (b) The width of the superstrip in each layer is further reduced by a factor of two and we test the four possible patterns consistent with the matched pattern in the previous step.

The pattern bank can thus be arranged in a tree structure (figure 2.20A) where increasing depth corresponds to increasing spatial resolution. The tree root corresponds to the incoming fat road. Each node represents one pattern and is linked to its sub-patterns (pattern block) corresponding to a factor of 2 increased superstrip resolution. We scan the pattern block (block test) and every pattern matched by the event data is a track candidate that enables the search at the next level. A refined track candidate (thin road) is found whenever the tree bottom is reached. If the TSP reaches the intrinsic detector resolution, pattern recognition is complete. Any remaining hit ambiguities in the output for a single input fat road are resolved by the track fitter which fits all of the output combinations. The tree search is obviously much faster than a purely sequential search. The average number of patterns one has to examine to find a single track [14] is proportional to the total number of levels in the tree and to the average number of patterns in a pattern block. If there is more than one track in a fat road, the number of nodes followed during the tree search increases faster than linearly [1] since many fake matches can occur at low resolution.

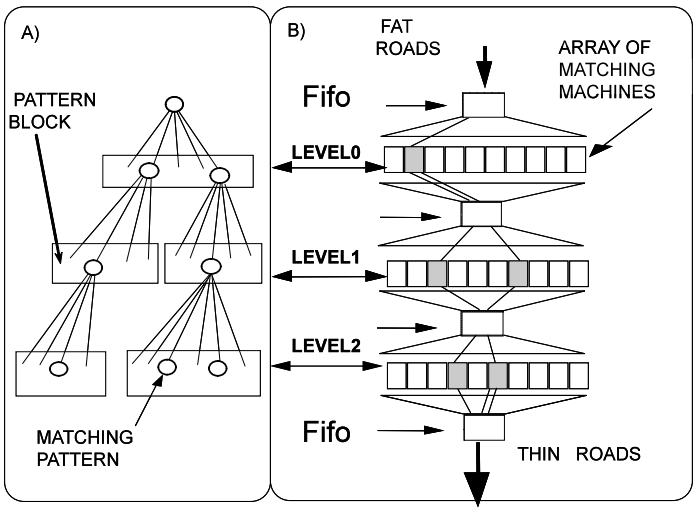

Figure 2.20: (A) Hierarchical pattern organization in a tree structure. (B) Parallel architecture of machines distributed in pipeline stages each corresponding to a tree level.

The tree search algorithm can easily be implemented in a parallel architecture since different patterns can be compared to the event data independently. Moreover, the tree search can be pipelined, since all tests at one level must be executed after the tests in the previous layer are complete. It is possible to reach a high degree of parallelism if we assign several simple machines to the parallel execution of the block tests at each level of the tree. Figure 2.20B shows an architecture where the machines are distributed in pipeline stages separated by derandomizing FIFO memories. For more details, see [15].

For FTK, the situation is simplified since we plan to build a single level TSP. The TSP matching logic will compare its patterns to the event data. We will use a mini-DO containing a mini-HLM using the ideas described in section 2.3.2.2.

Since pattern matching is divided between the AM and the TSP, the event information has to be partitioned. The AMBoard receives superstrips with a factor of 2 finer resolution than needed for the AM. All but the lowest order bit is sent to the AM as the AM-SS, with the remaining bit, the TSP-SS, used by the TSP to compare with patterns that are half as wide as those in the AM. The event data are stored in the mini-HLM while the SSs are being received and then used when matched roads are found by the AM. The mini-HLM is addressed by the AM-SS, and each location has only 2 bits, one for each of the TSP bins into which an AM-SS is split. If a half-SS is hit in the event, that HLM bit is set to 1, otherwise it is set to 0. For each SS that enters, the mini-DO determines whether the hit is in the left or right half of the AM-SS and sets the appropriate bit. This is done with a Read/Modify/Write operation to avoid resetting a half-SS that was set by a previous hit. The size of the HLM is 16 k locations for each layer, for a total of $16 \times 8 \times 2 = 256$ kbits. Even 4 or 8 HLMs can fit inside a XC6LSX150 FPGA which has 4.8 Mbits in block RAM.

A Road Map is also necessary to address the mini-HLM during the Read Mode (see section 2.3.2.2). It fits easily in a DDR3 chip as described in section 2.3.2.2.

The TSP also needs the pattern bank at twice the resolution of the AM boards. But a TSP pattern is extremely compact since for each layer only one bit is needed per silicon layer. This is sufficient since the TSP works within a Fat Road found by the AM and only has to test which half-SS should be hit. The content of the TSP pattern bit is set to 1 or 0 when a hit is required in the left or right half, respectively. There will be 2M patterns in a LAMB and to each we will allocate a packet of 16-bit words in a DDR3 RAM. The packet will store the pattern block (12-24 patterns each of 8 bits) and a block base Road ID to calculate the ID of roads found by the TSP. A packet will be deep enough to store the largest pattern block; in the DDR3 chip there is enough space for even 64-word packets. We will place multiple DDR3 chips around the TSP and HLM FPGAs to allow the immediate processing of each road even if it takes 20 ns or more to fetch an 8-16 word packet. Figure 2.21 shows the AMBoard PCB to show that there is space for 8 FPGAs (2 per LAMB) surrounded by 8 or more DDR3 RAMs (seen in the figure within a yellow box). The figure on the left has all of the signal layers turned on while the one on the right has only the bottom layer visible. The right-hand figure also shows the 2 new FPGAs, one near the P3 connector for the SS FIFOs and the second, where the input FIFOs are in the prototype (figure 2.16), to collect the TSP matched roads and transmit them through the P3 connector. We plan to provide 8 independent road outputs from the AMBoard to the AUX card. Since the AM boards in a crate are no longer in a single common pipeline, the SSs do not have to be sent off of an AM. So we can use the 6 serializers we have on the back of the board (currently used for SSs) plus those we already have for roads to create a very powerful road connection between the AM board and the AUX card. We can provide a transfer rate of 200 MHz or more for each LAMB. The links between the AMBoard and AUX card are shown in figure 2.22.

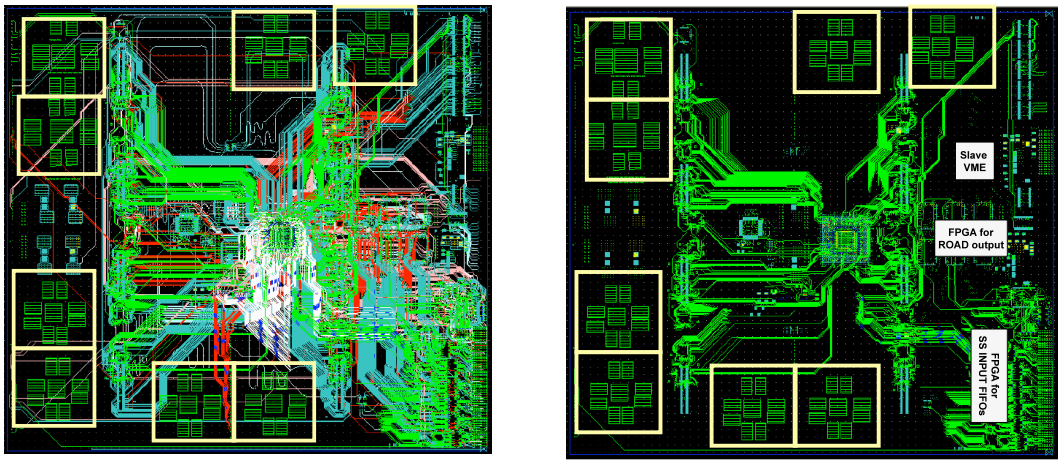

Figure 2.21: The AMBoard PCB with all signal layers turned on (left) or the bottom layer only (right).

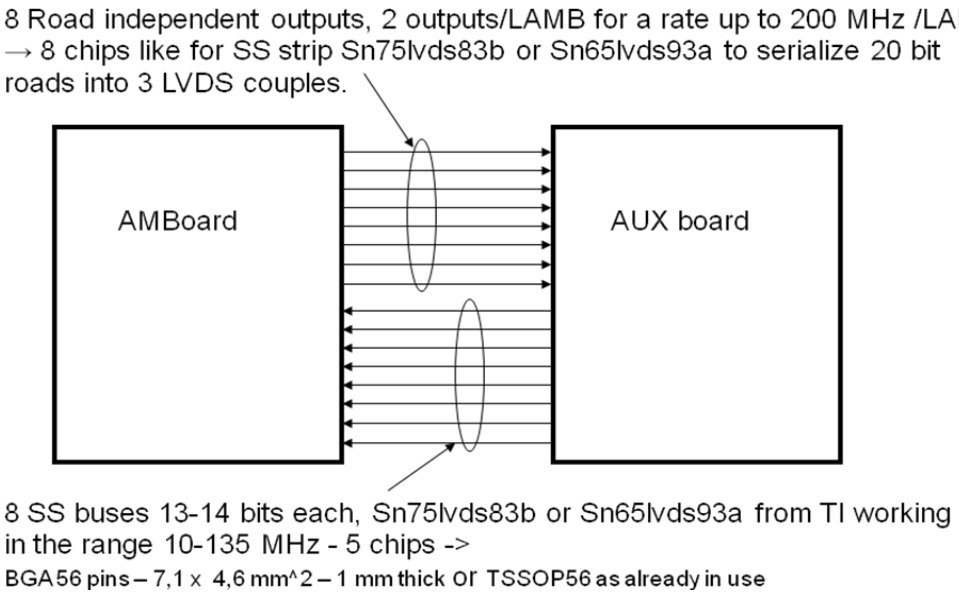

Figure 2.22: Data connections between the AMBoard and its AUX card.

2.3.2.7 The Track Fitter

A road is narrow enough that a helical fit can be replaced by a simple linear calculation. Each of the 5 helix parameters (p_i) can be calculated as the vector product of prestored constants (a_{ij}) and the hit coordinates (x_j): $p_i = a_{i0} + \sum_{j=1}^{N} a_{ij} x_j$ where N is the number of coordinates on the track, one for each SCT layer and two for each pixel layer. Since there are more than 5 coordinates, there are additional linear equations that correspond to constraint equations: $f_i = b_{i0} + \sum_{j=1}^{N} b_{ij} x_j$, again where the constants are prestored. There are $(N - 5)$ such equations. Each f_i would be zero for perfect measurement; the sum of the squares of the f_i's serves as the goodness of fit.

This linear approximation gives near offline resolution for regions considerably wider than a single road. We use a single set of constants for each sector of the detector (see section 3.1.2 for the definition of sector). The width of the sector at each silicon layer is the size of a physical detector module. Per sector, $5\times(N + 1)$ constants are needed for the helix parameters, and $(N - 5)\times(N + 1)$ constants are needed for the constraint equations. The total number of fit constants (FC) per sector is thus $N\times(N + 1)$.

For option A, $N = 10$ (6 from the three pixel layers and 1 each from the four axial SCT layers). This is the more difficult case since for option B, $N = 8$. Here we discuss option A, since the hardware for it will be more than enough for option B. We need 110 FCs, half for the χ^2 calculation and half for the helix parameter calculation. The number of sectors per core crate for option A is approximately 100 k, and they can be partitioned among the 64 Track Fitters (one per

LAMB in the crate). If we make the conservative assumption of 16 bits per constant, we need a total of 16 bits × 110 constants × 100 k sectors/64 TFs = 2.7 M bits for storing constants in a TF.

It is important to note that only roads with hits on every layer can be processed this way with the given FC set. If this requirement were imposed, then any single-layer inefficiency would produce an inefficiency in the tracking algorithm. A practical solution to this problem is to allow roads with missing hits, using $N-1$, $N-2$, etc. coordinates. However each such case would require separate sets of constants since the FCs depend on the number of coordinates and the identity of the missing layers. In this approach, the total FC storage would become an issue. For option A, if we allowed missing the hit on one layer, the 2.7 Mbits would have to be increased to include 3 additional FC sets with N=8 (missing a hit on one of the pixel layers) and 4 FCs with N=9 (missing a hit on one of the axial SCT layers). The total FC memory required per TF would thus be 17.1 Mbits. This would fit in an expensive and large Virtex-6 (at least the XC6VLX550T) or it could be placed in a static RAM on the back of the mezzanine (4M×18-bit Samsumg K7R641882M).

In order to reduce the total memory requirement, the FTK simulation currently uses the nominal set of constants (when hits are present on all layers) even when there are $N'<10$ measured coordinates. The procedure, which exploits the strong correlation among hits for real trajectories, is referred to as "guessing hits". Once the missing coordinates are guessed, the incomplete x_j vector is filled out and the nominal constant set is used. A good guess for the missing hits is to position them where they would minimize the χ^2, leaving of course the measured points unaltered. This essentially corresponds to placing the missing hits at the intersection of the N'-hit fit track and the inefficient layers. Let $M = N - N'$ be the number of missing hits. By storing some additional constants per sector, we can calculate the missing hits and avoid increasing the entire FC set by a factor of 6. The largest number of constants that will have to be added per sector is $N(N+1)/2=265$ (used by both pixel and SCT extrapolations). To minimize the number of arithmetic operations needed to calculate the missing hit, we will also store an additional 13 constants, 1 for each SCT layer and 3 for each pixel layer. When the 78 constants for tracks with a missing hit are added to the original 110, we get a total of 4.7 Mbits per TF. This fits in the smallest Virtex-6, which is housed in a 23×23 mm^2 package and also contains 288 DSPs for fitting.

To fit one track, the following number of vector products will be executed:

- 5 to calculate the 5 helix parameters (10 terms each)
- 5 to calculate the χ^2 (10 terms each)
- 2 for calculating a missing pixel hit or 1 for a missing SCT hit. To use these results, 3 additional multiplications and 2 additions are needed.

We will allocate 15 different DSPs to perform all of these calculations, working independently and in parallel. Moreover, since the vector products often have 10 terms and thus each DSP will need 10 clock cycles, we will increase the number of DSPs by a factor of 10 so that ten tracks are

fit in parallel (a total of 150 DSPs which as seen above are contained in a single FPGA), producing a fit result each clock cycle.

A next generation track fitter, the GigaFitter, which performs more than a fit per nanosecond within a single FPGA, was recently developed for CDF [19]. The core of the GigaFitter is implemented in a modern Xilinx Virtex-5 FPGA. The XCVSX95T contains several MB of memory and 640 dedicated arithmetic units (DSPs) each with one 18/25-bit multiplier and several adders. The clock runs up to 500 MHz, but it is run at 120 MHz in the GigaFitter. Figure 2.23 shows the main logical blocks in the GigaFitter.

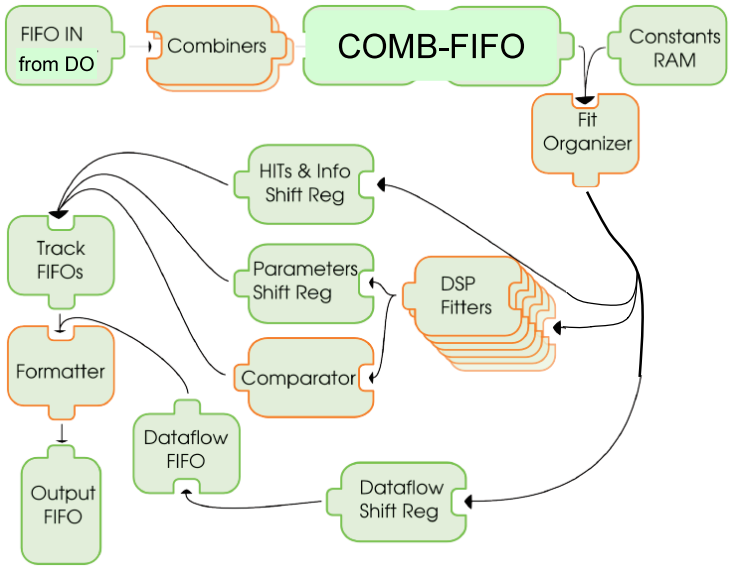

Figure 2.23: The logical blocks within the GigaFitter track fitter.

The Track Fitter is naturally divided into six parts: the Combiners, the Fit Organizer, the Serializers, the DSP Fitters, the Comparator, and the Formatter. Large RAMs are used to store the FCs, FIFOs are used to interconnect the various stages of the pipeline, and shift registers synchronize the data. When there is more than one hit per SS in any layer, the Combiner produces all of the hit combinations to be fit. The Fit Organizer coordinates the fetching of hit combinations and fit constants, and starting the Serializers. The DSP Fitter performs the fit. The Comparator judges the fit results and selects the best choice in the case of multiple fits of the same track. The Formatter forms the output data packets.

The Combiner has two sequential steps. First it pops road packets (road ID and hit list) from the Input FIFO and stores them inside small RAMs implemented in the FPGA distributed memory, one for each layer. Counters keep track of the number of hits in each layer. After all hits have been loaded, the Combiner starts the second step, processing the road. A road can have more than one hit per layer, any of which could belong to the track. Thus the Combiner forms track candidates from all of the combinations of one hit per layer. Using the counter information to

generate RAM addresses, it fetches in parallel one hit per layer to create one combination per clock cycle until all of the combinations are fetched. There are two independent Combiners working in parallel, each with its own set of RAMs. While one is processing one road, the other can pop and load hits from the following road from the Input FIFO. Both Combiners work continuously to produce a constant flux of combinations which are stored in a large FIFO called the Combination FIFO.

The Fitter Organizer pops combinations out of the Combination FIFO and completes them with the FCs extracted from a large RAM. The RAM, implemented in the FPGA's memory blocks (BlockRAM), has space for many sectors. The set of constants retrieved is for the sector containing the current road. The entire set of hits and associated constants are extracted in parallel in a single clock cycle by the Fitter Organizer. It then serializes them, associating each hit with the corresponding constant for the vector multiplication, and sends one hit-constant pair each clock cycle to its own dedicated DSP Fitter.

The DSP Fitter receives the hits and constants and then calculates the track parameters and the goodness of fit parameters (χ^2 components). The 10 vector products are executed in parallel by exploiting the large number of on-chip DSPs. The vector products are performed configuring the DSP as MACC (multiply and accumulate) and serially processing the hits. The products in a 6-term vector product (in the CDF GigaFitter) are calculated and accumulated sequentially in 10 clock cycles. In a DSP Fitter, there are 10 DSPs, each computing one helix parameter or fit parameter. Thus with 10 DSP Fitters, using a total of 100 DSPs, the GigaFitter is able to process on average one combination every clock cycle.

Once the results are ready, the χ^2 components are sent to the Comparator while the helix parameters and the hits used in the fit are stored in shift registers waiting for the Comparator decision. Additional information provided by the Combiner at the very beginning, but not used in the fit, is maintained in shift registers and provided to the Comparator when needed. This is particularly important when different fits of the same track have to be judged to choose the best one. The GF can perform many fits on a single track that has hits on all layers: first a fit using all the hits, and then fits in which the hit from one layer at a time is removed. The Comparator chooses the best one.

The Comparator calculates the final χ^2 using a DSP in MACC mode. Three clock cycles are necessary for each track and there are three such units to sustain the output rate of one track per clock cycle. The result is compared with a threshold set via VME. If the track passes the χ^2 cut, the χ^2 and the track quality (a function of the layers used and the quality of each hit such as the cluster width) are used to compute the *g* (goodness) factor which is compared to the *g* of the best track in the sequence. If the new fit is better, the registers that store the best track and its parameters are updated. Once the sequence is finished, if there is at least one track passing the χ^2 cut, the best track is stored in the Track FIFOs.

Finally the Formatter reads the parameters and the χ^2 of the accepted tracks from the Track FIFOs and merges this information with the hits, road identifier, and some status bits, pushing them to the output.

2.3.2.8 The Hit Warrior

The Hit Warrior (HW) receives the hit coordinates for each track with a good χ^2. Its task is the deletion of duplicate tracks. Our strategy is to start with loose criteria after the first stage (7-layer option A fit or 8-layer option B fit) when the track information is incomplete so that we don't make the wrong track selection and thereby lose efficiency. At this stage, the HW will only delete duplicate tracks within a single road. After stage two, when the full 11-layer fit is carried out, more stringent requirements are applied.

The Hit Warrior employs associative memory, here implemented in an FPGA rather than in an ASIC. The HW stores the hits on tracks in a small AM built on the fly when tracks are sent to the output. After the first stage fitting, it will be able to store up to 100 tracks, a large number considering that it will act only on tracks within a single road. A track in the AM consists of a row of N locations, each capable of storing a full resolution hit in a specific silicon layer. There will be 10 words for option A (4 SCT layer hits and both coordinates of the 3 pixel layer hits) or 8 words for option B (8 SCT layers).

When a new track is received, it is stored in a temporary location. That track is compared in parallel with all of the previously stored tracks. The comparison employs majority logic; a track is declared a duplicate if at least *k* out of the 8 or 10 hit words match one of stored tracks. The parameter *k* will be tuned to maintain high efficiency. In case of a match, the incoming track is not sent to the output. But whether the track is matched or not, it is inserted into the AM for comparison with later tracks. When all the track candidates for that road have been processed, the AM is cleared.

Since full resolution hit coordinates here are in local coordinates, 10 bits are sufficient for option A or 8 bits for option B. Thus our assignment of 12-bit words is conservative.

The HW logic can be implemented in an inexpensive Spartan-6 FPGA which has a distributed logic or "slice" architecture. Each slice contains 4 logic-function generators or look-up tables (LUT) and 8 storage elements. In particular, the SLICEMs contain features important for the HW: an arithmetic carry structure that can be propagated vertically up through the slice column, and LUTs for use as 64-bit distributed R/W RAMs. It is with 6-input R/W RAMs that we build the AM locations into which the tracks are written as they pass through the HW. A new incoming track will address all the previously loaded RAMs in parallel to compare the track with all of them in a single clock cycle. The figure 2.24A shows how the comparison is made for one of the AM locations. The high-speed carry logic (highlighted lines and MUX multiplexers in the figure) is used for rapid wide pattern decoding (in our case only 2 RAMs). The carry chain, initialized to a high level by Init, is propagated only if the output of the RAM is high. If the

RAM output is low, the MUX outputs a low level. A new incoming track is stored in a temporary location, one hit word per detector layer. Each RAM monitors a 6-bit field of the track and outputs a high level only if a match is found. Matches of different 6-bit fields are logically ANDed by the carry chain. If all hit bits match, the Layer Match flip-flop (LM) is set.

For option A (B), an entire pattern requires 20 (16) RAMs which fit in 10 (8) SLICEMs. For 100 tracks, we would need up to 1000 SLICEMs, which is small compared to the 5000 available in the top of the line Spartan chip. An additional slice, in this case a SLICEX, is needed to count the number of matching layers (see figure 2.24B) and declare the track a duplicate if that number exceeds the threshold. More than one AM section can be implemented in a chip, or multiple chips can be used in parallel, to increase the HW computing power and to allow the time needed to reset all of the small RAMs in one AM section (64 clock cycles) at the end of processing a road.

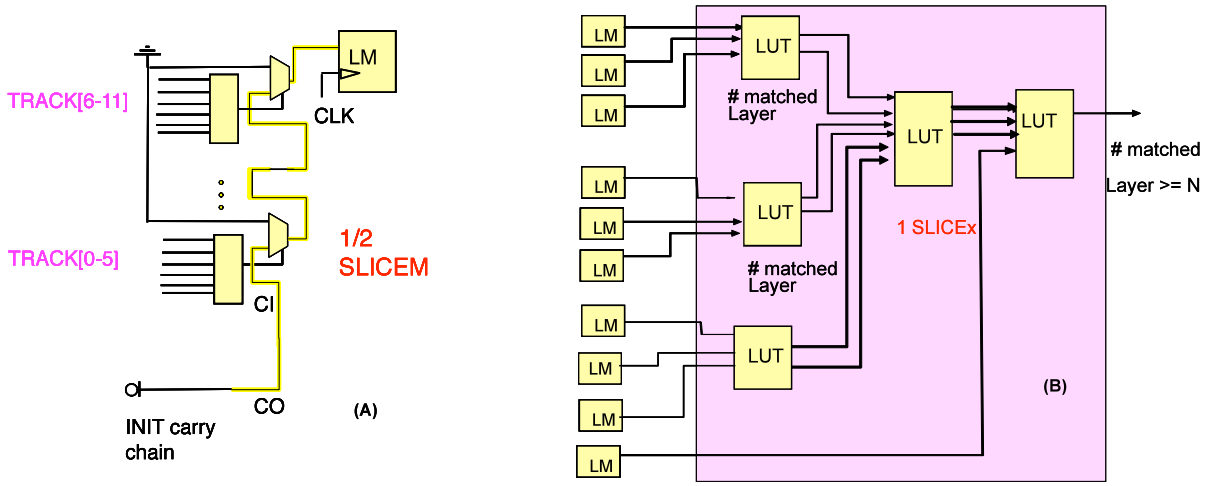

Figure 2.24: (A) The comparison logic for one AM location in the Hit Warrior. (B) The HW logic for determining an overall track match.

2.3.3 The links to the second stage

The first stage fitting and HW cleanup greatly reduce the quantity of data per event (see section 3.3). The input superstrip rate into an AMBoard is 8000 per event (8 buses each carrying an average of 1000 per event). The average rate out is

- Option A: 2000 roads per event per LAMB before track fitting. The number of good tracks after the TF is approximately 150, which drops to about 12 after the HW (working only within a road). For each track, we transfer to the second stage all of the hits used in the fit, the 5 helix parameters, and (for debugging purposes) the road ID to which the track belongs. Thus there are 16 words per track or 400 words per event. Each AM Board AUX card will have a direct link to one of the four 11-Layer Fit Boards (see figure 2.4). The data rate on that link will be 40 MHz for a 100 kHz level-1 trigger rate.

- Option B: As shown in section 3.3.2, the number of tracks to be passed from each first stage AM board to the corresponding second stage AM board is approximately 250. For each track, we pass the 8 hits in local coordinates plus the pseudolayer superstrip for the second stage road finding (encoded with course values of the curvature, azimuth, and polar angle from the first fit). This is 5 words per track or 1250 words per event which can be sent on cables connecting the AM boards.

2.3.4 The Second Stage

The second stage is needed to reduce fake tracks, which occur at too high a rate at high luminosity. It also improves helix parameter resolution, especially $\cot\theta$ for option A and the impact parameter and z_0 for option B.

2.3.4.1 11-Layer Fit Board for option A

Each 11-Layer Fit Board receives the output from 4 AMBoards and the stereo SCT hits for the 2 η-ϕ towers associated with those AMBoards. The SCT stereo data are transmitted using LVDS or the optical links described in section 2.3.2.1. For LVDS, there would be 2 links per layer, one for each η-ϕ tower, for a total of 8 pairs, which fit in a single cable.

A functional sketch of the 11-Layer Fit Board is shown in figure 2.25, with the light blue box expanded in figure 2.26.

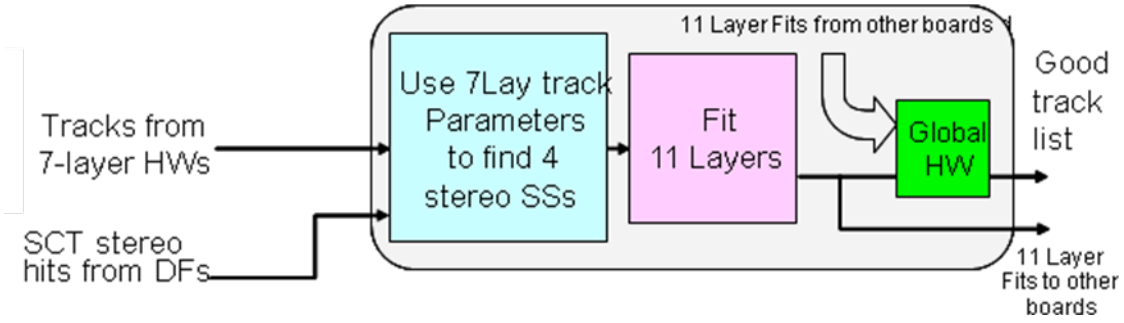

Figure 2.25: Functional overview of the 11-Layer Fit Board

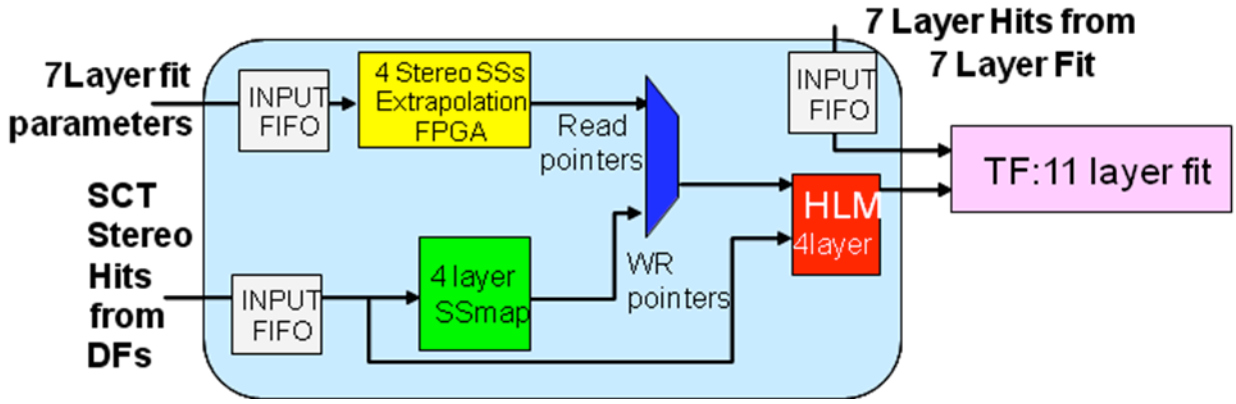

Figure 2.26: The extrapolation section of the 11-Layer Fit Board

The stereo SCT hits received by the DFs are organized in a local HLM for 4 layers. This is part of the Data Organizer function described in section 2.3.2.2. The correlated SSs have to be fetched when a 7-layer fit is received from the first stage. For this purpose we use the results of the 7-layer fit itself. We invert the fit calculation and use the 7-layer helix parameters to estimate the coordinate at each of the SCT stereo layers. These estimates are quite precise since the stereo modules are glued directly below the corresponding axial module. The 7-layer fit extrapolation is in fact able to determine the stereo hit location with a resolution of 3 silicon strips as shown in figure 2.27. The SS size for these 4 layers is accordingly chosen to be very narrow, 4 strips wide. For each 7-layer track, we fetch three contiguous SSs, the one found by the extrapolation and one on either side. The hits found in these SSs are combined with the 7-layer fit hits to produce the full 11-layer set of hits to be fit in the TF. Tracks with a good χ^2 in the 11-layer fit are sent to the Global HW which performs the final duplicate deletion in a core crate, taking into account the good tracks found by the other 3 11-Layer Fit Boards in the crate. The Global HW works on the same principles described for the HW in the AMBoard AUX card with a few important differences. The hit coordinates are not local, so 24 bits will be allocated for each. Correspondingly, the number of LUTs shown in figure 2.24A will be 4 instead of 2 because of the larger hit word size. Also the number of locations for tracks has to be larger than 100 (see section 3.3). But we can use a multi-FPGA system to implement the Global HW since the 11-Layer Fit Board will have plenty of space.

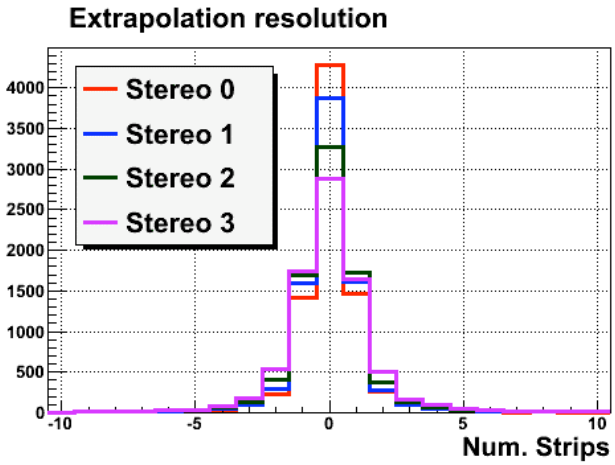

Figure 2.27: The resolution of the option B extrapolation into the stereo SCT layers in units of the strip width.

2.3.4.2 Interface for option B

A shown in section 3.3.2, there are a total of 60 tracks per event coming out of the second stage of each core crate. These tracks go through a final HW and then off of the crate.

2.3.5 FTK ROD

We will base our design on existing ATLAS RODs, but customize it to transmit data in packets that are best suited for the ROSs, given the need to read out tracks at the full 100 kHz level-1 trigger rate. As noted in section 2.2.2.2, we are pursuing two possibilities. Once that choice is made, design of the ROD will follow.

3 Performance at the SLHC Phase I luminosity of 3×10^{34}

3.1 The FTK simulation

3.1.1 General Structure

The simulation program for FTK (FTKSim) processes complete ATLAS events and creates exactly the same list of tracks that will be produced by the FTK hardware. The program serves a number of purposes:

- detailed and reliable evaluation of FTK physics performance by processing complete events produced by the full ATLAS detector simulation,
- evaluation of crucial parameters needed for the hardware design,
- determination of the large set of constants needed for programming FTK, and

- determination of tracking performance parameters for use in fast parametric detector simulation for high statistics studies of physics performance.

These goals are achieved with an intermediate-level simulation that describes the FTK algorithm and internal data accurately, but avoids detailed bit-level simulation of the hardware in order to attain sufficient speed for simulating moderate sized samples of complete events. The core code is based on a similar simulator created for the CDF SVT [20] [21] but with data structures appropriate for ATLAS Athena [22] [23].

FTKSim is a package containing a few standalone programs, written in C/C++, which interfaces with ROOT [24] and, with moderate changes, can be integrated into the Athena framework in the future. Figure 3.1 shows the connection between FTKSim and the Athena package.

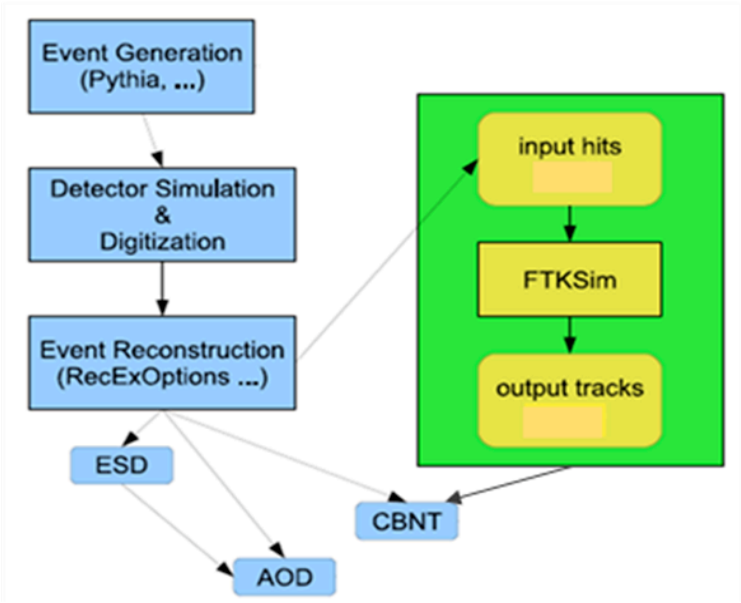

Figure 3.1: Functional sketch showing how FTKSim (green box) gets the silicon hit data from Athena.

FTKSim contains two main processes that simulate the Associative Memory and Track Fitters respectively. To reconstruct an event, FTKSim carries out two steps:

- road finding: all matched roads are found by comparing the event's SCT and pixel hits with the prestored bank of patterns, thus simulating the AM;

- track fitting: all matched roads are processed to find the best track helix parameters and to reject fake tracks. Within each road the χ^2 cut and Hit Warrior function are applied, and the helix parameters are calculated using the linear approximation with constants obtained from a principal component analysis [25]. The precision of the linear approximation method as described in section 2.3.2.7 was checked under many conditions [26] and found to be comparable to the full offline resolution. It has the important advantage of being very fast.

FTKSim uses data that are produced by two other programs, PattGen and CorrGen. They generate the two main data banks needed for FTK simulation: the pattern bank (PB) for the AM (PattGen), and the fit constants (FC) for the TF (CorrGen). These programs run on special training samples, most often single muon events. Figure 3.2 shows the connections among PattGen, CorrGen, and FTKSim.

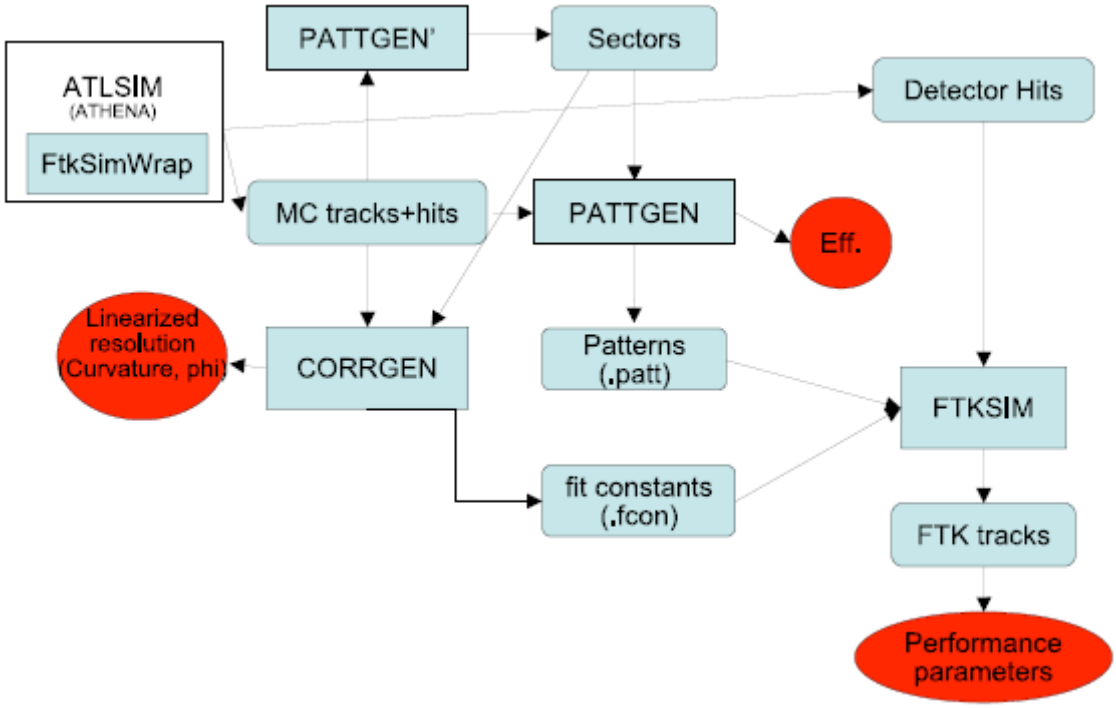

Figure 3.2: The functions and connections of the three programs used in FTK simulation: FTKSim, PattGen, and CorrGen.

3.1.2 Generation of internal FTK data banks

The internal FTK data banks encode information about the geometry and resolution of the silicon detectors as well as the appropriate subdivision of the detector into regions (one per core crate). The geometry and resolution information is extracted from the simulated training samples rather than from a geometric description of the detector. These large samples of single muon events are produced using the full ATLAS simulation (see section 3.2)[1].

[1] When FTK is operational, the training tracks will come from either full simulation using the latest inner detector geometry and beam location or from real data tracks. Because the training occurs for tracks with impact parameter up to 2 mm, ensuring good efficiency for *b* daughters, our patterns and fitting constants are insensitive to small changes in the beam position. In CDF, for example, new constants are needed if the beam moves by more than 0.5 mm in the transverse plane; new patterns are needed when the beam moves by more than 0.3 mm. Since the Tevatron beam position changes slowly with time, for example due to the motion of a quadrupole, we produce new banks well in advance of when they are needed.

In principle, the pattern bank should be 100% efficient, with a road to match any track that could pass through the detector. Such trajectories would include the effects of multiple scattering, detector resolution, strong interaction with detector materials, etc. If rare, large angle scattering is included, the size of the pattern bank becomes extremely large. Consequently we accept some pattern recognition inefficiency and generate the banks by including multiple scattering and detector resolution, but turning off strong interactions and delta-ray production.

We define 11 logical silicon layers, each consisting of a barrel layer and disks to cover the full range of track rapidity. We configure FTK to reconstruct any track leaving at least *M*-1 hits out of the *M* layers being used (for example the 7 layers of option A). A combination of *M* physical silicon modules (one per layer) that can be crossed by a single track is called a sector (see figure 3.3).

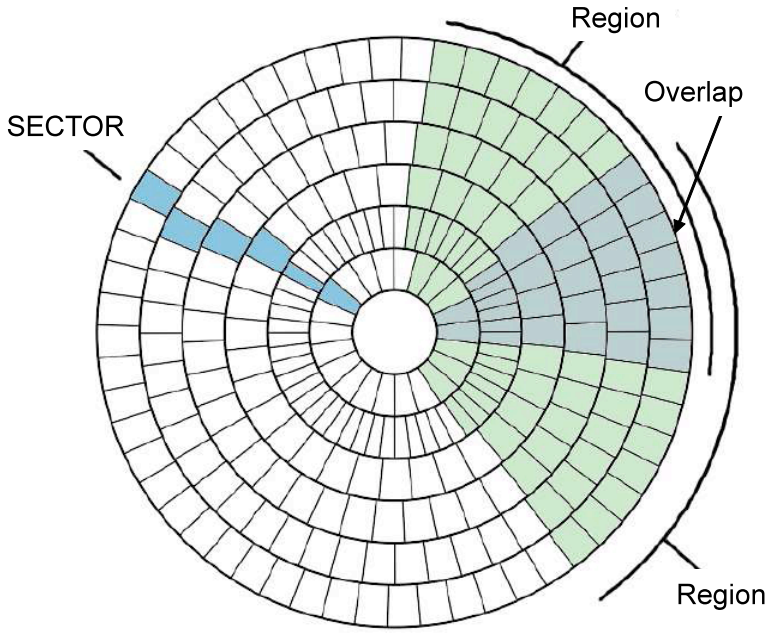

Figure 3.3: A simplified cross section of a silicon detector showing a sector, two regions, and a (large) region overlap.

The first step in generating the data banks is to create a list of valid sectors. It is determined from the large training sample by selecting the sectors that are hit by enough tracks to calculate the constants needed for the fitting stage, typically 15. An important measure for both the sector list and pattern bank is the coverage. This is a purely geometric quantity, defined as the probability that a track (with helix parameters within the desired range) intersects at least (M-1) of the *M* silicon layers within a sector/pattern in the bank. In short, it is the fraction of reconstructable tracks based only on the detector geometry. We first find a list of sectors and measure its coverage. The sectors are then grouped to formed regions, each covered by a single FTK core crate. Since each region contains separate AM banks, patterns crossing a region

boundary are not allowed by the system. This has no impact on the overall efficiency because regions are defined with sufficient overlap for tracks of $P_T \geq 1\text{GeV/c}$: we have 8 regions each covering 45° in azimuth plus an additional 10° of overlap. Patterns located in the overlap region are not duplicated, but are arbitrarily assigned to one of the regions. The regions extend along *z* for the full detector length. Figure 3.3 shows the definition of sector, region, and region overlap.

In the second step, the pattern bank is generated by finding the valid patterns in each sector. Each module is subdivided into bins (superstrips) of equal size, each a rectangle in ϕ-*z* space. A pattern is a combination of *M* superstrips, one in each of the modules in the sector. They are generated by the same algorithm used to find valid sectors: the set of training tracks is scanned and patterns hit by a track are kept. To maximize the efficiency of the pattern bank, we order the pattern list by the number of training tracks that pass through a pattern. We then fill up the available AM pattern locations using the most frequently hit patterns. The bank generation process is slow, but it is carried out only once.

The number of generated patterns depends on the helix parameter range of the training track sample. We have selected ranges appropriate to triggering on electrons, muons, taus, and heavy quark jets. Azimuth and rapidity are generated flat over the entire silicon detector, while z_0 is a Gaussian with a sigma of 8.7 cm. Curvature is generated flat for tracks of $P_T \geq 1\text{GeV/c}$ (chosen for *e* and μ isolation as well as τ and *b* daughters), and the impact parameter is flat out to 2 mm in order to be efficient for *b*-quarks.

The size of the pattern bank also strongly depends on the width of the superstrips. It is chosen as a balance between pattern bank efficiency (for a fixed AM size), the number of matched roads per event, and the number of hit combinations that have to be fit per event. This is discussed in section 3.3.

Finally, fit constants are determined for each sector by inverting the linear matrix connecting a track's hit coordinates to the particle's original helix parameters. We have verified that each sector covers a spatial region small enough to make the linear approximation accurate within the entire sector, although recent work shows that a smaller pixel B-layer sector width further improves the impact parameter resolution compared to that shown in figure 3.11b.

3.1.3 Size and coverage of the pattern banks for options A and B

The size of the pattern bank is a critical parameter for FTK. Cost provides a natural limitation, but there are other pressures that drive the selection. Narrower roads contain fewer pile-up hits at high luminosity, thus limiting the rapid growth in the number of fits due to the combinatorics in selecting one hit per layer for each fit. However with the two-stage architecture we have adopted, the TF execution time is no longer a serious system limitation. Rather it is the data flow in the AM-DO due to the number of matched roads. At high luminosity, these are dominated not by real tracks but by fakes, which grow roughly linearly with bank size.

For the system to work with a 100 kHz level-1 trigger rate at 3×10^{34} $cm^{-2}s^{-1}$, we have to control the number of fake matched roads. Reducing the road width certainly helps. However we have also discovered that once the road size is selected, it is best to have the smallest pattern bank that provides the needed physics performance. Figure 3.4 shows a typical curve of track efficiency vs. bank size. There is an initial rapid rise as the bank is filled with patterns that tracks commonly match. Beyond that, the curve rises slowly as less probable patterns are added from the tails of the multiple scattering distribution. Although the efficiency for real tracks grows slowly in this region, the number of fake matched roads rises nearly linearly with the bank size. Thus we have to choose a size that balances the need for good efficiency with the need to limit the rate of fake matched roads. To accomplish this, we base our choice not on single track efficiency, but on the physics performance for *b*-tagging, τ identification, and single lepton triggering.

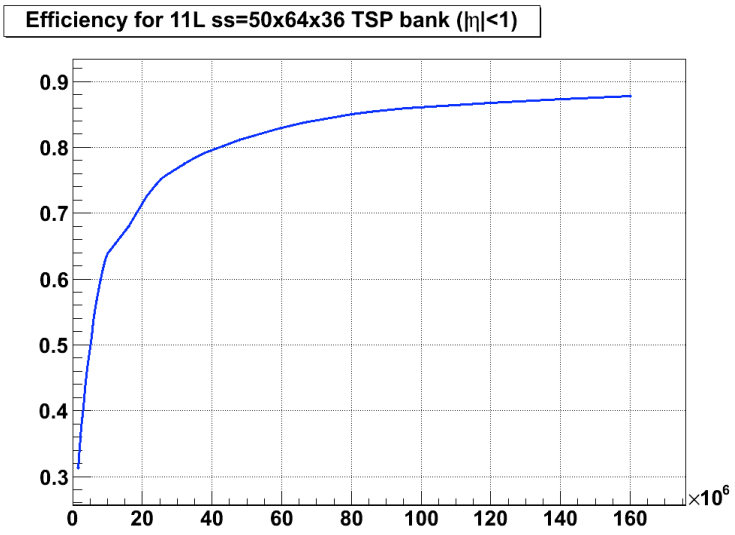

Figure 3.4: A typical plot of track efficiency as a function of the number of patterns in the AM.

In sections 3.3 and 3.9, where we present the data flow and timing of the proposed architectures, we use banks specific to options A and B. However in the physics performance sections (3.4 through 3.7), we have used the single-stage architecture in which all 11 layers are fit at once because we had to freeze the conditions before starting the time-consuming processing of all of the 3×10^{34} data samples. In section 3.5 we show that option B has physics performance even better than that obtained with the 11-layer simulation.

The coverage and efficiency for the 11-layer pattern bank used in the physics performance studies are shown in figure 3.5 as a function of the bank size, on the left for the barrel region and on the right for the full η coverage. We have not yet optimized the logical layer assignments for the transition region between the barrel and the forward regions, and consequently it has a somewhat reduced efficiency. This optimization will be done in the future. The coverage, which describes the geometric efficiency of a bank, is defined as the fraction of tracks that match a

pattern in the bank. A match requires that there be no more than one layer without a hit. The efficiency includes all of reconstruction effects such as the limited pattern bank size, the χ^2 cut after fitting, the application of the Hit Warrior, and the criteria for matching a reconstructed track with a truth track. Both coverage and efficiency are presented for the single muon sample without pile-up to show the intrinsic properties of the banks.

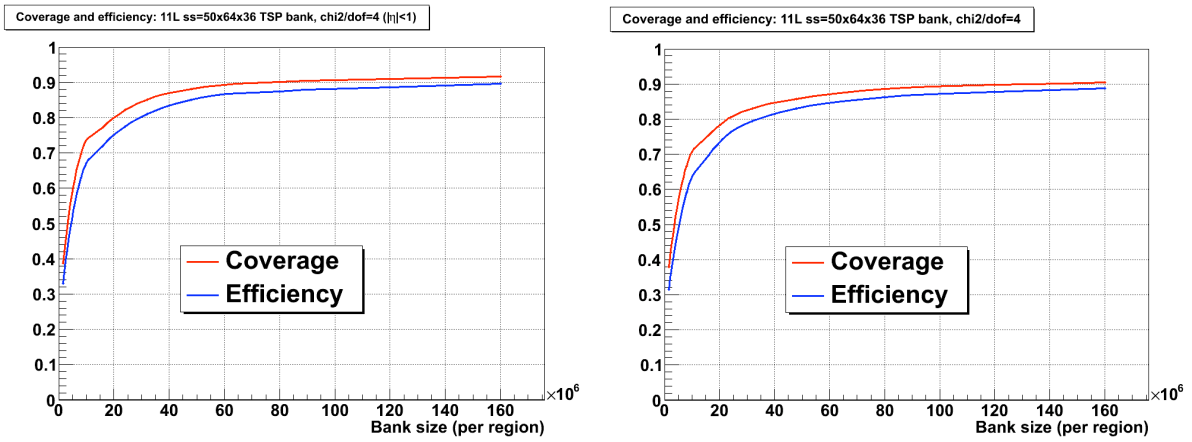

Figure 3.5: The coverage and efficiency for the 11-layer pattern bank as a function of bank size for (left) the barrel and (right) the full rapidity region.

For the 11-layer configuration, the SS width must be kept large to maintain a manageable AM size. Consequently the numbers of matched roads and track candidates are too large for FTK to handle at the full level-1 trigger rate at high luminosity. None the less, it provides excellent tracking performance and thus a good benchmark for the two-stage architectures we are studying.

3.1.3.1 Option A

We have two pattern banks, one that is downloaded into the AM and the other much larger bank that is used in the TSP. For the AM, the SS sizes are 16 strips in the r-ϕ SCT layers and 22 (36) pixels in the ϕ (z) direction in the pixel layers. Figure 3.6 shows the coverage and efficiency for the AM pattern bank. The working point is 128M patterns per core crate, noted by the arrows in the figure.

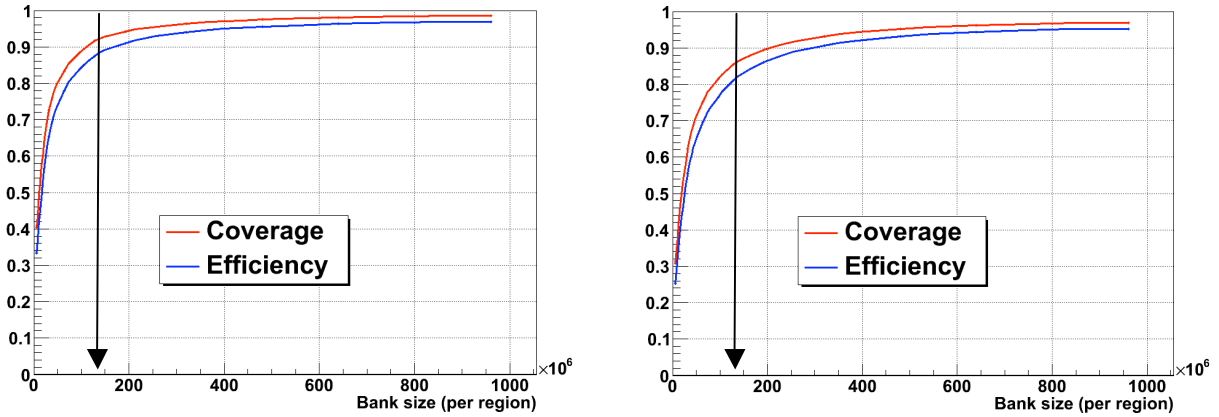

Figure 3.6: The coverage and efficiency of the AM bank for option A in (left) the barrel and (right) the full rapidity region. The arrows show the chosen operating point.

For the TSP pattern bank, the SS sizes are 8 strips in the r-ϕ SCT layers and 11 (36) pixels in the ϕ (z) direction in the pixel layers. The coverage and efficiency as well as the working point of 1280M patterns are shown in figure 3.7.

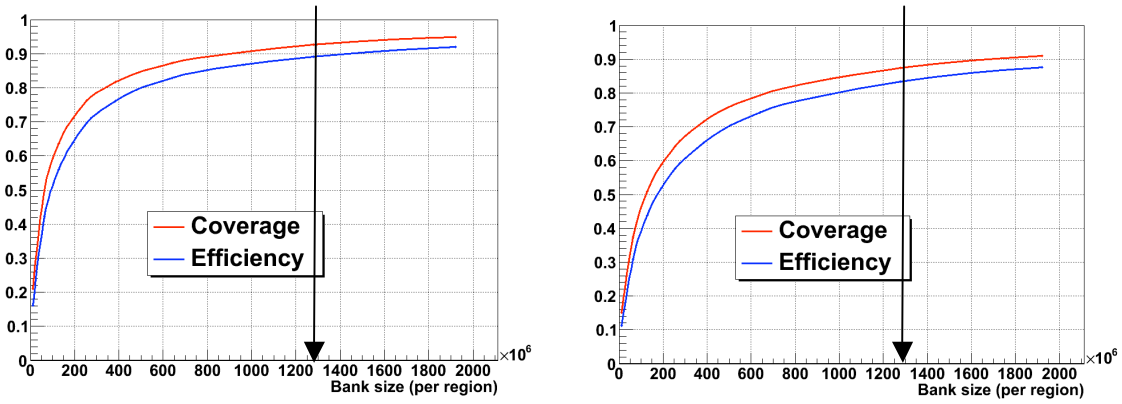

Figure 3.7: The coverage and efficiency of the TSP bank for option A in (left) the barrel and (right) the full rapidity region. The arrows show the chosen operating point.

3.1.3.2 Option B

For option B, at present we use AM banks for both the first and second stages, but not TSP banks. Their purpose is to reduce the number of matched roads and thus improve data flow. For the timing studies presented in section 3.9, we use a factor 2 for the TSP reduction in the number of matched roads for each of the stages. This was obtained from the TSP reduction for superstrip sizes twice what we are currently using. In the near future we will produce the large TSP banks for the current SS size and obtain improved reduction factors.

The SS width for the first stage is 16 strips in the SCT. In the second stage, the pixel SS widths are 20 in the ϕ direction and 25 in the z direction. The sizes of the pattern banks per core crate

are 64M for the first stage and 23M for the second stage. Figure 3.8 (3.9) shows the AM coverage and efficiency for first (second) stage.

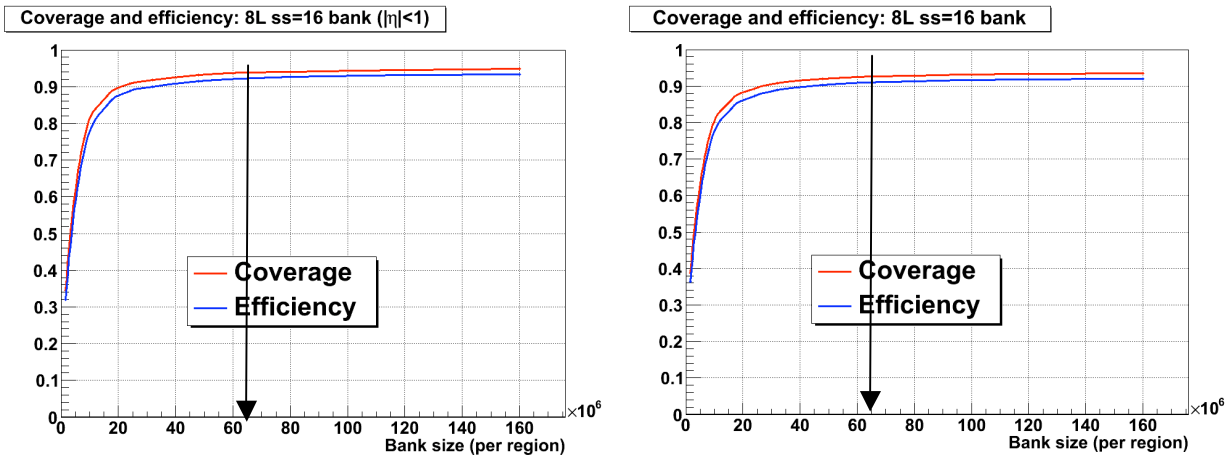

Figure 3.8: The coverage and efficiency for the AM bank in the first stage of option B in (left) the barrel and (right) the full rapidity region. The arrows show the chosen operating point.

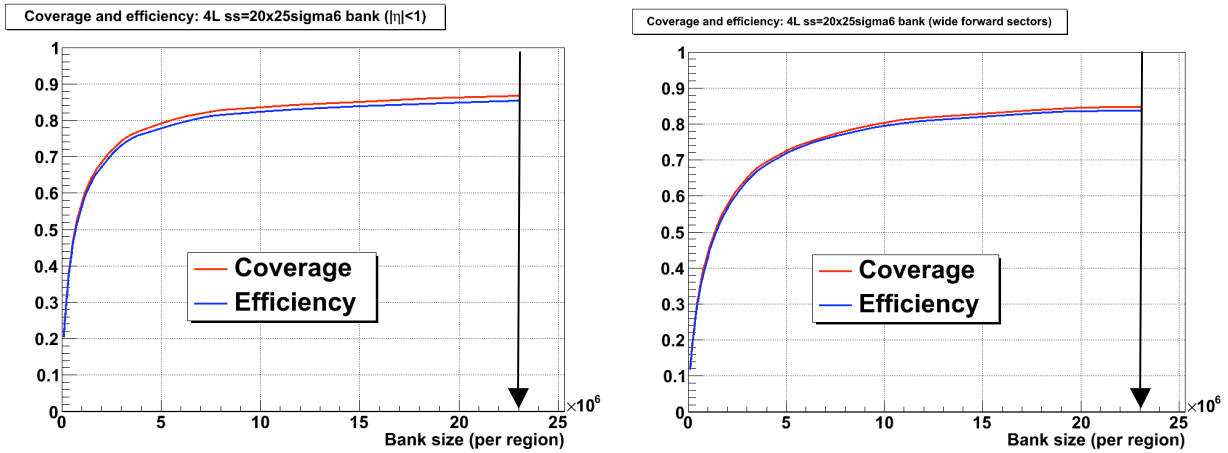

Figure 3.9: The coverage and efficiency for the AM bank in the second stage of option B in (left) the barrel and (right) the full rapidity region. The arrows show the chosen operating point. The efficiency includes that of the first stage.

3.2 Data sets

The simulated datasets used for performance studies and pattern bank generation are listed in table 3.1. They were produced using code and settings derived from MC08, namely Athena version 14.2.25.10 with the ATLAS-GEO-02-01-00 geometry tag and the standard job transforms, with the following differences:

- The beam energy was increased from 10 TeV to the 14 TeV design energy.

- The beamspot offset was set to zero during both simulation and reconstruction (overriding the default mean beam position of approximately 3 mm away from the coordinate origin in the transverse plane and 6 mm from the origin along the beam). This choice was made to be consistent with the large pattern and constant banks we had already produced. However FTK can handle beamlines not in the detector center (as SVT does in CDF).
- The SCT digitization was set to emulate the "edge mode" readout instead of the "level sense mode" default. Edge mode reduces SCT occupancies with little effect on tracking efficiency. In more recent versions of the offline software, it is the default for luminosities greater than 5×10^{33} $cm^{-2}s^{-1}$.
- The TRT was turned off to allow comparison between FTK and silicon-only offline tracking performance.
- Simulation of cavern backgrounds, which can be neglected for the present studies but which demands large computing resources at high luminosity, was turned off.

Samples were produced at three luminosities with 25 ns bunch spacing: no pileup, 1×10^{34} $cm^{-2}s^{-1}$ pileup, and 3×10^{34} $cm^{-2}s^{-1}$ pileup. The events for each luminosity were drawn from the same evgen+atlasG4 output. The cache of minimum bias events used for pileup digitization consisted of 90k events produced with the same (centered) beam spot distribution and 14 TeV collision energy as the primary interaction samples.

For non-zero luminosities, the digitization was configured to process hits from a range of out-of-time bunch crossings narrower than standard for the outer detectors, while still processing the usual range of out-of-time crossings for the silicon detectors. This change reduces the computing demands of the 14.2.25.10 simulation to the point where it can be run on typical Tier2 grid nodes. For luminosity up to 1×10^{34} $cm^{-2}s^{-1}$, the digitization processed bunch numbers from 8 bunches before to 6 bunches after the nominal crossing. For 3×10^{34} $cm^{-2}s^{-1}$, the range was narrowed, from 4 before to 3 after the nominal crossing. While these changes have no effect on the primary interaction or on the out-of-time pileup in the inner detector, the calorimeter and muon simulation will underestimate contributions from out-of-time-pileup.

Sample	Standard Job Options	Zero pileup events	1×10^{34} pileup events	3×10^{34} pileup events
$W\rightarrow$all + $H120\rightarrow bb$	MC8.105850.WH120bb_phythia.py	10k	10k	10k
$W\rightarrow$all + $H120\rightarrow uu$	MC8.105851.WH120uu_phythia.py	10k	10k	10k
VBF $H120\rightarrow\tau_h\tau_h$	MC8.105338.HerwigVBFH120tautauhh.py	10k	10k	10k
VBF $H120\rightarrow\tau_l\tau_h$	MC8.105334.HerwigVBFH120tautaulh.py	10k	10k	10k
VBF $H120\rightarrow\tau_l\tau_l$	MC8.105333.HerwigVBFH120tautaull.py	10k	10k	10k
Pythia dijet (17,35) GeV	MC8.105010.J1_pythia_jetjet.py	10k	10k	10k
Pythia dijet (35,70) GeV	MC8.105011.J2_pythia_jetjet.py	10k	10k	10k
Pythia dijet (70,140) GeV	MC8.105012.J3_pythia_jetjet.py	10k	10k	10k
Pythia dijet , jet filter	MC8.105802.JF17_pythia_jet_filter.py	20k	20k	20k
$W\rightarrow e\nu$, lepton filter	MC8.106020.PythiaWenu_1Lepton.py	10k	10k	10k
$W\rightarrow\mu\nu$, lepton filter	MC8.106021.PythiaWmunu_1Lepton.py	10k	10k	10k
Pythia bb, soft muon filter	MC8.108405.PythiaB_bbmu15X.py	40k	40k	40k
Minbias (pile-up cache)	MC8.105001.pythia_minbias.py	90k		
Single muons	(custom ParticleGenerator)	1.7B		

Table 3.1: Simulated datasets used for FTK studies. The numbers in parentheses for the dijet samples refer to the range of generated parton P_T. The filters mentioned are requirements made after generation but before simulation. The pile-up cache is the sample of minimum-bias events used for adding additional *pp* interactions at high luminosity.

3.3 The numbers of roads, fits, tracks in multijet events

In section 2, we presented the FTK data flow capability. Here we see whether it is sufficient for high luminosity operation. We use the pattern banks described in section 3.1.3 and *WH* events at 3×10^{34} $cm^{-2}s^{-1}$ as typical of the multijet events that come out of the level-1 trigger. The number of hits in each silicon layer was shown in section 2.3.2.1 to be handled adequately. Here we

focus on the rest of the chain, in particular the numbers of matched roads, track candidate fits, and output tracks.

3.3.1 Option A

The average number of roads per core crate per event is 167k. Since there are 16 AMBoards each with 4 LAMBs, this becomes 2.6k roads per LAMB going to a TSP. The TSP reduces the number of matched roads down to 1.5k which then are sent to a DO.

The number of track combinations that have to be fit per LAMB per event is 4.1k. The number of good tracks passing from the TF to the HW per LAMB is 150, and the number out of the HW is 12. Thus for all 4 LAMBs on an AMBoard, the total number of tracks to be passed to the second stage is 50.

These numbers are consistent with the transfer rates described in section II, and there is some margin. Moreover, there are a number of ways to increase the margin. The possible AM bank size described in section 2 is larger than that used here, allowing further bank optimization to reduce the number of output roads while maintaining high track efficiency. Another possibility to reduce fakes without increasing the bank size too much is to use a pattern bank with finer resolution only for low-probability patterns. To achieve this goal, we can have higher resolution superstrips, but use the finer resolution only for low-probability patterns. With the addition of the "don't care" feature that exists in commercial CAMs, we can maintain the efficiency gain from the low-probability patterns without suffering from the relatively large fake road rate they produce.

3.3.2 Option B

In the first stage, the average number of roads matched in the AM is 48k per core crate. There are 8 AM boards dedicated to the first stage, each with 4 LAMBs. Thus the number of matched roads per LAMB is 1.5k. This rate is small enough to pass directly to the DO on the AUX card, but we further reduce it with the TSP. The factor of 2 reduction that we currently estimate (see section 3.1.3.2) brings the number of roads per LAMB down to 750. The number of track fits per LAMB TF is 6k. The number of good tracks transferred to the HW is 110, with 65 coming out of the HW and sent to the second stage.

In the second stage, the number of matched roads per LAMB is 1.8k. Again the TSP reduction is estimated to be a factor of 2, bringing the number of roads sent to the AUX card down to 900. The number of track fits per TF is 3.5k. There are 20 good tracks that go to the HW, with 2 coming out of the HW and sent out of the Processor Unit (60 for an entire core crate).

These rates present no serious dataflow problems.

3.4 Single track efficiency and resolution

If FTK tracking is to be useful in e, μ, b, and τ triggering, it must perform well on single tracks. In this section, we show the efficiency and resolution for single-muon events without pile-up, but using the FTK settings that would be used at high luminosity. Comparison is made with offline tracking using selection criteria recommended for silicon tracking at high luminosity by the ATLAS ID group. Our requirements include a minimum of 9 silicon hits, and we apply a χ^2/dof cut of 5. For efficiency, the matching of a reconstructed track with a generated track requires that at least 70% of the silicon hits used in the fit be hits that were generated by the original particle. The plots show results for tracks with $P_T > 1\text{GeV/c}$.

The track efficiency is shown in figure 3.10 as a function of rapidity and transverse momentum. As we noted previously, we have to further study our logical layer assignment in the transition region between the barrel and the disks, where we see a drop in FTK efficiency. If necessary, we can use a trick we employed in the CDF SVT of adding fake hits in roads in which a logical layer is not adequately covered. This also works to deal with silicon modules that fail as the detector ages.

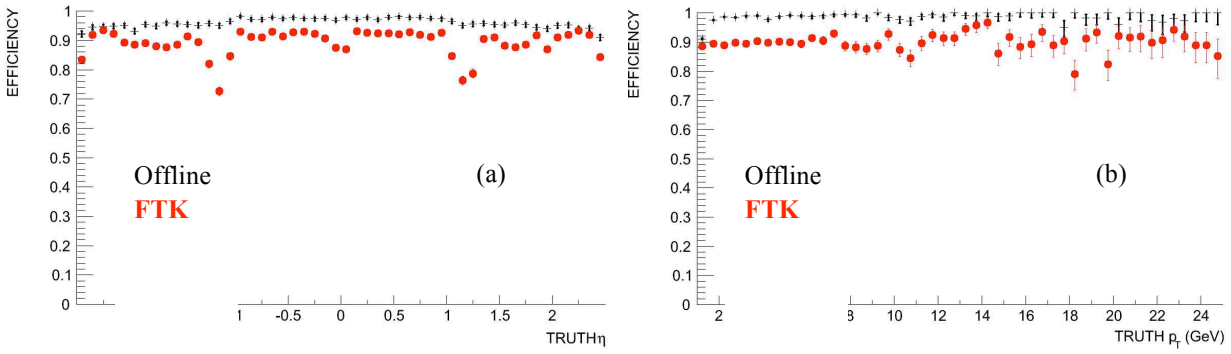

Figure 3.1 muon tracking efficiency as a function of y and (b) transverse momentum, comparin h offline tracking.

Helix parameter resolutions are similar to but not quite as good as offline reconstruction. Figure 3.11 shows the comparisons in the barrel region.

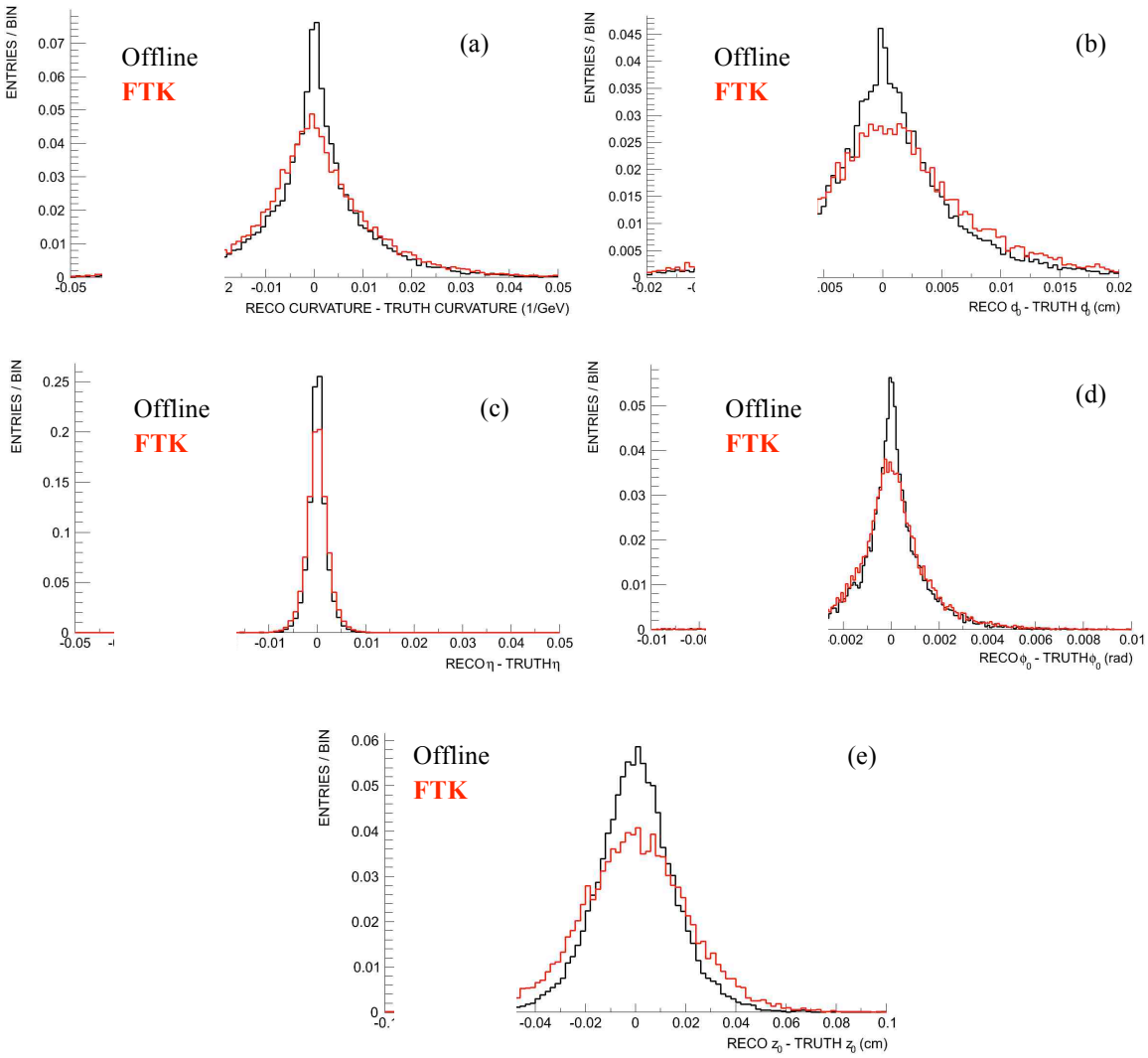

Figure 3.11: Comparison of FTK and offline helix parameter resolutions in the barrel region: (a) curvature, (b) d_0, (c) η, (d) ϕ_0, and (e) z_0.

3.5 b-tagging

New physics that couples to heavy fermions may be rich in final-state *b* quarks but, depending on the process, not necessarily leptons or very high P_T jets. Given the enormous QCD production of light quarks and gluons, it is important that the ATLAS trigger be able to efficiently select *b* jets while providing a large rejection factor against other jets. Secondary vertex *b* taggers like the ATLAS IP3D+SV1 tagger do well at separating *b* jets from light jets offline because they are able to obtain very high quality track information and to fully exploit it. Time constraints at level-2 make both the tracking and tagging difficult, and one must make compromises to perform both for all Regions of Interest within the 20 ms level-2 decision time. Since FTK tracks have near-offline efficiency, helix parameter resolution, and fake rates, the

immediate availability of high quality FTK tracks following a level-1 trigger would allow the entire 20 ms to be used for more sophisticated tagging algorithms. This could increase the correlation between level-2 and offline taggers and thus allow level-2 to have an operating point close to that of the offline, increasing the light jet rejection factor at level-2. This improvement in background rejection may prove crucial at very high luminosities where the tracking environment will be much more complex than at lower luminosity. In the following, we compare FTK and NewTracking in aspects key to *b* tagging and compare several FTK-based tagging algorithms with their offline equivalents.

To compare the performance of a tagger using FTK tracks with that of a tagger using NewTracking, we use the fully simulated samples of *WH* production (M_H = 120 GeV/c^2) described in section 3.2 to provide realistic samples of signal and background jets, *b*-quark and *u*-quark jets respectively. To suppress the substantial rate of fake tracks in NewTracking at 3×10^{34} cm^{-2}s^{-1}, we apply cuts very similar to those recommended in [27]:

- $N_{hits} \geq 9$
- $N_{pixel\ holes} = 0$
- $N_{SCT\ holes} \leq 2$
- $|d_0| < 1$ mm
- $|z_0| < 15$ cm

Figure 3.12 shows the efficiency to reconstruct tracks in these samples at 0, 1×10^{34}, and 3×10^{34} pile-up for NewTracking with the recommended cuts and for FTK tracking with the 11-layer simulation. Efficiency is measured for $P_T > 1$ GeV/c primary-interaction truth tracks with $|d_0| <$ 1 mm and $|z_0| < 12$ cm. A reconstructed track is required to have at least 70% of its silicon hits contributed by the truth track. As expected, NewTracking efficiency is rather stable with an 86% efficiency in the barrel at low luminosity and about 81% efficiency at 3×10^{34} pile-up. The 11-layer FTK efficiency is a bit lower at zero pile-up but degrades significantly at high pile-up, to about 60% in the barrel at 3×10^{34}. Also shown is the efficiency from simulation of the Option B FTK architecture at 3×10^{34} which is about 6% higher than that of the 11-layer simulation.

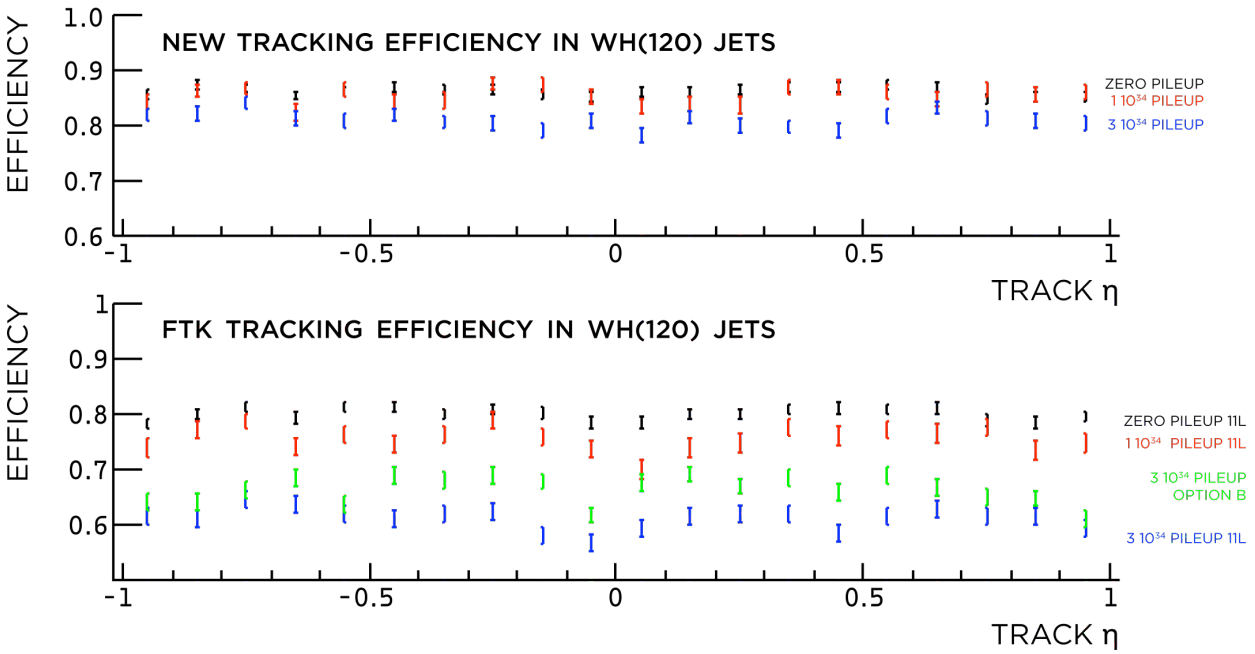

Figure 3.12: Track reconstruction efficiency in jets with 0 (black), 1×10^{34} (red), and 3×10^{34} (blue) pile-up for NewTracking and FTK 11-layer simulation. Also shown is FTK Option B at 3×10^{34} (green).

The difference in efficiency is under investigation. There is a contribution from FTK's stringent requirement of exactly 10 or 11 layers hit (at most 1 hole) and its disregard of multiple hits in a single layer (for example NewTracking uses both hits when there is one each in two silicon modules that overlap in ϕ in a single layer; thus there are often more than 11 hits on a barrel track). An estimate of the size of this contribution is the 5% lower NewTracking efficiency when tighter cuts are applied: $N_{hits} \geq 10$ and $N_{SCT\ holes} \leq 1$. Note, however, that in the following discussion, removal of these offline tracks actually improves the likelihood tagger performance.

Another contribution to the efficiency difference between NewTracking and the 11-layer FTK simulation at 3×10^{34} pile-up comes from the difference in χ^2 cuts. All NewTracking tracks considered have $\chi^2/\text{dof} < 10$, whereas the 11-layer FTK simulation requires $\chi^2/\text{dof} < 4$. We have not yet measured the gain from relaxing the 11-layer FTK cut to match the offline cut, but the same change in simulation of the Option B architecture increases the FTK efficiency by 6%.

Jet *b* tagging starts by identifying tracks associated with a particular jet. Level-1 or offline calorimeter clustering algorithms may require retuning for 3×10^{34} pile-up. To remove the effects of unreasonable clustering performance which are separate from tracking performance, we begin with 0.4-cone truth jets, associating with the jet all tracks having momenta within a distance of 0.4 in the η-ϕ plane from the truth jet centroid. We then apply a Δz_0 cut relative to the highest P_T track to reduce interference from pile-up and recompute the jet direction as the P_T-weighted track average η and ϕ. A jet is labeled a *b*-quark jet if, within a 0.3 radius of the jet direction, there is a *b* quark from the Higgs decay. If a jet is not labeled as a *b*-quark jet, it is labeled as a light jet.

Figure 3.13 (a), (b), and (c) show the distribution of transverse impact parameter significance for tracks in light quark jets at various luminosities. The FTK d_0 resolution, while about 30% larger than that of NewTracking, is still well suited for discriminating prompt tracks from those of *b*-jet decay (see figure 3.14), and both offline and FTK resolutions are stable up to 3×10^{34} pile-up (see figure 3.13d)[1]. Figure 3.15 plots the number of tracks in light jets with $d_0/O_{d_0} > 3$ at high luminosity for both algorithms, showing that the rate of spuriously high impact parameter tracks is roughly the same for both types of tracking.

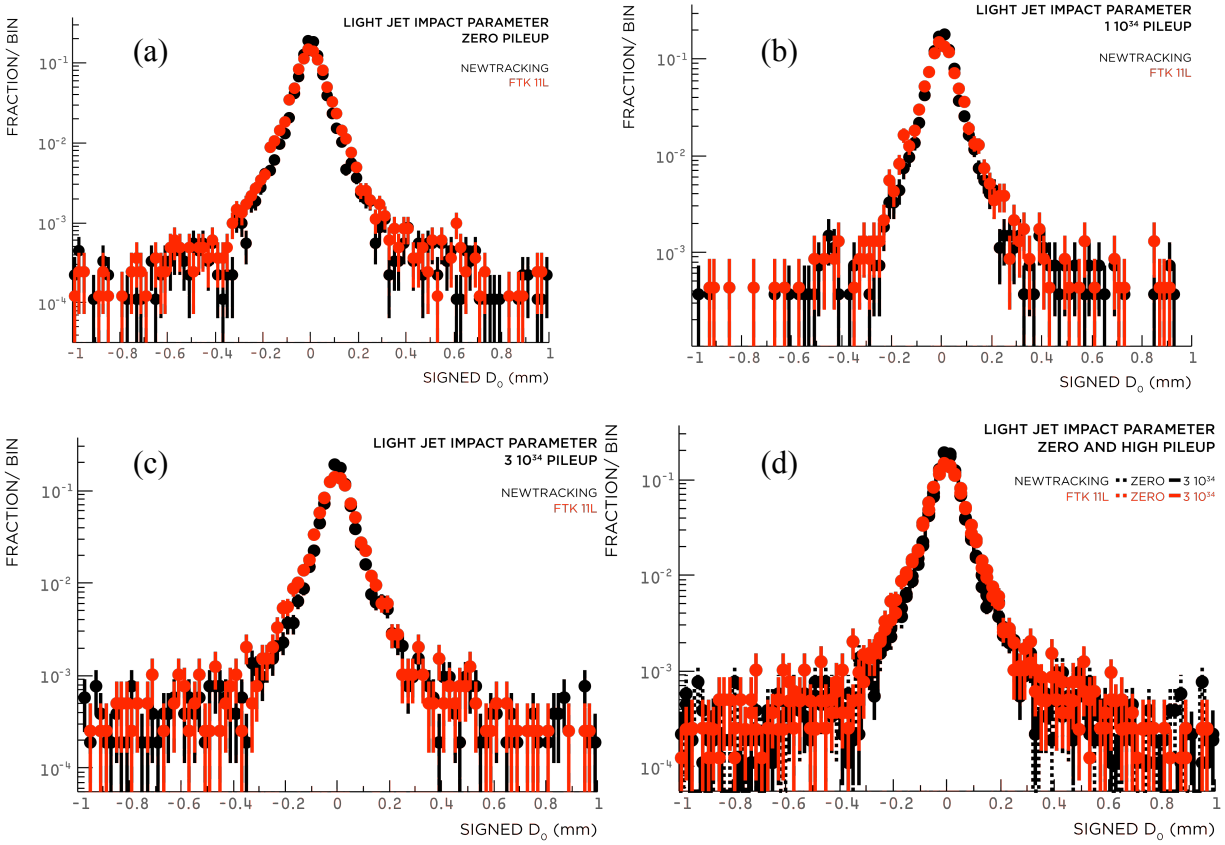

Figure 3.13: The signed impact parameter distributions from NewTracking (black) and FTK (red) for tracks in light-quark jets at (a) 0, (b) 1×10^{34}, and (c) 3×10^{34} pile-up. The stability as luminosity increases can be seen in (d).

[1] We note that FTK settings have not yet been optimized. For example, by reducing the sector width by a factor of 2 in the B layer only, which improves the linear fit approximation and can easily be implemented in the hardware, the difference between the FTK and NewTracking impact parameter resolution is reduced by a factor of 2.

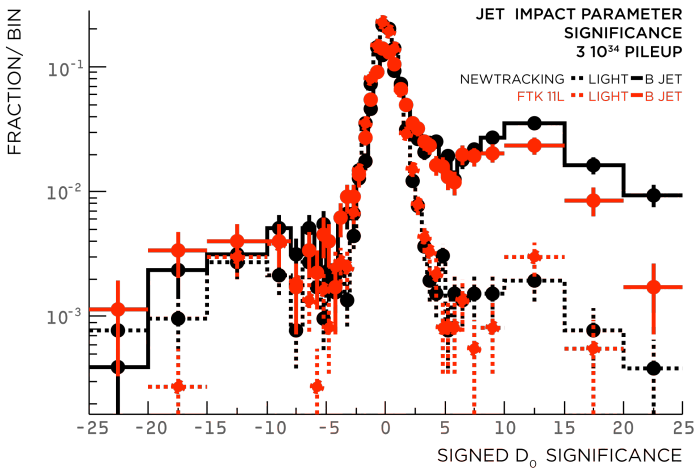

Figure 3.14: The signed impact parameter distributions at 3×10^{34} for *b* jets (solid) and light jets (dotted) for both NewTracking (black) and FTK (red). The increase at significance between 5 and 15 is due to the variable size bins.

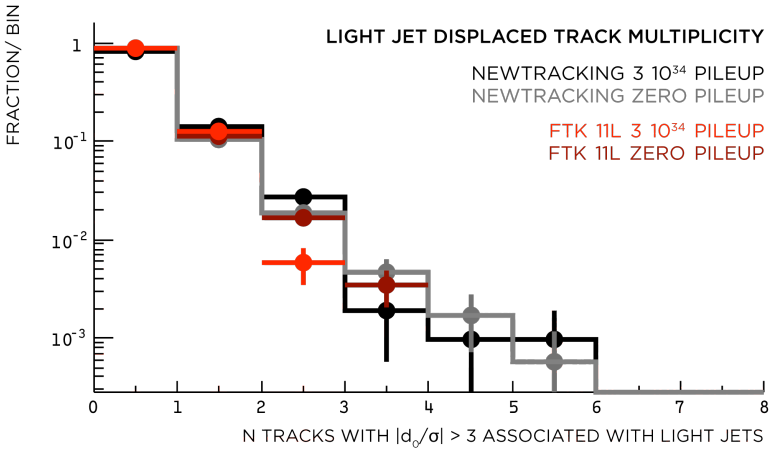

Figure 3.15: The number of light-quark tracks with signed impact parameter significance greater than 3 for NewTracking (black) and FTK (red) at 0 pile-up and 3×10^{34}.

With similar single track efficiencies, resolutions, and fake rates, we expect *b*-tagging performance using FTK tracks to be similar to that using NewTracking. To test this, we compare FTK with NewTracking using a tagging algorithm equivalent to the present level-2 d_0 likelihood tagger, which in turn is similar to the offline IP2D algorithm. The algorithm builds likelihood functions from the signed d_0 significance distributions of tracks in a sample of *b* and light jets (labeled as described above). The ratio, L_b/L_u, of the product of the likelihood functions evaluated for each associated track is used as a discriminant between the two types of jets, and one can tune the performance of the algorithm by varying a cut on this ratio. Figure 3.16 shows the light-quark rejection power, the inverse of the probability to tag a light-quark jet as a *b* jet. It is plotted as a function of the *b*-tagging efficiency for a statistically-independent sample of *WH* events. The likelihood functions were built for each track type (either FTK or NewTracking) and

for each luminosity. NewTracking-based tagging does not noticeably degrade within the available statistics from 0 to 3×10^{34} pile-up. FTK-based tagging performs well at all luminosities and at 3×10^{34} pile-up does approximately as well as the current level-2 algorithm at zero pile-up.

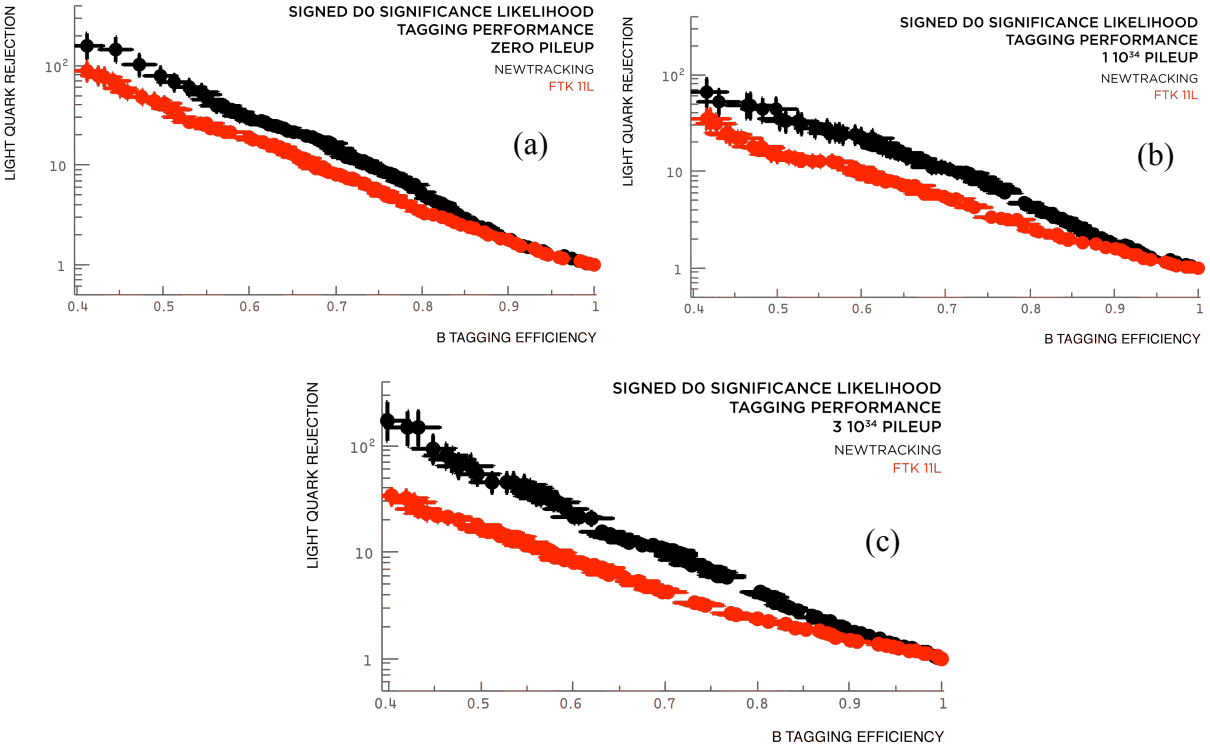

Figure 3.16: The light-quark rejection vs. *b*-jet efficiency for NewTracking (black) and 11-layer FTK simulation (red) at (a) 0, (b) 1×10^{34} and (c) 3×10^{34} pile-up.

With the simulation of the Option B FTK architecture, which has a higher single-track efficiency than in our 11-layer simulation, the *b*-tagging performance is even better at high luminosity. Figure 3.17 compares NewTracking with the Option B FTK simulation for 3×10^{34} pile-up.

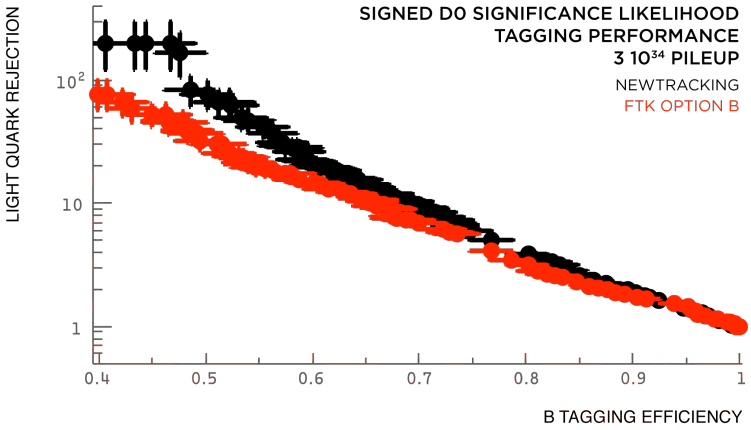

Figure 3.17: The light-quark rejection vs. *b*-jet efficiency for NewTracking (black) and Option B FTK simulation (red) at 3×10^{34} pile-up.

A possible improvement that we are studying makes use of the pixel cluster width information in track selection and/or the likelihood function definitions. We are also studying a secondary vertexing tagger that uses the same underlying constrained vertex fitting code as the offline secondary vertexing.

3.6 τ-identification

As the heaviest charged lepton, τ's are often among the decay products in electroweak symmetry breaking scenarios. To maintain good sensitivity to these as well as other new physics processes that produce τ leptons, high trigger efficiency for τ's down to the lowest possible τ P_T is important. Of necessity, this means good efficiency and high background rejection for hadronic τ decay, *i.e.* τ jets.

In hadron collider experiments, τ-jet identification has usually been track-based. Signal and the large QCD jet background are separated by the presence of 1 or 3 tracks in a very narrow cone with little or no track activity in a surrounding isolation cone. When tracking is not available in early trigger levels, calorimeter-based selection requires a narrow isolated jet. This is the essence of the ATLAS level-1 τ trigger. Here we propose rapid rejection of the QCD background by using FTK tracks at the beginning of the level-2 τ selection process. We show that FTK tracking does nearly as well as offline tracking when applied to τ selection.

Tau leptons almost always decay to 1 or 3 charged particles (plus neutrals). The latter presents a greater challenge because of its smaller branching ratio, the size of the QCD background, and the requirement that track reconstruction be efficient for tracks very close to each other. But for some processes, important τ polarization information can be extracted from 3-prong decays.

The τ-tagging algorithm is based on the offline algorithm which utilizes conical η-ϕ regions to search for tracks around the level-1 τ cluster and count tracks within signal and isolation regions.

Tracks with $P_T > 1.5\text{GeV/c}$ and within $\Delta R = 0.35$ of the level-1 τ cluster are considered. Within this cone, the FTK track with the highest P_T is found. If it has $P_T > 6\text{GeV/c}$, a signal cone of $\Delta R = 0.13$ is defined around it. Within the signal cone, there must be exactly 1 track in the case of single-prong τ's, and 2 or 3 tracks in the case of triple-prong τ's. An isolation cone of $\Delta R = 0.26$ is defined around the highest P_T track, and there must be no tracks of $P_T > 1.5\text{GeV/c}$ found between the signal and isolation cones. The cone sizes and kinematic ranges are chosen to maximize $S/\sqrt{B}$, where S is the τ signal and B is the background from dijets.

Figures 3.18 and 3.19 show the τ reconstruction efficiency for 1-prong and 3-prong τ's respectively. The data sample used is vector-boson-fusion Higgs production at 1×10^{34} $\text{cm}^{-2}\text{s}^{-1}$ luminosity with the Higgs decaying into two τ's each of which decays hadronically. The efficiency denominator contains truth τ's that are successfully matched to a level-1 τ cluster. FTK and NewTracking give similar efficiencies for both 1-prong and 3-prong τ's.

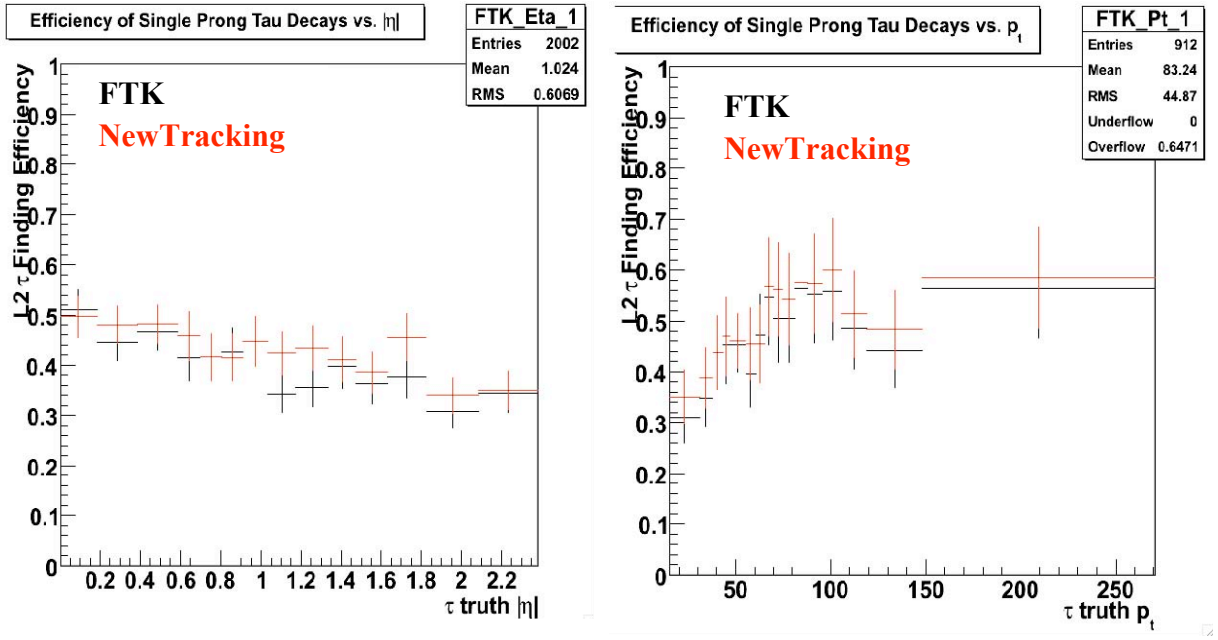

Figure 3.18: Single-prong τ efficiency vs. rapidity and transverse momentum, the latter for $|\eta| < 0.8$. FTK tracking (black) is compared with NewTracking (red) at 1×10^{34} $\text{cm}^{-2}\text{s}^{-1}$.

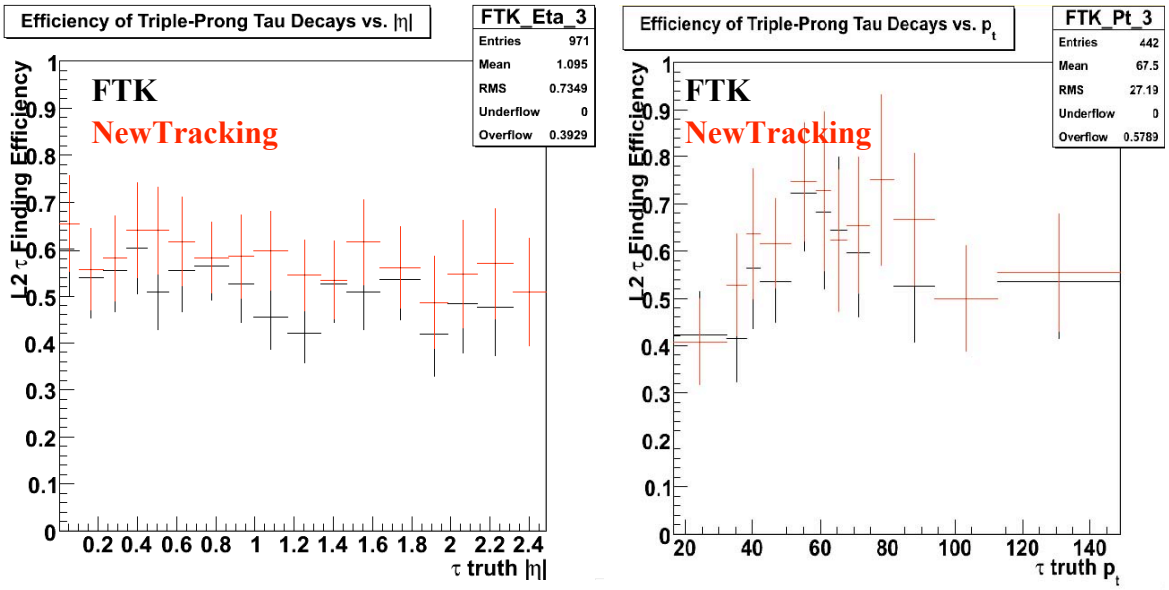

Figure 3.19: Triple-prong τ efficiency vs. rapidity and transverse momentum, the latter for $|\eta| < 0.8$. FTK tracking (black) is compared with NewTracking (red) at 1×10^{34} cm^{-2}s^{-1}.

The plots in figures 3.20 and 3.21 show the 1-prong and 3-prong fake probabilities. The numerator contains jets passing the full τ reconstruction, and the denominator is all jets. Again, both FTK and NewTracking results are shown.

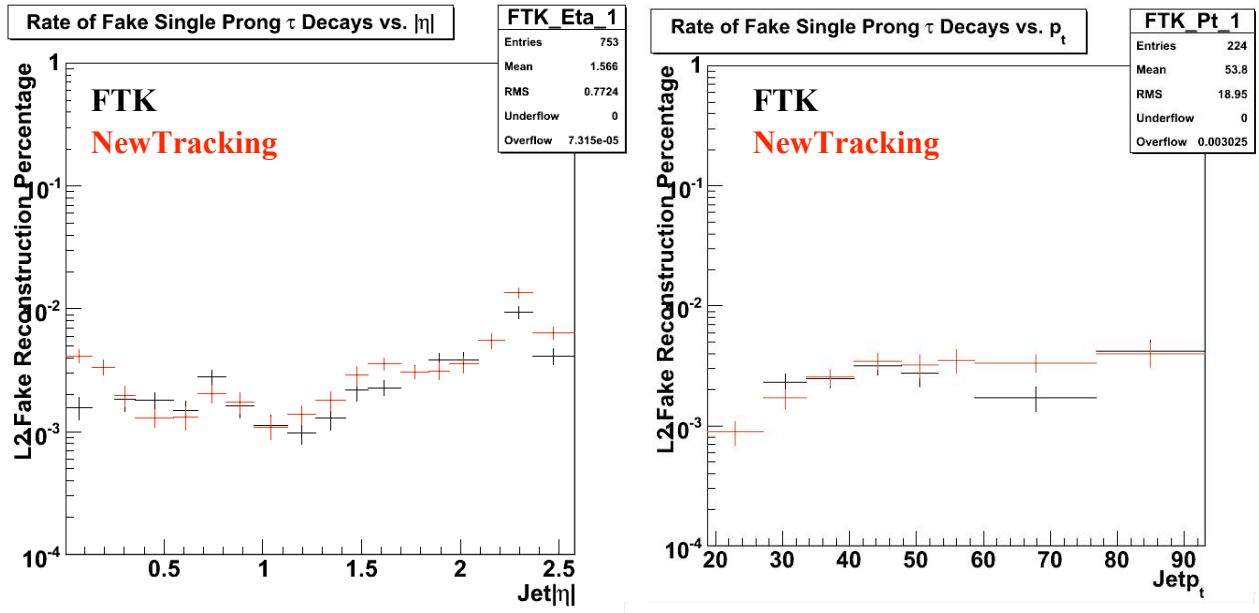

Figure 3.20: Single-prong fake τ probabilities vs. rapidity and transverse momentum, the latter for $|\eta| < 0.8$. FTK tracking (black) is compared with NewTracking (red) at 1×10^{34} cm^{-2}s^{-1}.

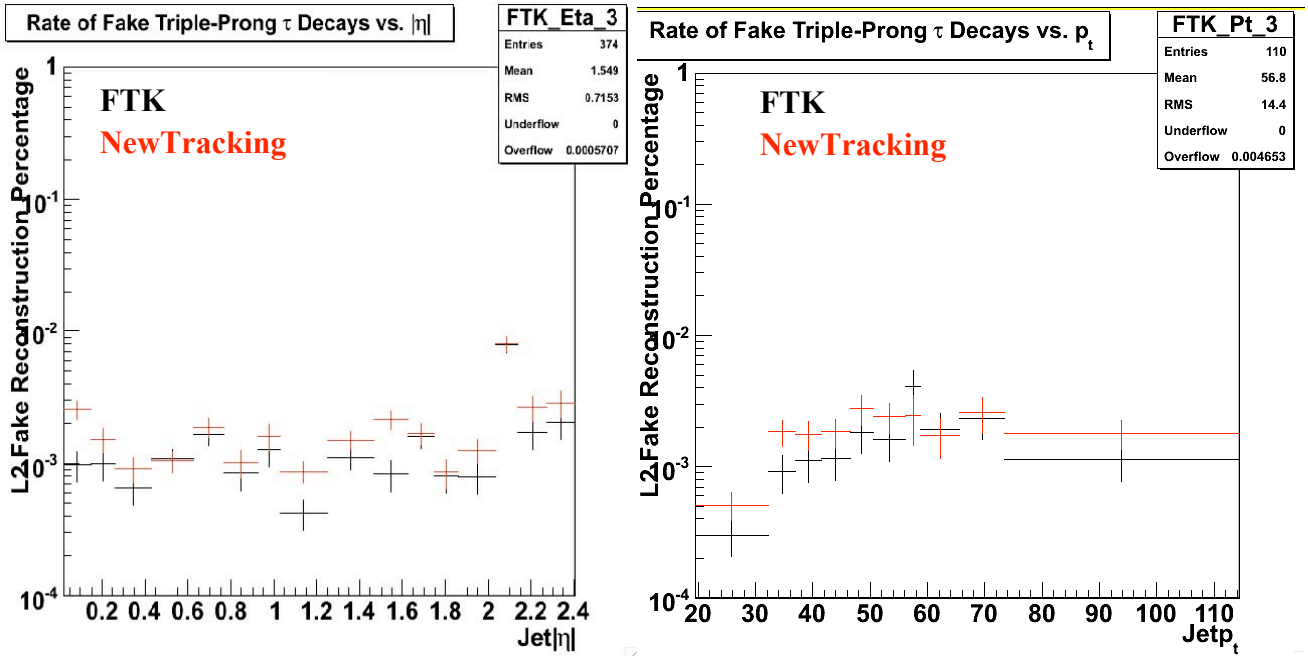

Figure 3.21: Triple-prong fake τ probabilities vs. rapidity and transverse momentum, the latter for $|\eta| < 0.8$. FTK tracking (black) is compared with NewTracking (red) at 1×10^{34} cm^{-2}s^{-1}.

The τ identification at 3×10^{34} cm^{-2}s^{-1} is lower than at 1×10^{34} cm^{-2}s^{-1} since tracking efficiency drops and pile-up tracks in the isolation cone causes τ's to be lost. The results are shown in figures 3.22 and 3.23 for 1-prong and 3-prong τ's respectively.

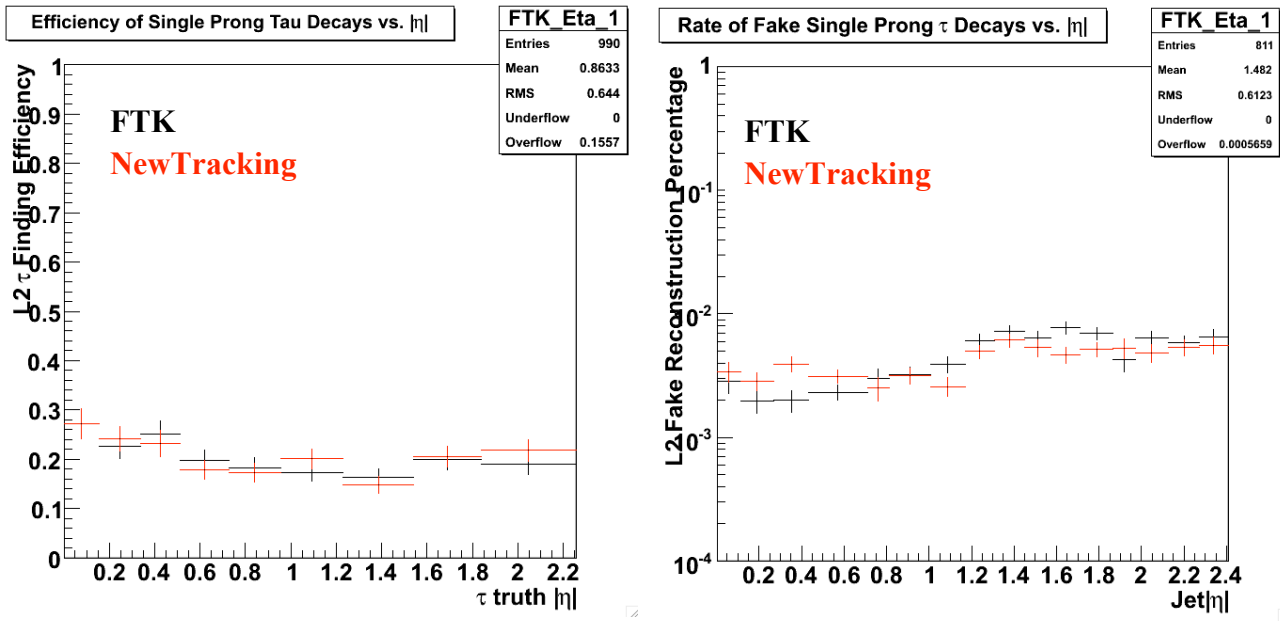

Figure 3.22: The FTK and NewTracking efficiency (left) and fake probability (right) for 1-prong τ's at 3×10^{34} cm^{-2}s^{-1}.

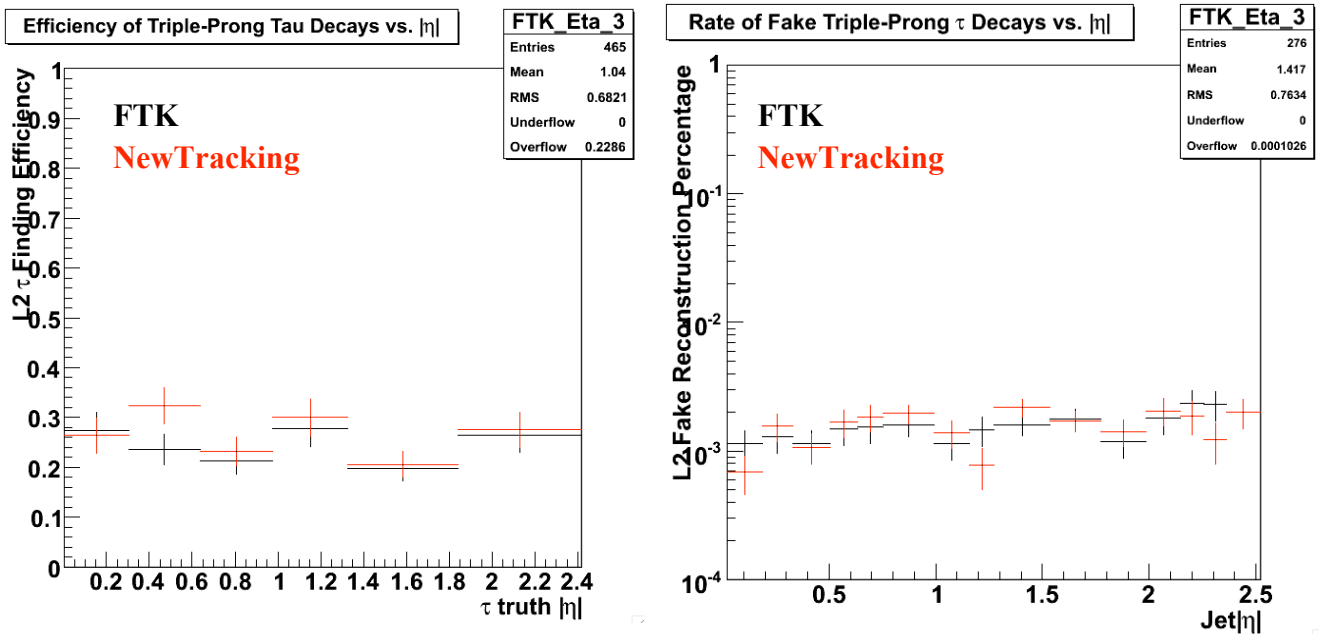

Figure 3.23: The FTK and NewTracking efficiency (left) and fake probability (right) for 3-prong τ's at 3×10^{34} cm^{-2}s^{-1}.

Efficiency can be recovered with very little change in the fake probability by only using tracks with z_0 close to that of the seed track ($|\Delta z_0| < 3$ mm for the signal cone and < 1 mm for the isolation cone). Figure 3.24 shows the efficiencies for 3×10^{34} cm^{-2}s^{-1} in this case. There is about a 30% relative efficiency improvement over not using the $|\Delta z_0|$ requirement.

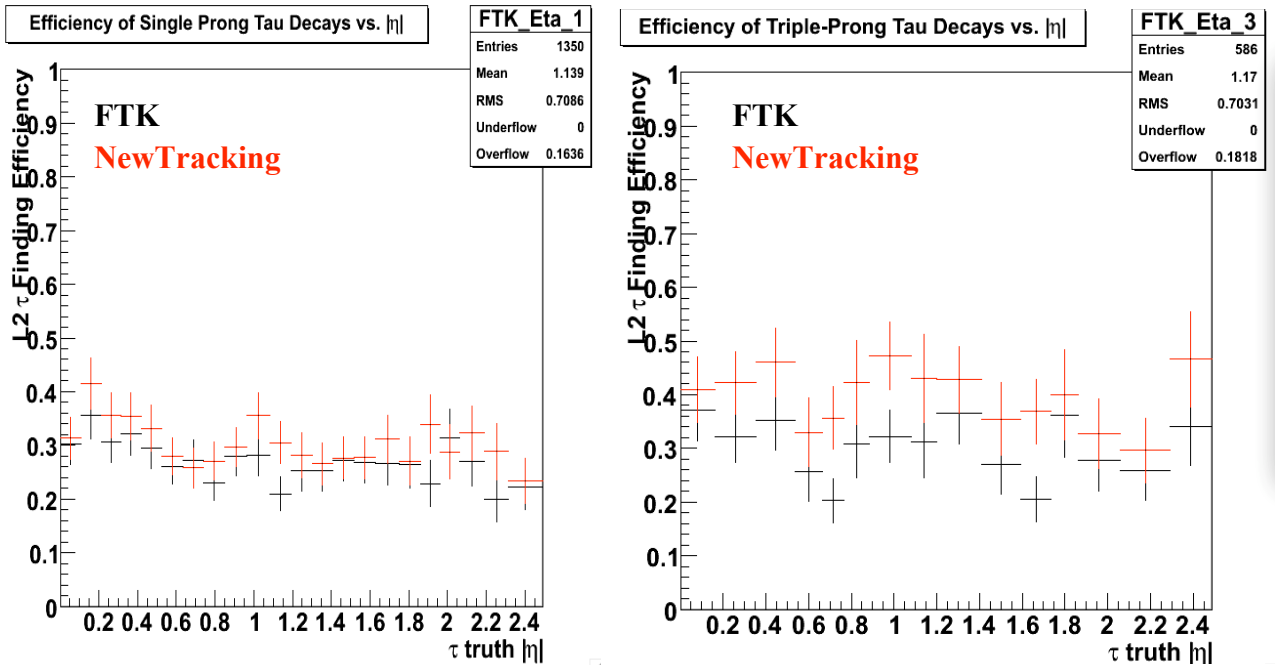

Figure 3.24: : The FTK and NewTracking efficiency for 1-prong (left) and 3-prong (right) τ's at 3×10^{34} $cm^{-2}s^{-1}$ with a $|\Delta z_0|$ requirement on tracks relative to the seed track.

3.7 μ and e triggering at high luminosity

Traditionally, single lepton triggers rely on calorimeter isolation to suppress backgrounds from real or fake leptons in hadronic jets. This works well at low luminosity. However at high luminosity, when there is substantial calorimeter energy due to the pile-up interactions, this strategy deteriorates. If the isolation threshold is kept low to maintain background rejection, the efficiency drops for leptons of interest. If the isolation threshold is raised, lepton efficiency can be maintained, but at the price of decreased background rejection.

An alternative strategy for high luminosity is to apply isolation-based on reconstructed tracks. If all tracks in the event are used, pile-up remains a serious problem. However for tracking, unlike calorimeter deposition, the pile-up and hard-scatter particles can be separated. Here we analyze a track-based isolation using only those FTK tracks that have a z_0 close to that of the lepton candidate.

3.7.1 μ trigger

The selection of isolated muons is critical to the searches for new physics such as SUSY cascades involving high P_T muons and a high mass Z' decaying into two muons, as well as the study of Standard Model processes such as $W \to l\nu$ or $Z \to ll$. In the ATLAS HLT, both calorimeter-based and tracking-based isolation are used to reject major backgrounds like $b\bar{b}$

events where one *b*-quark decays to a muon and light-quark jets where the jet produces tracks in the muon spectrometer, faking a muon. Since the rates for $b\bar{b}$ and light-quark jet production are significantly higher than those for isolated muon processes, it is critical to suppress the background rate in the trigger while maintaining high efficiency for isolated muons.

At high luminosity, calorimeter-based isolation loses its effectiveness in selecting real isolated muons. When the number of pile-up collisions increases, the amount of energy in the calorimeter, especially the EM calorimeter, also increases. With fast tracking information available from FTK, calorimeter-based isolation in the trigger can be replaced with a tracking-based isolation that uses only tracks pointing to the z_0 of the muon track. This tracking-based isolation removes any requirements on the calorimeter cell energies, while maintaining high efficiency for isolated muons in environments with large pile-up.

In the current implementation of the HLT, the isolated muon trigger requires that the muon candidate pass isolation cuts based on the energy in EM and hadronic cells and the P_T of inner detector tracks. EM isolation is defined as the sum of all cells with energy above a 60 MeV threshold that lie in a ring of ΔR between 0.07 and 0.4 around the muon track. For hadronic isolation, the cell energy threshold is also 60 MeV and the isolation ring extends from R of 0.1 to 0.4[1]. Tracking isolation is defined as the P_T of the muon track as measured by the inner detector divided by the P_T sum of all inner detector tracks within ΔR<0.2 of the muon track, including the muon track.

The EM calorimeter isolation energy is shown in figure 3.25a for isolated muons from $W \to \mu\nu$ and non-isolated muons from $b\bar{b}$ jets at 1×10^{34} $cm^{-2}s^{-1}$ luminosity.[2] As seen in the figure, isolated muons tend to have smaller energy in the EM isolation ring. However the EM isolation distribution for muons from the $b\bar{b}$ events has significant overlap with the isolated muon distribution. For the same events, tracking isolation is shown in figure 3.25b. For isolated muons, this distribution is centered close to one. In contrast to EM calorimeter isolation, the track isolation distribution for $b\bar{b}$ events is more separated from the isolated muon peak.

[1] The calorimeter cell energy cuts in the HLT have not been optimized for pile-up scenarios. As will be demonstrated later, this does not change the conclusion that tracking-based isolation is more robust to pile-up than calorimeter-based isolation.

[2] Fake muons from light quark jets are not shown since the calorimeter isolation energy for these events is much larger and the overlap with the isolated muon distribution is minimal.

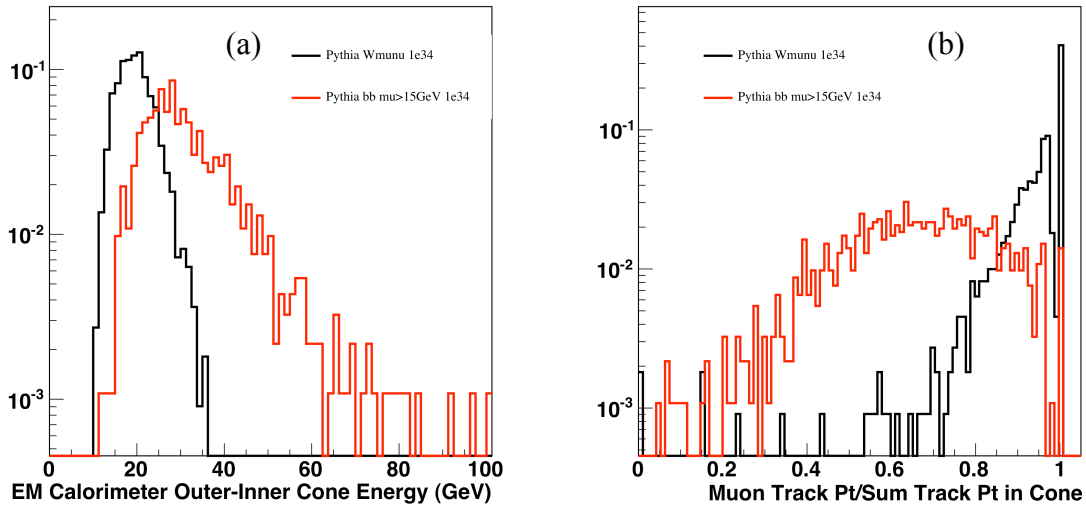

Figure 3.25: (a) EM calorimeter isolation and (b) track isolation for signal and background with 1×10^{34} $cm^{-2}s^{-1}$ pile-up.

To quantify the trigger efficiency for isolated muons at high luminosity, the calorimeter and tracking isolation cut values are set so that the trigger rejection factor for $b\bar{b}$ events is 10. Figure 3.26a shows the isolated muon trigger efficiency as a function of the number of pile-up interactions using only EM calorimeter isolation. The trigger efficiency quickly deteriorates with increasing pile-up. Also shown in the figure is the trigger efficiency when the EM cell energy threshold is increased by a factor of two. The efficiency degradation with increasing pile-up is still clearly visible. Figure 3.26b shows the isolated muon efficiency when instead tracking isolation is used. Here inner detector tracks, selected using offline reconstruction, must have $P_T > 1\text{GeV/c}$ and have at least one hit in either the pixels or SCT. In contrast to calorimeter isolation, tracking isolation is less sensitive to pile-up. The trigger efficiency can be further improved by using only inner detector tracks in the cone that have z_0 within 10 mm of the muon z_0. As seen in the figure, the trigger efficiency for isolated muons using this track selection is insensitive to pile-up at this luminosity.

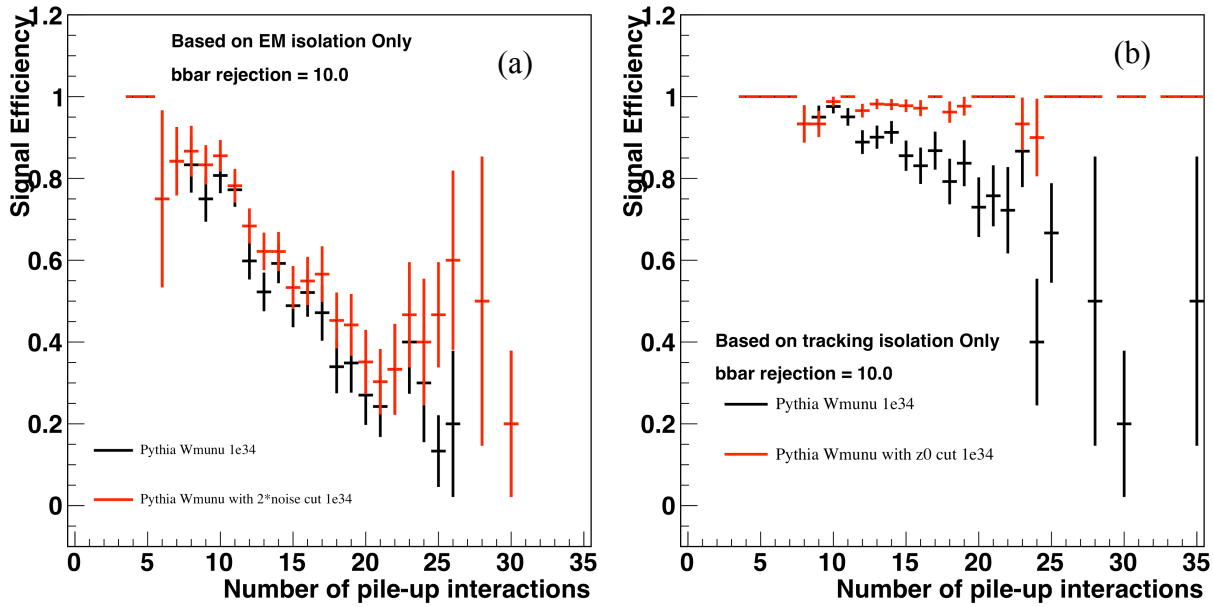

Figure 3.26: Isolated muon efficiency using (a) EM calorimeter with two cell energy thresholds or (b) tracking isolation without (black) or with (red) a Δz_0 cut as a function of the number of pile-up interactions in the event. The isolation cut is selected to provide a $b\bar{b}$ rejection factor of 10.

Using FTK tracks to calculate the tracking-based isolation yields similar trigger efficiencies for $W \to l\nu$ events. As seen in figure 3.27, the efficiency as a function of pile-up is constant for large numbers of pile-up events using FTK tracking. For comparison, the trigger efficiency using EM-based isolation is also shown in the figure.

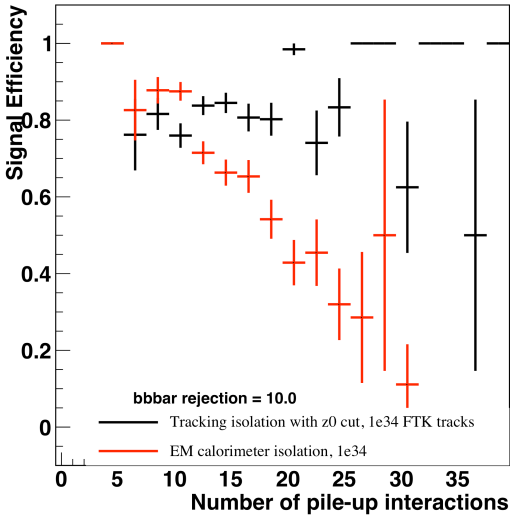

Figure 3.27: The $W \to l\nu$ trigger efficiency using FTK track isolation (black) or EM calorimeter isolation (red) as a function of the number of pile-up interactions in the event. The isolation cut is selected to provide a $b\bar{b}$ rejection factor of 10.

At a higher luminosity of 3×10^{34} cm^{-2}s^{-1}, track-based isolation using FTK tracks continues to be effective. As seen in Figure 3.28, the efficiency for selecting isolated muons using FTK tracking-based isolation is constant even for events with 100 pile-up interactions. For both of the 1×10^{34} cm^{-2}s^{-1} and 3×10^{34} cm^{-2}s^{-1} samples, the track-based isolation cut was tuned to give a $b\bar{b}$ rejection factor of 10. The overall efficiency for isolated muons is approximately 80% for both luminosities.

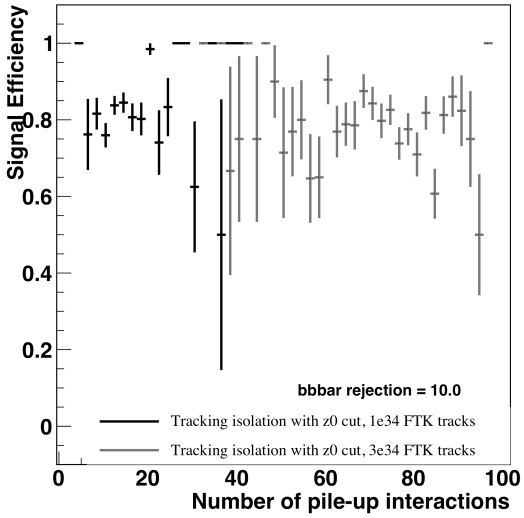

Figure 3.28: The $W \to l\nu$ trigger efficiency using FTK track isolation at 1×10^{34} cm^{-2}s^{-1} (black) and 3×10^{34} cm^{-2}s^{-1} (gray) as a function of the number of pile-up interactions in the event. The isolation cut is selected to provide a $b\bar{b}$ rejection factor of 10.

For large luminosities, the isolated-muon trigger efficiency deteriorates dramatically with increasing pile-up when calorimeter-based isolation is used. In contrast, tracking-based isolation with fast tracking from FTK is insensitive to the amount of pile-up. Using fast tracking information in the isolated-muon trigger results in high trigger efficiency for signal events while still maintaining good rejection of the $b\bar{b}$ background.

3.7.2 *e* trigger

The electron case is more difficult to assess than the muon case. Calorimeter isolation is inherent in the level-1 selection. We have to understand how pile-up at very high luminosity would affect level-1 and how best to mitigate the effect. Then we have to investigate the rate reduction needed in level-2 using tracks with a $|\Delta z_0|$ cut and/or calorimeter information. Since the bulk of the events passing the level-1 trigger comes from the fragmentation tail of QCD jet events, a background sample is needed for these studies. The e-gamma group uses a very large number of such events, many more than we could process through digitization at 3×10^{34}. We

will have to investigate ways of further preselecting those QCD events that would pass the level-1 trigger and then digitize only those.

3.8 FTK timing

3.8.1 FTK timing simulation

A simulation tool for calculating FTK execution time was developed for tuning the system architecture and parameters and to ensure that FTK can handle a 100 kHz level-1 trigger rate at high luminosity. The system is divided into functional blocks: DF, DO write mode (receiving hits from DF and sending SSs to AM), AM, DO read mode (receiving matched roads from AM and sending roads and hits to TF), TF, and HW. The focus so far has been on the most time consuming steps, from DO write mode through TF. The DF should add little to the overall latency since each cluster found is immediately sent to the DO. Thus the DF and DO execution times almost completely overlap. Similarly, the HW has a relatively short latency for each track that enters before the track is either sent to the output or discarded. Both the DF and HW functions will be added in the near future.

For each functional block, the time of the first and last words into and out of the block are calculated. Since each core crate operates independently, the FTK event execution time ends when the last word exits the busiest crate for that event. The execution time for a block depends on the number of input words, the processing time per word, and the number of output words. We estimate the processing time per word for each block type from the architecture, our experience with previous prototypes, and the available chips on the market (see table 3.2). The numbers of input and output words for each block come from FTKSim events. The results reported here use *WH* events with 3×10^{34} $cm^{-2}s^{-1}$ pileup.

Function:	Data word processing rate:	Input data word rate:	Latency for a data word:
DO write mode	100 MHz	-	40 ns
AM	400 MHz	100 MHz	250 ns
TSP	200 MHz	400 MHz	250 ns
DO read mode	200 MHz	200 MHz	200 ns
TF	500 MHz	200 MHz	300 ns

Table 3.2: The properties of the functional blocks currently implemented in the FTK timing simulation. A dash indicates that the speed is high enough to have negligible effect on the overall timing.

The solid angle covered by a core crate is divided into η-ϕ towers. Each tower has its own unique set of AM roads. However the hits from the DFs often have to be sent to more than one

tower due to curvature in ϕ of ≥1 GeV/c tracks, the length in z of the LHC luminous region, and multiple scattering. The amount of hit duplication was determined from FTKSim and used by the timing simulation.

These studies assume very large input FIFO buffers. During the engineering design phase, we will do queuing studies to determine the needed depth of each input buffer.

3.8.2 FTK timing results at 3×10^{34} cm^{-2}s^{-1}

Figure 3.29 shows example timing charts for individual *WH* events with 3×10^{34} cm^{-2}s^{-1} pileup. In each case, the timing is shown for the core crate that finishes processing the event last. For the event on the left, the previous event had already completed processing before the new one arrived. For the event on the right, FTK had not completed processing the previous event and thus waits to start the new event.

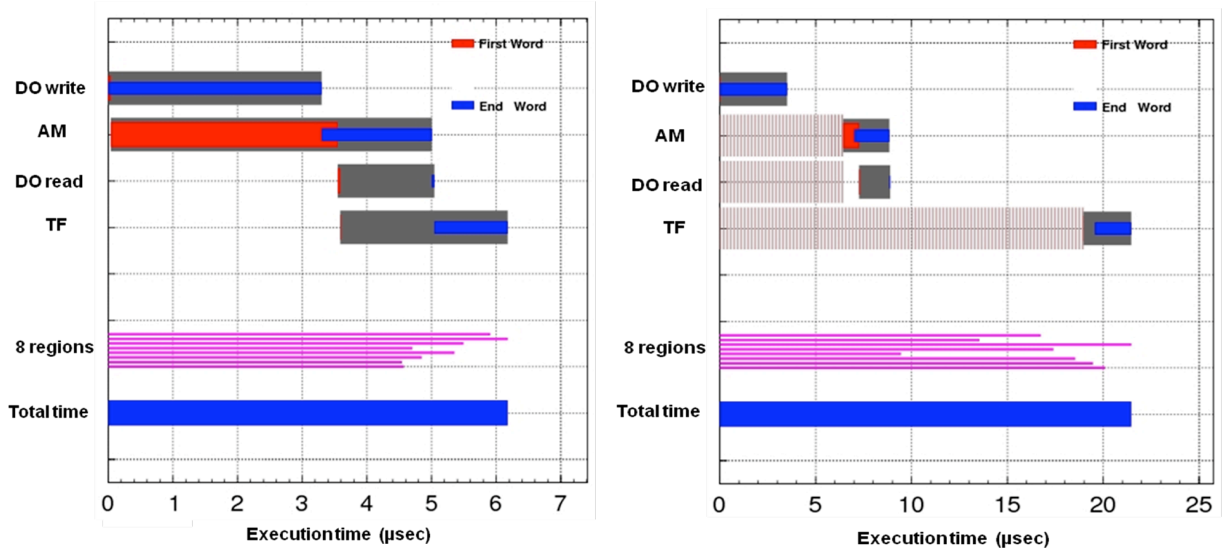

Figure 3.29: FTK execution time for two events at 3×10^{34} cm^{-2}s^{-1}. In both cases, the timing of the functional blocks is given for the core crate (region) that takes the most time. The time for each of the 8 regions is shown below that, with the total execution time shown in the bottom bar. For the event on the left, FTK had completed processing the previous event before the current event arrived. For the event on the right, FTK was still processing the previous event (shown as the light shading in the AM, DO read, and TF bars).

To get higher statistics on event execution times and see whether FTK can handle the 100 kHz level-1 trigger rate, we analyzed a larger sample of events. If this rate were too large for our system, we would see the event execution time (from input hits available to event completion) steadily increase as FTK falls behind, working on a stack of previous events before getting to the current one. This does not happen, as seen in Figure 3.30. Some events take longer than others to do global tracking, but after such an event the execution time quickly returns to the typical range. Figure 3.31 has the timing histograms for options A and B.

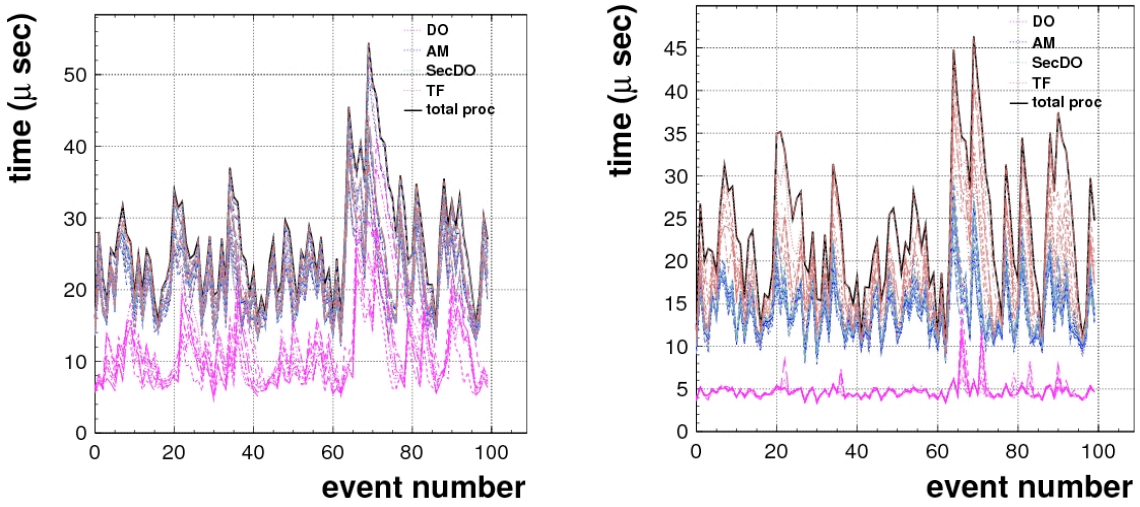

Figure 3.30: FTK execution time for 100 *WH* events at 3×10^{34} $cm^{-2}s^{-1}$. For each event, the execution time starts when the event is available (10 μs after the previous event, corresponding to a 100 kHz level-1 trigger rate) and ends when the FTK has completed analyzing that event. The plot on the left (right) is for option A (B).

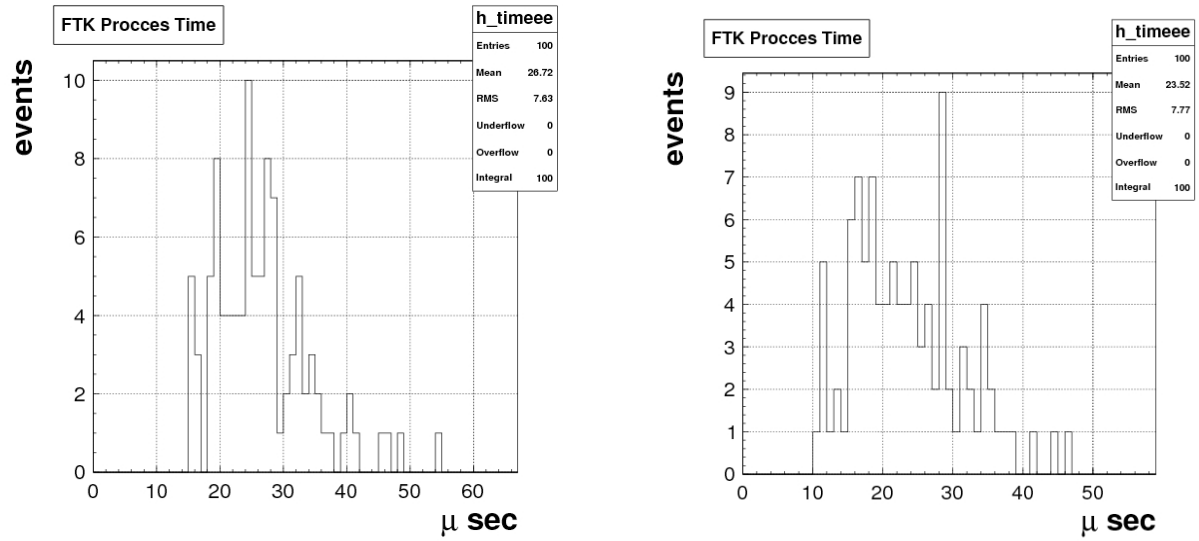

Figure 3.31: FTK execution time for 100 *WH* events at 3×10^{34} $cm^{-2}s^{-1}$ for (left) option A and (right) option B.

The times shown here are for the first stage in each option. We expect the time added by the second stage will not be large because the number of track candidates out of the first stage is not great and the execution of the two stages largely overlap since the second stage starts processing as soon as the first track is output from the first stage. We will add the second stage to the timing simulation in the near future.

The bottom line is that both options operate well for a 100 kHz level-1 trigger rate at 3×10^{34} $cm^{-2}s^{-1}$.

3.9 Timing studies with the current level-2 system

At high luminosity, sophisticated level-2 algorithms will be needed to suppress high-rate backgrounds while maintaining good signal efficiency. FTK will enable early rejection of background events and leave more time for sophisticated algorithms by moving track reconstruction from the level-2 farm to dedicated hardware. In the current system, the level-2 farm must do track reconstruction in all relevant Regions of Interest (ROIs). FTK will have global track reconstruction completed shortly after level-2 processing begins.

Determining the size of the HLT farm needed for high luminosity running is a substantial task with many current unknowns that will be better understood when there is more beam-on experience with the current system and when the future direction of the marketplace is better known. In this section, we study track reconstruction timing with the current system. A modern multi-core computer (Intel Core2 Duo 3.0 GHz E8400 CPU) processes RDO files produced from high-luminosity simulation.

We are running TrigSiTrack for the current studies. The HLT software is the code obtained from setting up the Athena framework with –tag=AtlasProduction,15.6.1.3,opt,slc4,runtime,32. The data samples contain $WH(uu)$ events generated at 1×10^{34} and 3×10^{34} $cm^{-2}s^{-1}$ to represent busy events involving leptons, jets, and pile-up interactions that the trigger tracking algorithms need to process. We enable only a simple L2_b*x* chain that is run on level-1 jet ROIs above some E_T threshold *x* (*i.e.* J*x*) which we vary over a wide range in the menu configuration.

Table 3.3 summarizes the results as a function of jet ROI E_T threshold and luminosity. Figure 3.32a shows the distribution at both luminosities of the TrigSiTrack execution time per ROI for a level-1 jet ROI threshold of 20 GeV. Figure 3.32b shows the execution time per event for the same threshold at 1×10^{34} luminosity.

L ($cm^{-2}s^{-1}$)	L1 jet threshold (GeV)	# of ROIs	<ROIs>/event	msec/ROI	msec/event	offline jets/event
1×10^{34}	20	558	6.6	11	73	28
1×10^{34}	40	162	1.9	14	31	8
1×10^{34}	70	29	0.3	23	26	2
1×10^{34}	100	11	0.1	31	31	1
3×10^{34}	20	5100	51.0	26	1347	85
3×10^{34}	40	5036	50.4	26	1331	50
3×10^{34}	70	3888	38.9	28	1089	17
3×10^{34}	100	2062	20.6	29	590	7

Table 3.3: TrigSiTrack level-2 execution time based on 84 events at 1×10^{34} and 100 events at 3×10^{34}.

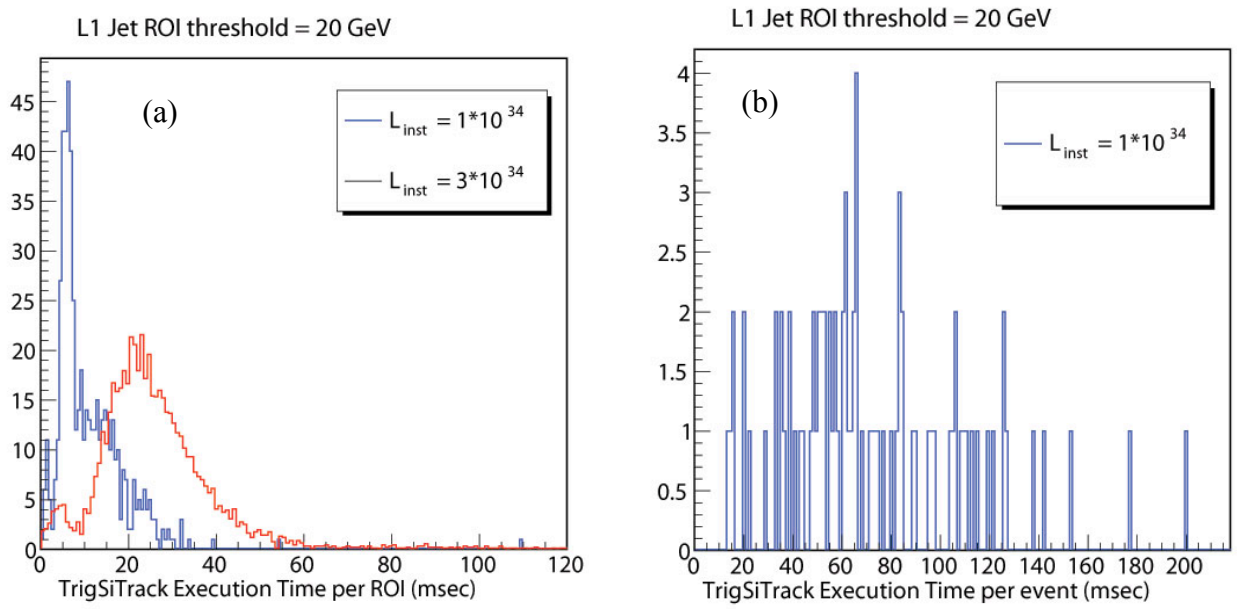

Figure 3.32: The TrigSiTrack execution time (a) per ROI at 1×10^{34} and 3×10^{34} and (b) per event at 1×10^{34}.

The behavior of the number of ROIs per event at 1×10^{34} as the E_T threshold increases seems reasonable, but the behavior at 3×10^{34} is surprising. On the other hand, the uniform execution time per ROI as a function of E_T is reasonable since the number of silicon hits at 3×10^{34} is dominated by the pile-up interactions. As a sanity check on the variation of the number of ROIs, we looked at the number of jets found by offline reconstruction, and those numbers are also in the table. The jet collection is the H1 algorithm using towers in a cone of 0.4. The number of jets is a bit underestimated because we had to use a smaller range of crossings for pile-up in the calorimeter in order to keep the 3×10^{34} digitization jobs from running out of memory.

There is surely optimization that will have to be done for high luminosity, for example in the minimum energy needed for an element to be added to the jet. However it is clear that we have to be prepared for a substantial number of jet ROIs. This is particularly true if the main characteristic of the new physics is multiple jets with displaced vertices, which occurs for example in $b\bar{b}Hb$® 4's and some of the Hidden Valley scenarios which also produce 4 jets with displaced vertices, some of moderate E_T. Such processes would likely need a 4-jet level-1 trigger. At our current level of understanding, the TrigSiTrack execution time for multijet events could easily range upward of several hundred milliseconds.

4 Upgrade path to SLHC Phase II luminosity

All of our effort has been focused on designing a system that will work well at 3×10^{34} $cm^{-2}s^{-1}$. In the future, we will simulate 5×10^{34} $cm^{-2}s^{-1}$, which is expected during the SLHC Phase II era.

Implementing the details of the replacement inner detector including its geometry, channel count, and read-out scheme will be essential. With that knowledge, we will be able to understand the challenges to FTK and how they can be best solved.

The heart of our system is pattern matching. As the luminosity increases, the number of fake matched roads increases rapidly because of the increased silicon occupancy. To counter that, we must reduce the width of our roads. If we accomplish this by increasing the size of the AM pattern bank using the current technology, the system would become very large and extremely expensive. To solve the problem in an affordable way, we are pursuing two directions. The first is to increase the number of patterns stored per chip by exploiting 3D chip technology. The second is to employ Tree Search Processors as described in section 2. However in Phase II the large data flow of matched roads and the sequential nature of the TSP will require high parallelism. The ideal solution would be a TSP for each AM chip, which could be implemented in a 3D chip that combines both functions. We could have a few layers for the associative memory, a layer for the TSP logic, and a few layers for the TSP bank memory. We will study the performance of such a system with 5×10^{34} $cm^{-2}s^{-1}$ *WH* events and investigate the feasibility of building such a chip.

There are other possible changes to FTK that might be needed to handle the 70% increase in the mean number of hits per silicon layer. We could increase the number of core crates to 12 to handle the increased number of matched roads and track fits. Another possibility is to increase the minimum track P_T from 1.0 to 1.5 or 2.0 GeV/c to reduce the size of the pattern bank and the numbers of matched roads and fits. A physics study using ATLAS data will enable us to assess the impact of such a change on *b*-tagging, τ ID, and single lepton isolation.

5 Resources

5.1 The FTK collaboration

The FTK collaboration currently consists of nine institutions, four from the U.S., four from Italy, and one from Japan. One of the groups, Fermilab, is not at present an ATLAS institution. However the Fermilab Director has publicly stated on a number of occasions that the lab will apply for ATLAS membership as soon as the Fermilab Tevatron Collider run ends, in about 18 months, and he has discussed his plans with ATLAS management. Until that time, Fermilab FTK scientists can participate as visitors through Argonne or Chicago.

5.2 Possible distribution of responsibilities

We have an initial plan for the design, construction, testing, and operation of FTK. The Bologna group will work on the initial vertical slice implementation using the EDRO board as described in section 6. The Pavia group will focus on commissioning tests starting with the initial vertical slice and on software development. The remaining seven groups will design and build the full FTK system. The responsibilities in the initial plan are as follows:

- Frascati – the Data Formatter mezzanine card that receives the pixel hits and does two-dimensional clustering.
- Chicago – the Processor Unit auxiliary card that contains the Data Organizer, Track Fitter, and Hit Warrior.
- Pisa – the Processor Unit main board including the LAMB cards and the Associative Memory chips.
- Fermilab – the Data Formatter main board and applying 3D silicon chip technology to the next generation Associative Memory chip.
- Argonne – the Read-Out Drivers
- Illinois – the final boards in the core crates that in Option A extrapolate the 7-layer tracks into the stereo SCT layers, carry out the 11-layer fits and pass them through a Hit Warrior; and in Option B that pass the 11-layer tracks from the second stage through the final Hit Warriors.
- Waseda – the Data Formatter mezzanine card that receives and clusters the SCT hits.

5.3 Firmware and software

There are many FPGAs in FTK and consequently a lot of firmware to be written. Each group that designs a board will also write the associated firmware. For each design in previous hardware projects, we have had an engineer work with a student or postdoc. The former focuses on layout, routing, etc., while the physicist writes the firmware and analyzes the board simulation.

We will need software for system initialization, communicating with the ATLAS DAQ, and monitoring the performance of FTK. Each group building a board will write its initialization code to be inserted in an overall FTK initialization module. Argonne, which will build the RODs, will be responsible for communications with the rest of the DAQ system. Pavia will work with Argonne in defining the connection to the level-2 system and writing the basic monitoring code that will run in the FTK monitoring PC, with other institutions providing code needed for monitoring their boards.

5.4 Preliminary cost estimate

We have made a preliminary cost estimate that covers either Option A or Option B. The estimate is based on our previous experience building large trigger systems and on the types of chips that are in our preliminary design. The cost is in the European system, *i.e.* it includes parts and production but not engineering done within our institutions. Since it is a pre-engineering estimate, there is (in the U.S. system) a contingency needed that is not included here. There are reasons to believe that some costs might grow, due to inflation and to changes that come out of

the full engineering design. On the other hand, some costs should decrease. Specifically, the cost of FPGAs of a given complexity decrease with time. For U.S. costs, the assumed conversion is 1€ = $1.50.

The estimated costs below are in Euros. The crates include CPUs and power supplies, and the core crates were chosen to handle 5.5 kW each. Board counts include 10% spares and some in addition for test stands.

The TSP is not included in this production list. Both the TSP and the new AM chip need further R&D and the production costs will depend on the results. For the TSP there is a good chance that it won't be needed for the Phase I SLHC luminosity, and if it is, that won't be until the luminosity exceeds 2×10^{34}. The necessity for the TSP is correlated with the capability of the final AM chip, which will not be needed until the luminosity exceeds 1×10^{34}. According to the summary of the 2010 Chamonix workshop, that will not occur before 2016. The production cost listed here assumes that prices for the 90 nm process will have dropped by the time the new chips are needed, as has been the case with other processes in the past. The AM chip R&D, which will dominate over the final production, is costed below based on our past experience with the CDF AM chip.

The production costs are:

		number		cost
1.	Infrastructure			
	a. Racks	8		16k
	b. Crates (with CPUs and power supplies)	17		180k
	c. ROD to FTK fibers	250		65k
	d. Dual-output HOLAs	250		100k
	e. Intra-FTK fibers and cables	200		20k
			TOTAL:	0.4M
2.	FTK Boards			
	a. DF motherboard	60		160k
	b. DF mezzanine card	180		180k
	c. Processor Unit Auxiliary card (DO/TF/HW)	150		500k
	d. Processor Unit main card	150		200k
	e. LAMB	600		300k

f.	AM chip	19200	100k
g.	HW/TF final board	43	115k
h.	ROD	4	20k
		TOTAL:	1.6M

In addition, there is the cost of prototypes. We have assumed two iterations, which totals 0.1M Euros for everything including a TSP mezzanine, but not the new AM chip.

The R&D and prototyping of the new AM chip will be expensive and involve at least the following steps.

1. We are already producing a 20 k€ mini-ASIC with 90 nm planar technology to evaluate the advantages in terms of area and power consumption from a full custom cell designed for squeezed low power consumption pattern logic.

2. We will need a first 40 k$ 3D prototype with 130 nm technology to evaluated the yield and procedures for producing and stacking four equal tiers with the ability to test each before stacking. The input/output signal handling will be identical on all tiers.

3. We will need a second 40 k$ 3D prototype at 90 nm where we use the experience of steps 1 and 2 to produce on a small area the final logic and mechanical architecture.

After the final design is frozen, there are additional expensive steps which are best carried out shortly before we need the final chips. These steps will become cheaper with time, and until the luminosity exceeds 1×10^{34}, we can use the CDF AM chips which can be produced at very low cost (15 k€ for 3000 chips). For the following steps, we base the costs on our experience with the CDF AM chip. From 2004 to 2009, the price of the 180 nm MPW run dropped by a factor of 2, and the 130 nm technology costs today what the 180 nm process cost then. We expect that in 5 years, the 90 nm process will have a similar cost. After the MPW, we will produce the masks for the pilot run. They are expensive but allow us to use the entire wafer for our project. We paid approximately 200 k€ in 2005 for the 180 nm CDF masks. We expect to pay a similar price for 90 nm masks 5 years from now.

4. We will do an MPW run for a large area chip at a cost of approximately 100 k€.

5. We will produce the masks for the pilot run for roughly 200 k€.

The total cost for the new AM chip prototyping is 0.37 M€.

The total cost including prototypes, final production, and infrastructure is 2.5 M€.

5.5 Schedule

We expect to complete the engineering for the Technical Design Report within two years after approval to proceed to the TDR. The exception is the new AM chip and the TSP which will take longer, but they aren't needed until the LHC luminosity exceeds 1×10^{34} and 2×10^{34} respectively. We can begin with the existing AM chip (see section 6) while we continue the AM and TSP R&D. After approval of the TDR, production, testing, and installation of the system will proceed as rapidly as funding will allow. A technically limited schedule would allow this to be done in approximately 12 months. A possible staging process is described in section 6.

5.6 Maintenance needs

We have many years of experience with the CDF SVT system. The record of needed maintenance is excellent. The reliability of the boards we built is exceptionally good. Typically we have to swap a board with a spare a few times a year, and the original board is then repaired in our shops. In addition, tracking constants are only changed once or twice a year. The maintenance load is very small. Even as the size of our CDF SVT group has shrunk in recent years because of the migration to ATLAS, we have had no problem keeping the system performance very high.

6 Staging

The current plan for the LHC accelerator upgrades was presented at the conclusion of the Chamonix 2010 Workshop. It has the luminosity reaching somewhere between a few times 10^{33} and 1×10^{34} $cm^{-2}s^{-1}$ prior to the Phase I shutdown now scheduled to begin at the end of 2014, which is several years later than had been previously planned. Phase I accelerator operation would commence in 2016, with Phase II operation beginning in 2021. The peak luminosity in Phase I has been reduced from 3×10^{34} $cm^{-2}s^{-1}$ to 2-2.5$\times10^{34}$ $cm^{-2}s^{-1}$, with Phase II operating with luminosity leveling at about 5×10^{34} $cm^{-2}s^{-1}$ rather a peak luminosity of 1×10^{35} $cm^{-2}s^{-1}$ with the usual exponential decrease with store time. Thus the system presented in this document has a luminosity margin for Phase I and a capability not far from that needed in Phase II. In this section, we present a staging plan that would be well matched to the accelerator plan. It also has options to match the eventual funding profile. The actual plan for construction and installation will of course come out of discussions with ATLAS management and the funding agencies.

6.1 Advantages to staging and a vertical slice test

It is important to note that installation of part or all of FTK has virtually no impact on normal ATLAS data taking. HOLA mezzanine cards in silicon RODs are replaced by dual-output HOLAs that have successfully operated in CDF for a number of years. The additional fiber output that goes to FTK has no flow control; thus FTK cannot affect the usual DAQ path. Installation and testing of FTK components with beam is truly parasitic.

We propose to initially install FTK prototype boards to cover a small projective wedge in the detector. Such a "vertical slice test" allows prototypes of all of the FTK boards to be tested in beam-on conditions, provides a full test of the entire FTK data chain, and it encourages early development and testing of the software needed for full operation of FTK within the ATLAS TDAQ environment.

If the funding profile dictates a window between the vertical slice test and full FTK production, we could use the first batch of production boards to expand the size of the vertical slice wedge in the barrel region so that *b*-jet, τ-jet, and single lepton performance can be measured and optimized. Since this would occur after the end of the Tevatron Collider run, the AM chips and LAMB cards currently in CDF could be used. Since the LHC luminosity will still be relatively low, the size of a core crate could be kept small, no more than 4 Processor Units per region.

6.2 Possible early staging

The 2012 long shutdown of the LHC is the natural time to connect a vertical slice crate to the detector so that prototype FTK boards can be tested as they are produced. To be fully prepared for this, we will start in 2010 to assemble a vertical slice in which the major FTK functions are carried out using existing boards as described in section 6.2.1. With this setup, we will be able to test whichever prototype boards are ready first. An early start is also important because we have to develop the tools for connecting to the ATLAS DAQ, including the control and monitoring software.

At the start of the 2012 shutdown, we would replace the HOLAs in a small detector wedge with dual-output HOLAs, run their fibers to the vertical slice rack, and test them with data injected from the RODs. Depending on the duration of the shutdown and the funding profile, we could then install the rest of the dual-output HOLAs and fibers so that the expansion of the vertical-slice wedge and then the installation of the full system would be independent of the accelerator running schedule. We also note that since HOLA parts could become obsolete, it would be better to build the dual-output HOLAs as early as possible.

6.2.1 Modifying an existing board to serve as a low luminosity Data Formatter

In this section we describe how to initially perform the main FTK functions in the vertical slice using boards that already exist.

As noted above, we previously built an early prototype of an FTK Associative Memory board that could be used initially in the vertical slice test. There also exists a board that can feed the AM board with data. The EDRO card was built for the SLIM5 collaboration to be used with the AM board [28]. They communicate through the P3 connector, with the AM board receiving from the EDRO card all of the incoming hits on 6 buses using the LVDS serializer/deserializer chips described in section 2.

The EDRO board is a 9U VME master holding 5 mezzanine cards and capable of an integrated input/output of 30 Gb/s. An annotated image of the board is shown in figure 6.1. The master board provides the clock, power, VME interface, and the network connection to the various mezzanines. A CERN TTCrq mezzanine card is used as a 40 MHz clock source and for EDRO-EDRO synchronization.

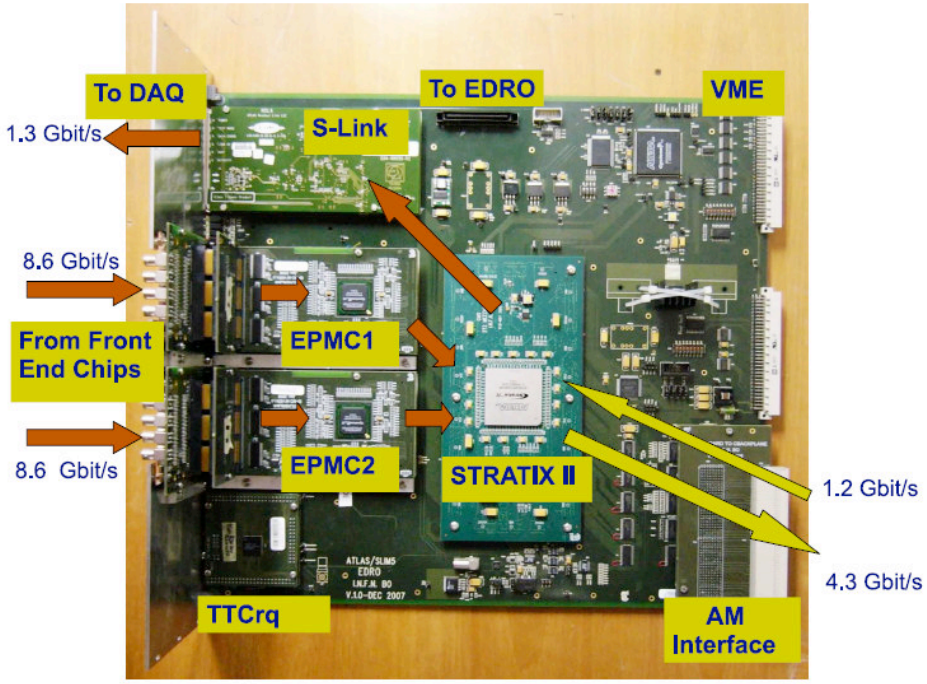

Figure 6.1: The EDRO board that could be used in place of the DF initially in the vertical slice.

Two Programmable Mezzanine Cards (EPMC) are used for communicating to and from the front-end chips. We will replace them with the first prototype Data Formatter mezzanines which will receive hits on HOLA fibers and perform pixel clustering.

The old mezzanines and most of the data paths have been run extensively with a 120 MHz clock, which corresponds to an input/output data rate of 12 Gb/s. The hits collected from the EPMCs are forwarded to the main mezzanine of the EDRO board which holds a large Altera Stratix FPGA. The large number of logical elements (>100 k) and memory (>6 Mb) in the FPGA were exploited in the SLIM project to implement, at a speed of 120 MHz, event building and triggering for the test beam. Hits from the EPMC were forwarded to the AM. Triggered events were stored in local buffers and forwarded to the DAQ PC via a CERN S-Link Link Source Card (LSC). A set of connectors for EDRO-EDRO and EDRO-AM communication and some LEMO connectors complete the board.

We can use this board in the first phase of the vertical slice and write the needed software. We will replace the EPMC mezzanines with DF mezzanine prototypes to receive the data from the dual-output HOLAs, implement the Data Organizer function inside the FPGA, and use the

existing capability to transfer hits to the AM boards and receive back the matched roads. Initially we could transfer the roads and their hits into the monitor PC to perform the linear fits and test the performance of the system.

We also plan to use the vertical slice crate for early technical tests such as the ability to cool a fully-loaded AM card sitting between two other fully loaded AM boards and receiving hits in parallel from the 6 input buses.

The EDRO boards would be replaced by the FTK prototypes as soon as they are available.

6.2.2 Using the existing Associative Memory chips up to 1×10^{34}

There are significant advantages in waiting to produce the new AM chips until they are needed. Even if we make a firm decision to use the 90 nm technology, the price will decrease with time, making the total cost of FTK significantly less expensive than it would be if we built it all today. As noted in section 5.4, this was our experience with the previous AM chips.

The availability of the CDF AM chip at low cost (20 k€ for 3000 chips) makes it an attractive option for running at luminosities up to 1×10^{34} $cm^{-2}s^{-1}$, while we do the R&D needed to prepare for full production of the new chip.

We have begun to study the possibility of using the CDF AM chip at luminosities up to 1×10^{34} $cm^{-2}s^{-1}$. Since the silicon detector occupancy is much lower here than at 3×10^{34} $cm^{-2}s^{-1}$, we can organize the core crates for a single 11-layer pattern recognition stage or the 2-stages of either Option A or Option B. So far, we have investigated the single-stage possibility. In this case, we would use the 12 multifiber channels to transfer the data from the 11 layers to the core crates. The data from the SCT layers would be multiplexed onto 4 buses, making a total of 7 buses carrying data from the AUX board to the AM board.

We will use large superstrip widths to limit the size of the pattern bank since with the existing AM chip we could have up to 5 million patterns per core crate. With superstrips 50 pixels and 128 strips wide in r-ϕ and covering a full silicon detector module length in z, there would be 4 million patterns per region. *WH* events simulated at 1×10^{34} produce 900 roads from each AM board, which can be handled by a simple TSP built on an AUX card mezzanine. The TSP is needed here to reduce the number of candidate tracks to be fit which results from the large road width. By reducing the SCT superstrip width by a factor of 2 and the pixel z width by a factor of 4 in the TSP, there would be 6250 fits per AUX card Track Fitter, which can be handled with the current TF design.

We will also investigate Options A and B for use with the existing AM chips. As for the high luminosity case, this might further reduce the demands on the system.

7 Summary and conclusions

As LHC luminosity rises, the challenges for the trigger will significantly grow. The increase in event size, event complexity, and the rates for both signal and background will require ever more sophisticated algorithms to be run in the level-2 trigger farm. In addition, tracking will become increasing powerful and necessary in trigger selection. FTK will help with both of these problems. By completing global tracking by the start of level-2 processing, more than three orders of magnitude more quickly than can be done in the farm, FTK will enable early rejection of background while maintaining high signal efficiency for *b*-quark and τ-lepton jets. The rapid application of isolation using tracks from the hard scattering interaction but suppressing those from pile-up interactions will keep the single lepton triggers efficient and pure. And the level-2 processor time freed up by FTK can be used to better deal with the complexity of events at high luminosity.

We have presented a system design that works well at luminosities up to 3×10^{34} cm^{-2}s^{-1} and has an upgrade path to even higher luminosities. Following approval to proceed to a Technical Design Report, we will commence a full engineering design aimed at prototypes within 2 years, with the exception of the next generation Associative Memory custom chip whose R&D will continue for an additional few years. At the same time, we will assemble a vertical slice crate using existing boards and write the software to operate it in the TDAQ environment. During the 2012 shutdown we will install dual-output HOLAs on silicon RODs and run fibers from them to the vertical slice. We will then be able to test FTK prototype boards as they are produced and then enlarge the system until it is complete. The schedule for increasing the number of Processor Units per core crate, and thus the spending profile, can be structured to match the LHC luminosity plan.

Until the luminosity exceeds 1×10^{34} cm^{-2}s^{-1}, we can operate FTK using the existing CDF AM chip. That allows us to operate a full FTK system at a lower initial cost while continuing R&D on the most technically challenging and costly component, the AM chip. Producing it later, when process costs have dropped, will reduce the total system cost while allowing FTK to contribute to ATLAS physics as early as possible.

We have a team that is experienced, has strong technical support, and is ready to complete the design, build FTK, and operate it to enhance the ATLAS physics program.

Glossary

AM: Associative Memory – The content addressable memory that holds the array of track roads, each of which is tested in parallel as the silicon hits flow from the detector.

AM Map: provides the starting addresses in the Hit List Memory for the hits to be sent out of the Data Organizer for a matched road.

CAM: Content Addressable Memory – the technology basis of the AM chip.

Core Crate: a crate of Processor Units covering an azimuthal section of the inner detector

Coverage: characterizes a pattern bank. It is the fraction of single particles above the FTK minimum P_T that passes through all superstrips in a road or all but one superstrip. It does not apply other requirements to tracks, like passing a χ^2 test, which are part of the efficiency measure.

DF: Data Formatter –receives the pixel and SCT data, does clustering, and sends the hits to the appropriate Processor Units.

DO: Data Organizer – stores the full resolution hits for easy retrieval by road number and sends courser hit information to the Associative Memory.

EE: End Event – A bit on a data path signifying the end of the data stream for the current event.

EP: End Packet – If data is sent on a data path in packets, this bit indicates the last word in the packet.

FC: Fit Constants – the constants used in the linear approximation in the Track Fitters.

HLM: Hit List Memory – stores hits from the current event in the Data Organizer.

HOLA: The CERN mezzanine card that converts electrical signals into S-Link optical data.

HOLD: A signal on a data path informing the data source that the destination buffer is almost full.

HW: Hit Warrior – removes duplicate tracks.

LAMB: A mezzanine card on the Associative Memory board that holds the AM chips.

Option A: a two stage architecture in which tracks are first found in the pixels and axial SCT layers, and then in the stereo SCT layers.

Option B: a two stage architecture in which tracks are first found in the SCT and then in the pixel layers.

Processor Unit: a board and AUX card sitting in a single slot in the core crate and containing the AM, DO, TF, and HW.

Region: the range in azimuth covered by one core crate

RW: Road Warrior – removes all but one matched road containing identical hit superstrips.

Sector: one silicon module in each layer defining a region covered by one set of fit constants.

SS: SuperStrip – The course hit bin that is used in the Associative Memory for pattern matching.

SS Map: provides the starting address for writing hits for a superstrip into the Hit List Memory.

TF: Track Fitter –finds the track helix parameters and goodness of fit using a linear approximation.

TSP: Tree Search Processor – can be used as a second stage in pattern recognition, following the Associative Memory, to reduce the width of roads and decrease the rate of fake matched roads.

WEN: Write Enable - A signal on a data path indicating the presence of a data word to be read on the next clock cycle.

References

[1] M. Dell'Orso, L. Ristori, "*VLSI Structure For Track Finding ",* Nucl. Instr. and Meth. A 278 (1989) 436.

[2] J. Adelman et al., "*The Road Warrior for the CDF online Silicon Vertex Tracker*", IEEE Trans. Nucl. Sci., **53**, (2006) 648 – 652.

[3] A. Annovi et al., "*Hadron Collider Triggers with High-Quality Tracking at Very High Event Rates*" IEEE Trans. Nucl. Sci., **51**, (2004) 391

[4] J. Adelman et al., "*On-line tracking processors at hadron colliders: The SVT experience at CDF II and beyond",* Nucl. Instrum. Meth. A581, (2007) 473.

[5] J. Adelman et al., *"The Silicon Vertex Trigger upgrade at CDF",* Nucl. Instr. and Meth. A 572 (2007) 361–364.

[6] G. Aad et al., "*The ATLAS Experiment at the CERN Large Hadron Collider*", JINST 3, (2008) S08003.

[7] A. Annovi and M. Beretta, "*A Fast General-Purpose Clustering Algorithm Based on FPGAs for High-Throughput Data Processing*", submitted to NIMA, arXiv:0910.2572v1.

[8] G. Batignani et al., *"The Associative Memory for the Self-Triggered SLIM5 Silicon Telescope*", NSS Conference Record, (2008). 2765 – 2769

[9] http://www.xilinx.com/, http://www.xilinx.com /support/documentation/user_guides/ug388.pd

[10] A. Annovi et al., "*A Pipeline of Associative Memory Boards for Track Finding",* IEEE Trans. Nucl. Sci., **48**, (2001), 595.

[11] A. Annovi et al., "*A VLSI Processor for Fast Track Finding Based on Content Addressable Memories*", IEEE Trans. Nucl. Sci. **53**, (2006), 2428.

[12] P. Battaiotto et al., "*The Tree-Search Processor for Real Time Track Pattern Recognition*", *Nucl. Intsr. and Meth.*, vol. A287, pp. 431-435, 1990.

[13] P. Battaiotto et al., "*A Fast Track Finder for Triggering Applications in High Energy Physics*", *Nucl. Intsr. and Meth.*, vol. A293, pp. 531-536, 1990.

[14] M. Dell'Orso and L. Ristori, "*A highly parallel algorithm for track finding*", *Nucl. Intsr. and Meth.*, **A287**, (1990) 436-440.

[15] http://www.pi.infn.it/~paola/TSP_v14.pdf.

[16] S. Belforte et al., "*The SVT Hit Buffer*", IEEE Trans. Nucl. Sci., **43**, (1996), 1810.

[17] A. Annovi et al., "*The Data Organizer: a High Traffic Node for Tracking Detector Data*", IEEE Trans. Nucl. Sci. **48**, (2001), 1313

[18] F. Crescioli et al. "*Linear Fit within Missing-Hit Roads in FTKSim*", Internal Note.

[19] S. Amerio et al, "*The GigaFitter for fast track finding based on FPGA DSP Arrays*", IEEE 2007 Nucl. Sci. Symp. Conf. Rec. **3**, (2007), 2115.

[20] SVTSIM. http://www-cdf.fnal.gov/upgrades/daq trig/trigger/svt/svtsim.html.

[21] S. Belforte et al. Svt TDR (silicon vertex tracker technical design report. CDF/DOC/TRIGGER/PUBLIC/3108.

[22] The ATLAS Collaboration. ATLAS Computing Technical Design Report, July 2004. ATLAS TDR-017, CERN-LHCC-2005-022.

[23] ATHENA. http://atlas.web.cern.ch/atlas/groups/software/oo/architecture/.

[24] ROOT Object Oriented Data Analysis Framework. http://root.cern.ch/.

[25] H. Wind, "Principal component analysis and its applications to track finding," in Formulae and methods in experimental data evaluation, vol. III, R. Bock, K. Bos, S. Brandt, J. Myrheim, M. Regler, Eds. European Physical Society, 1984, pp. k1-k16.

[26] R. Carosi and G. Punzi, "*An algorithm for a real time track fitter*," Conference Records of the 1998 IEEE Nuclear Science Symposium, Toronto, Canada, Nov. 1998.

[27] ATL-COM-INDET-2009-055.

[28] G. Batignani *et al.*, "*The associative memory for the self-triggered SLIM5 silicon telescope"*, Nuclear Science Symposium Conference Record, 2008. NSS '08 IEEE http://ieeexplore.ieee.org/xpl/RecentCon.jsp?punumber=4747668, pp. 2765-2769. Also see http://www.pi.infn.it/%7Eorso/ftk/level1/AM@L1.pdf.