

Field programmable gate arrays (FPGAs) excel at doing repetitive high-speed operations on streams of data. For some tasks, they can outperform microprocessors by factors of 10 to 100. The BTeV trigger group is currently considering two solutions using FPGAs to implement tracking in the L1 Pixel trigger. While these schemes use different processing algorithms, they both generate similar data which consists of pixel coordinates tagged by track identifiers. They also have similar output data rates of tens of gigabytes per second. (The average data from the pixel detector is expected to be 41 Gbyte/Sec. Various FPGA tracker implementations convert this to 21, 33, 94, or 135 Gbytes/Sec.) Since the outputs are all similar, it is possible to propose a processing strategy that will accommodate both FPGA front end proposals.

Three tactics are used to deal with the high data rates. First the data flow is divided geographically. Since pixel hits come from the detector by quadrant, it is natural to keep this division throughout the processing network until the vertex processing stage.

Second, events are sorted into 8 time slices by timestamp. This division is maintained up to the final trigger output.

Third, processing is divided into two stages. The first stage processes tracks. It reduces the amount of data per event by at least a factor of 4 so that the output of all 4 quadrants can be combined in the vertex processing stage.

The vertex processing stage produces L1 triggers. Triggers from all 8 time slices are merged and passed to the L2 trigger system.

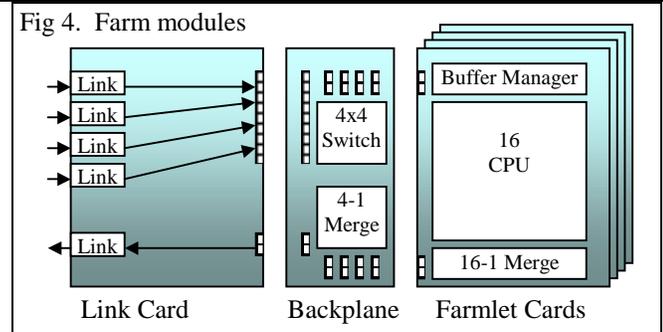
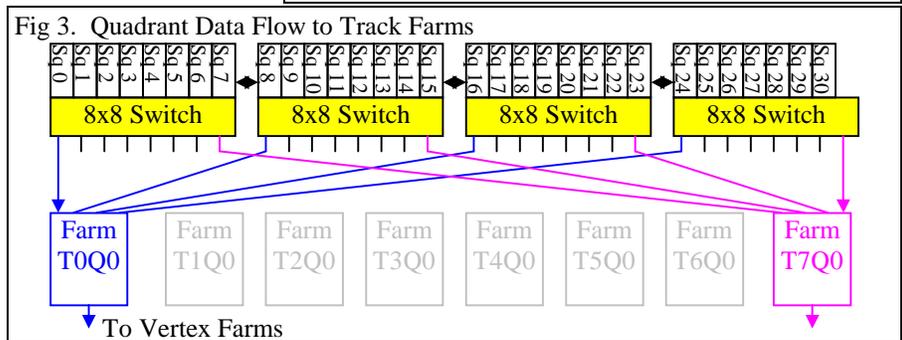
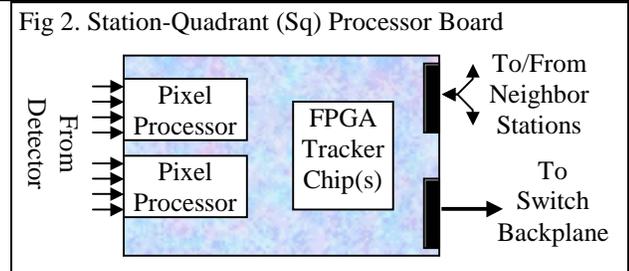
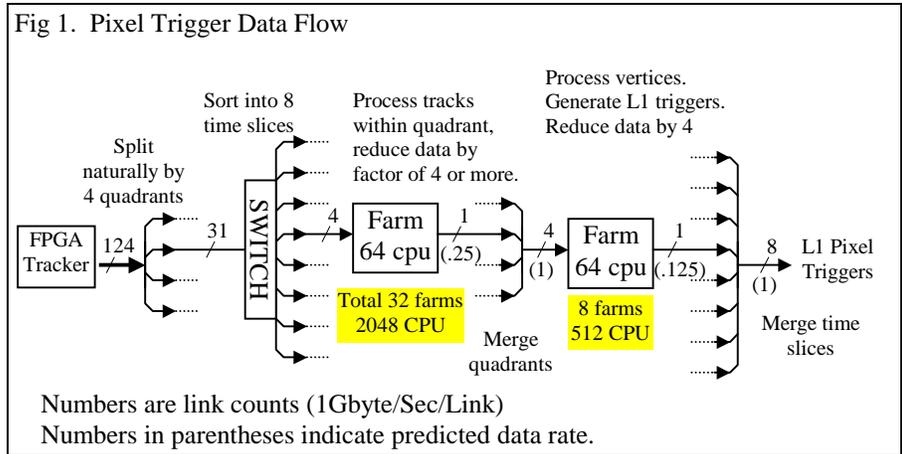
Implementation

A single board processes the detector output for each station-quadrant (Sq). In addition to the FPGA tracker chips, an Sq board contains two pixel processors (for two pixel planes per station) which receive and format the raw pixel data from the detector. The FPGA tracker chips do their magic and send their result to a switching backplane.

The 31 SQ processors in a quadrant are packaged into 4 independent 8 x 8 switching backplanes. Each backplane sorts the output of 8 stations into 8 time slices, and sends it to 8 Track Farms.

A track farm receives the same time slice from all four backplanes. It is implemented as a 4 x 4 switching backplane that sorts data from 4 input links to 4 "Farmlet" cards. Each Farmlet has 16 processors which operate independently on events. The output of all farmlet cards is merged on the backplane and sent to the vertex farm via a single link.

There is one vertex farm for each time slice. A vertex farm receives tracks from all four quadrants, finds vertices, and produces a trigger. The output from all 8 vertex farms must be merged together. This can be done by the global L1 trigger hardware, or it can be done by a special piece of hardware that merges 8 links into 1.



Algorithm-Specific Implementation

Figures 5 and 6 show station-quadrant data flow for two proposed FPGA trackers. The bus widths are given as the number of bits necessary to carry information for 1 pixel hit or 1 track. Each bus would have to cycle at 42 MHz to keep up with an average rate of 6 tracks per event per station. Cycling at 50 MHz would give a small safety margin.

For the Associative Memory scheme, a 76-bit track token passes from station to station. 50 bits of the track token are stripped off and sent to the farms. This delivers a global average of 33 Gigabyte/Sec to the Farms. A variation of the associative memory scheme that only delivers track-ends to the farm would produce 21 GB/S.

Fig 6 shows flow for the CMU algorithm with 2 correlators. It must share hit data between stations in addition to sending 144 bits per cycle to the farms. It delivers, on average, 94 GB/S to the farm. A variation which uses 4 correlators would send an extra 64 bits per cycle, or a total of 135 GB/S. If only 2 correlators are used, the packaging can be optimized by keeping X view and Y view data separated and putting multiple (2-4) SQ processors on a single card.

Implementation of the links and switching backplanes scales, at best, with bandwidth. A network that supports 94 gigbytes/second will cost at least 3 times as much as one that supports 33 gigbytes/sec.

Costs

The table is a system cost summary for both FPGA tracker front ends. Costs are in thousands of dollars and include PC-board and parts costs.

Qty	Description	Associative		CMU	
		Unit	Cost	Unit	Cost
124	SQ processor boards including 8 input links.	2.5	310.0	3.4	421.6
16	Switched 8x8 backplanes	2.0	32.0	3.8	60.8
12	Links between backplanes	0.2	2.4	0.4	4.8
128	Links to Track Farms	0.2	25.6	0.6	76.8
32	Track Farms	15.0	480.0	16.0	512.0
32	Links to Vertex Farms	0.2	6.4	0.6	19.2
8	Vertex Farms	15.0	120.0	16.0	128.0
Total			976.4		1223.2

Fig 5. Associative Memory Data Flow

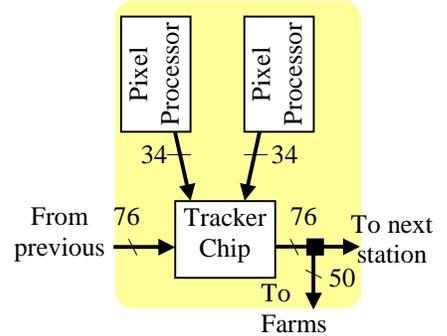
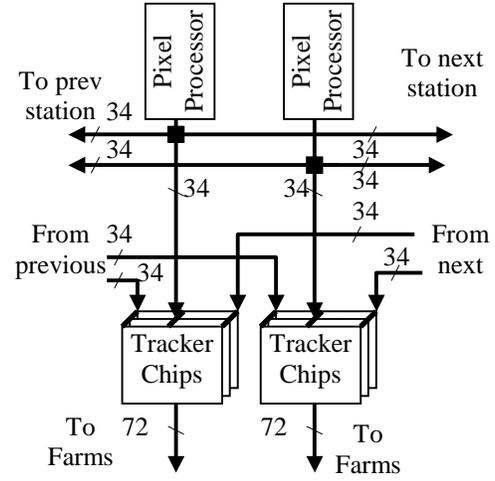


Fig 6. CMU Data flow



Performance

The table below summarizes the performance of the two FPGA tracker algorithms. Several measures are considered:

Array Size

This is the number of logic blocks to implement the tracking function. For the associative tracker, it takes 16 CLB to implement each associative memory cell. An array capable of processing 16 plane hits (maximum) requires 16 associative cells. For the CMU tracker, each CLB can do a 1x1 correlation for 1 view. It takes 16x16 CLB to implement a correlator capable of processing 16 hits (maximum).

Cycle Counts, Clock rate

This is the number of cycles required to process 1 hit. Since the CMU tracker uses 12-bit serial arithmetic, it requires 12 cycles to process each hit. In principal, serial arithmetic allows higher clock rates, but in reality, the clock rate is limited by the speed of longlines that are required

	Associative Memory	CMU
Array Size	16x16 Virtex CLB	16x16 Virtex CLB
Cycle Counts	N	12*N (Serial Arithmetic)
6 Hits	6	72
10 Hits	10	120
16 Hits	16	192
Clock Rate	50 MHz	100 MHz or more
Throughput	20 ns/Hit/Quadrant sustained	120 ns/H/Q per array/view
Resolution	16 bits	12 bits
Arrays/SQ	2	16
Chips/SQ	1 (XV300 equivalent)	5? (XV400)
Links/ SQ	3	8
Output Format	50 bits (T,X,Y,ID,M)	144 bits 2*(T,X,Y,Mf,Mr)
Output to Farms	33 Gbyte/Sec (50*124*.042/8)	94 Gbyte/Sec
Expandability	Linear with Hits/Plane	N^3 with Hits/Plane

to broadcast data to a large number of cells. The CMU array must be cycled at 100 MHz or higher to get the desired throughput. This is near the chip limit. The associative array is cycled at 50MHz, however an array has been demonstrated that runs at 100 MHz.

Throughput

At 100 MHz a CMU array can process a hit in 120ns. To achieve a higher throughput, several arrays must be run in parallel. It will take at least 8 arrays to achieve 20ns/hit throughput. Even if 80% of all events have 8 or fewer hits, the average throughput will be 144ns per event. (.8*120ns + .2*240ns). Running arrays in parallel creates further problems of how to split and re-combine data flow. The associative tracker can process a hit per cycle. There is a small fragmentation effect that occurs when very short events are followed by very long events, however this effect can be kept very small with good pipelining.

Resolution

12 to 16 bits is adequate. 16 bits over a 5cm plane represents a resolution of 2 microns. Decreasing the resolution from 16 to 12 would shrink the associative tracker array to 12x16 CLB. Increasing the CMU tracker resolution to 16 would increase processing time to 160ns per hit.

Arrays/SQ, Chips/SQ

For a station-quadrant, the associative tracker requires 1 array for finding doublets, and 1 array for matching doublets to tracks. Two arrays and their support circuitry can probably be put into a small XV300 Xilinx chip. The CMU tracker requires 8 arrays per view, or a total of 16 arrays. Even if 4 CMU arrays can be fit into an XV400 chip, four of these chips will be required, plus an additional chip to do data splitting, recombination, B-plane alignment.

Links/SQ

This is the number of links that must pass the SQ board boundary.

Output Format

Output format assumes 15-bits for X and Y and a 10-bit time stamp added for routing each hit. Five of the T bits will be replaced by the station ID when the word passes through the switching network. Note that for CMU scheme, we assume some circuitry to keep (Xb,Yb) and (Mf,Mr) aligned, otherwise each must be tagged with their own T. Also note that the CMU scheme assumes that the switch transmits hits in order. Otherwise data must be tagged with a 4-bit hit index. The output word width affects the number of pins per chip, and therefore scalability.

Expandability

The size of each CMU array grows as the square of the Maximum Hits-per-Quadrant (HPQ_{max}^2). Since the arrays are run at their maximum clock rate, the number of arrays required increases linearly with HPQ_{avg} . The current CMU plan assumes $HPQ_{max}=16$ and $HPQ_{avg}=6$. Also note that the size of the output format depends on HPQ_{max} ($80+4*HPQ_{max}$). This will affect bus widths, connector sizes, and packet sizes for the entire output path, making it difficult to make changes once hardware is built.

The size of each associative array grows linearly with HPQ_{max} . A larger HPQ_{ave} can be accommodated by increasing the clock from 50MHz up to 100MHz. The output bus width does not grow to accommodate higher hit rates.