Draft, Version 4
April 17, 1997

Conceptual Design of the SVT Hit Finder Board

J. Berryhill, R. Culbertson, H. Frisch, M. Shochet, P. Wilson

with the SVT and Silicon Readout Groups

1 Introduction

In this note we specify the algorithms of the Hit Finder board of the SVT. After an L1 accept, this board receives the SVX data from the FIB. It forms clusters of hit strips and computes the centroids of the clusters. Through a front panel connector, the Hit Finder passes the cluster information to a merger which passes it on to the Associative Memory Sequencer.

In this note we describe the algorithm that decodes the input stream, forms the clusters, and constructs the output stream. Since the design of the board is underway, this note is preliminary. This note builds on information in reference [1].

1.1 Overview

As explained in reference [1], each Hit Finder will receive data from a G-link Receiver Transition Module (GRT) [2] card containing four G-links running in 20-bit mode at approximately 53MHz. This is the data from two SVX wedges, each with five layers for a total of 10 8-bit data streams or 2 1/2 streams per G-link. See reference [1] for a specification of the data stream format. The G-link output will be passed in parallel through P3 to the main hit finder board.

The Hit Finder functions can be broken down into four steps which we call: alignment, preprocessor, engine, and output.

Since the G-links run asynchronously, we cannot rely on the data appearing on all the G-links at the same time. (An estimate of this skew is < 20ns.) Because of this skew, the two 4-bit half-streams that arrive on two different G-links will have to be reunited and aligned.

The main purpose of the preprocessor is to subtract pedestals from the ADC data. The FIB will also subtract pedestals, but we repeat the function here in case the FIB subtraction is not used or not suitable for the SVT. There will also be the ability to suppress individual bad channels. If a channel is explicitly suppressed or the pedestal subtraction results in an ADC count under threshold, it will not be output from the preprocessor. There will be one threshold each for the 10 data streams on the board.

The engine is the heart of the hit finder. The data stream arrives here in order of strip number so that adjacent hit strips will be adjacent in the data stream. The engine will search for clusters of adjacent hit strips, surrounded by non-hit strips or perhaps a high pulse height strip surrounded by other lower pulse height strips. The engine computes the cluster centroid: $(\Sigma Q_i x_i)/(\Sigma Q_i)$ for each cluster, where Q_i is the charge on a strip and x_i is the strip number; i runs over the strips in the cluster.

The output stage combines the outputs from the engines into one stream which is transmitted on a flat ribbon cable on the front panel. A top level diagram for the hit finder is shown in Figure 1.

2 Alignment

The input to the Hit Finder board is four 20-bit data streams each containing 2 1/2 independent 8-bit data streams. This leaves two pairs of half-streams that need to be reunited. Since the clocks between the G-link may be skewed, the half-streams will have to be aligned by running each half-stream into a separate FIFO, then clocking the output of the FIFOs together as one data stream.

3 Preprocessor

The preprocessor has two functions. It recognizes both the first byte of an event and imbedded flag bit patterns, and takes action based on the value of the flag bytes or possibly based on the position of the flag in a sequence. The second function is to subtract pedestals, channel by channel, from the ADC data in the input stream and suppress channels that are below threshold.

Each sequential pair of bytes are related: stream ID followed by a second stream ID, chip ID followed by chip status, and channel number followed by ADC data. All flag bytes will be even-numbered bytes, 0, 2, 4,...

The four ID bytes at the beginning of an event can be ignored in terms of the Hit Finder functionality. They may eventually be checked for error conditions.

The identification of a hit strip on input is stored in the physical data stream it is in, the chip ID which proceeds it, and its channel number. Which stream the strip appears in identifies the ϕ, barrel, and layer number of the data. Each ϕ slice of the SVX is processed in separate SVT hardware so we do not need to track the wedge number in the Hit Finder, but we do need to track the barrel and layer. The equivalent information is three bits of z (to cover barrel zero to five) and three bits of layer (to cover layer zero to four). In addition, each chip has 128 channels (seven bits) and there are up to seven ϕ chips in a data stream. Here the information is three more bits (to cover chip number zero to six).

Adding it up, 7 bits are needed to identify the hit within a chip, 10 bits to identify it within a ladder, 13 to identify it within a wedge, and 16 to identify it within a complete ϕ slice of the detector. So input stream number, chip ID, and seven bits of channel number is equivalent to 16 bits of information.

This is important because the input streams will be merged, losing the stream position information. The chip ID will also be stripped, losing that encoded information. The Hit Finder algorithm will add bits to the data stream as the encoded information is dropped. Note that the AM system requires the layer and barrel numbers to be in distinct bit fields so we can not encode this information in fewer bits.

Here is a description of what happens in one stream. At the beginning of an event, a 3-bit register (call it CR for Chip Register) has been loaded with -1. The first four bytes of the data stream are header words and are not transferred to the output. Since even and odd-numbered bytes have different meanings (usually channel number and ADC), they are demultiplexed into two separate 8-bit streams, clocking at 26MHz. The streams are called the channel stream (even bytes) and ADC stream (odd bytes). The demultiplexing may be implemented on the preprocessor programmable chip or on a dedicated demultiplexing chip.

When the preprocessor finds a chip ID bit pattern in the channel-stream it adds 1 to CR. In subsequent bytes that contain channel numbers we take the 7 LSB from the channel stream and attach the value of CR as the MSB, creating a 10-bit strip number that uniquely identifies the hit within the ladder. Table 1 shows what the data stream looks like at this stage in the preprocessor.

When the preprocessor system finds a special flag bit pattern on an even-numbered byte (the first z chip ID), it stops and signals an end-of-event on the output. In the current design, only 1 of the 8 ADC bits and chip number 7 is not used and it would be possible to use these patterns to signal the EOR. To be conservative, we allow for the full possible range on the data fields, and another method is used for the EOR.

7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
chip #			channel							ADC							
chip #			channel							ADC							
								⋮									
← strip # →																	

Table 1: An example data stream after demultiplexing and adding chip number to the channel number.

The ID words, chip ID's and chip status' are being dropped here but there is more that could be done with them for error checking. In all, more than 15% of the words in a data stream are predictable. We could also check that channel numbers are strictly increasing and that ADC values are less than 8 bits and non-zero. If this checking is implemented, it would be a significant check for bit errors in the data stream. (The G-link itself makes no check for bit errors.) At a lower rate this checking will be done with monitoring.

The main function of the preprocessor is to subtract pedestals from the ADC data and suppress channels that are not over threshold. The pedestal subtractions must be done on a channel-by-channel basis. There must be a mechanism to suppress broken channels, maybe by subtracting a pedestal corresponding to the maximum ADC count. The ADC data will probably use only the seven LSB of the 8 bits available but the 8-bit possibility will be accommodated.

The preprocessor looks up the pedestal based on the wedge strip number, bits 8 to 17, and subtracts the pedestal from the ADC data, bits 0 to 7 in Table 1. If the result is less than less than the programmed threshold, then it does not transmit this data word. If the strip survives this test, the subtracted value of the ADC replaces the raw value in bits 0 to 7. The pedestal is an integer; subtraction of fractional ADC counts is not implemented. The largest layer that can be accommodated is 8 chips or 1024 strips so the memory storing the pedestals needs to be eight bits by 1024. There is one threshold for each data stream.

Currently, portions of the SVT are designed to time-out on an event, or even sense when an event is too big before it does significant processing. When this happens, the event will be truncated. The FIB, the source of the data, will have a register containing a byte count limit. If the the data stream is longer than the limit, the FIB will truncate the stream. The hit finder needs this capability on its output.

4 Engine

The engine searches the data streams for adjacent hit strips, forms clusters of strips, and computes the charge weighted centroid of the cluster. At this time we favor including one engine for each data stream. The engines would run synchronously with the preprocessor output which is 26MHz maximum but possibly much slower due to sparsification of the data stream.

If we were to chose to have fewer engines than data streams, there would need to be several modifications including adding FIFO buffering between the preprocessors and the the engines and adding stream ID bits at the engine stage. The effect of fewer engines on the timing in the board is investigated in Section 7.

Reference [5] uses SVX' data to analyze several simple cluster algorithms, two will implemented initially. In both cases the output is a 14-bit centroid that specifies the cluster uniquely within the ladder. Since 10 bits specifies a strip, the other 4 bits specify the centroid within a strip. This gives a granularity of pitch/16 or a resolution of 1μm for a 60μm pitch. This granularity resolution is much smaller than the cluster measurement resolution of 10μm or more. The centroid has a range between the lowest strip number, 0, and the highest strip number for the ladder.

4.1 Mixed Algorithm

This algorithm attempts to mimic the offline clustering algorithm. All the strips outp[ut from the preprocessor are considered good and can be used in clusters. The first step is to locate groups of adjacent strips by checking for sequential strip numbers. A group of adjacent strips can not be longer than 6 strips. If there are more than 6 adjacent strips then the first 6 become a group, then the next 6, etc.

What happens next depends on whether a group is short (3 or fewer strips) or long (4 to 6 strips). For short clusters the most significant 5 bits of the ADC's create an address for a memory. The memory contains the centroid indicated by this ADC pattern, that is, it is computing $\Sigma Q_i x_i/\Sigma Q_i$. The memory must be 8 bits by 32K.

The implementation is that 5 bits from the memory output become the offset from one of the strips in the cluster to the centroid. The appropriate strip number is shifted 4 bits to the left and the offset is added to make a 14-bit centroid which specifies the cluster uniquely within the ladder. The full range of the 5 bits are not needed, only values up to 16, which corresponds to a full strip offset. A 6th bit from the memory indicates whether the offset should be added or subtracted from the strip number.

Note that since only the msb of the ADC are used, it is possible for the memory to see all zeros on its address. In this case the returned offset is zero. This puts the centroid at the second strip of a 2-strip cluster and the center strip of a 3-strip cluster.

While this lookup is occurring, the full ADC values are being summed for this group of adjacent strips. Before a group is allowed to be output as a cluster, it must pass a programmable threshold on this total charge. A typical cut might be 13 counts (1/2 of a MIP).

The other case occurs when the group contains 4 or more strips. In this case the centroid is put at the geometric center of the group of strips, $\Sigma x_i/\Sigma i$. The cluster is also required to pass the total charge cut.

Attempting to calculate a centroid for a long cluster does not improve the resolution. The point is that the cluster must be there for the efficiency of the pattern recognition which will require a minimum number of hits. However, it does not contribute to the knowledge of the track parameters. If the cluster is long, the long bit is set in the output word.

In order to effectively use this algorithm, the track fitter must be able to recognize that there are clusters which have good resolution (short ones) and clusters with very poor resolution (long ones). It is not clear at this time if the track fitter will have this capability.

4.2 Peaks Algorithm

This algorithm searches for peaks in adjacent strips. For example an ADC pattern of three adjacent strips of low-high-low would produce one cluster. A pattern of high-low-high would produce two clusters.

Consider three consecutive strips in a data stream, called A, B, and C. The 5 most significant bits from each of their ADC values are placed on the address lines of a memory. Strip A is routed to the LSB of the address, strip C to the MSB. If strip A is not adjacent to strip B on the detector, the ADC value of strip A is blocked from the RAM address so the RAM sees these bits as set to zero. If strip C is not adjacent to strip B on the detector, its ADC is also blocked from going to the RAM address.

The RAM returns one bit set on if this a valid cluster pattern, a peak around strip B. It also returns 4 bits which serve as an offset from strip B. The fifth bit indicating an offset of one full strip is not needed due to the peaks algorithm. The 6th bit is used to indicate if the offset is positive or negative. The offset range is plus or minus 15/16 of one strip.

There are a few important characteristics of this algorithm which should be noted. First, the peak requirement is integral. If it were relaxed, we would get many more clusters than we want. For example if we had two strips, with the pattern high-low, the peaks requirement selects the valid cluster to be when high is position B and low is in position C. It returns with no valid cluster when the high strip is in position A and the low strip is in position B. If this peak requirement were relaxed, both positions would look like valid clusters.

A 4-strip cluster with the pattern low-high-low-high causes a cluster to be made from the first three strips (low-high-low) and a second cluster to be made from the 3rd and 4th together (low-high). Lastly, a decision has to be made for the case where two adjacent strips have the same pulse height.

4.3 End Event

The engine is implemented as pipeline which means the data must come from the preprocessor in a form where adjacent strips are in adjacent clock cycles. This also means that the engine pipeline needs to be cleaned out when the end of event comes. The preprocessor has to continue to clock the engine after end of event.

The output from the engine is pushed into a FIFO. there are 14 centroid bits and one long cluster bit. In addition, the end of event must be signaled from the preprocessor to the engine and passed out of the engine into the FIFO. Table 2 shows the data stream that is output from the engine.

5 Output

At this point the clusters from the 10 streams are sitting in 10 FIFOs. The output controller needs to empty these FIFO's and add stream ID bits to the words. The words may be popped from the FIFO's in any order. When an end of event is found in one of the FIFO's, that FIFO is ignored until the end of event is found on all 10 FIFO's. At that point, the output controller produces and end of event record itself.

bit	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
use	EE	data														
clus		LC	centroid													
clus		LC	centroid													
⋮																
clus		LC	centroid													
end	EE	empty														

Table 2: An example data stream out of a engine into the FIFO. EE=End of Event.

The information appended to the FIFO data is 3 bits of layer number and 3 bits of z number. These are different for each FIFO and are programmable. The entire data word should be routed through the output controller for flexibility.

The end of event word generated by the output controller contains error flags, a parity bit for the event, and the bunch number which are to be designed.

The output controller drives the data on the front panel connector. The output stream is shown in Table 3. The signals are LVDS and the protocol is described in detail in reference [1] and [6].

bit	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
use	AF	DS	EE	EP	data																				
clus				EP	layer			z			LC	centroid													
clus				EP	layer			z			LC	centroid													
⋮																									
clus				EP	layer			z			LC	centroid													
clus				EP	layer			z			LC	centroid													
end			EE	EP	error flags												PA	event tag							

Table 3: An example data stream of the hit finder to merger interface. AF=Almost Full, DS=Data Strobe, EE=End of Event, EP=End of Packet, PA=PArity, LC=Long Cluster.

6 Spy Buffers

The SPY buffers record the recent history of all the data going into or leaving the board. There is one buffer for each input stream, and one on the output stream. The buffers are RAMs that cycle from the lowest address to highest continuously, writing the data that is passing through the stream.

The cycling can be stopped two ways. The first is when an error occurs on board. The second is when a SVT backplane freeze line is asserted. The freeze line may be triggered by an error detected on another board in the SVT or randomly for monitoring. After a freeze, the Spy buffers are read out over VME.

The input buffers are 64K by 16 bits and are filled by the preprocessors with the data immediately after it is demultiplexed. The output buffer is 64K by 23 bits and is filled by the output controller as soon as the data is sent to the output connector.

7 Engine Design Based on Timing

In the baseline design (ref [3]), the hit finder must be finished transmitting all its output 5.75μs after a L1 accept although this may be relaxed to as much as 8μs. The SVX-III digitization will take 1.25μs and readout overhead will take about 1.75μs for a total of 3μs. This leaves the hit finder anywhere from 2.75μs to 5.0μs to complete an event from the arrival of the first data on the input to the last cluster output. L1 accepts are expected at the rate of 50KHz or every 20μs, Poisson distributed.

With the tuned Monte Carlo of reference [4] we can begin to understand the requirements of the hit finder engine. The Monte Carlo is run with reasonable assumptions. The readout threshold is 3σ, the S/N is set to 16, nearest neighbors are read out, there are 2.7 min-bias per event (Poisson distributed), and there is a 2-track trigger imposed.

The Monte Carlo is fed into a hit finder simulation program. The bank containing the SVX data is arranged in a form that resembles the data stream and ID words are inserted. All the data streams are clocked through preprocessors. The preprocessor suppresses channels that are not over a 4σ threshold and the output time for each surviving strip is recorded. We assume the preprocessor runs at a 53MHz input rate, the G-link speed. To accomodate the possibility that there is less than one engine for each stream, a merge stage is also simulated. The engine, including actual clustering, is simulated. Each cluster is output at the time all its strips have been input. Again the time is recorded. The final merge of the engine outputs into a single stream then the merge of three hit finders in to a single stream is simulated and the transmission times recorded.

For each event we can record the time that the last cluster was sent to the associative memory board. Figure 2 shows the average time between the first data arriving and the last cluster output for the average and the slowest AM system in the event. (An AM system consists of three hit finders transmitting data to one Associative Memory Sequencer; this is 1/12 of the system, one complete ϕ sector.) The plot is the time *vs* the assumed engine input speed in MHz. Here we are using only the first 4 inner layers; we ignore the fifth layer.

The three curves are three different assumptions: 6 engines per ϕ slice (2 per hit finder board), 6 engines per ϕ slice (wedges mixed), 12 engines per ϕ slice, and 24 engines per ϕ slice. Wedges mixed means that not all of the layers from one wedge go to the same engine. Half of the layers from the wedge go to one engine, half to the other. This helps because a very highly occupied wedge will be shared between two engines. The mixing is barrel 0 and 1, 2 and 3, and 4 and 5. We found that mixing 0 and 3, 1 and 4, and 2 and 5 didn't improve things any more.

The point marked "baseline" is the case where six engines input at 10MHz and output at 5MHz. If all six engines worked continuously for the same length of time, this would saturate the 30MHz rate that clusters can be merged in the final stage and transmitted to the AM. Clearly the full simulation shows that we are far from this ideal. This is due to fluctuations in the occupancy distribution - frequently there are highly occupied wedges with and less-occupied wedges. The busier engines output at 5MHz for a long time while the less-busy ones are idle.

With large programmable chips available it may be simple to include one engine for each input stream (24 engines per AM system). In this case, the 30MHz input to the AM system is easily saturated.

Figure 3 shows the same analysis for the 12 and 24-engine cases but now the AM system is running at 50MHz. We see the AM system is saturated with the 12-engine design if the engines can be run at more than 30MHz, or the 24-engine design if they are running at more than 20MHz. The Hit Finder can be done with its job 40the AM can receive data at 50MHz.

Figure 4 shows the actual distribution of times for all AM systems and the AM system with the longest time in the event. The assumption is six engines, wedges mixed, 30MHz engines.

8 Some Failure Modes

There are three possible timing failure modes identified so far, they can occur individually or together.

The first is "input too long". This can occur on an event regardless of previous or following events. It is caused by too many strips read out. This could be an unusual event or or an unusual circumstance such as the thresholds on the SVX-III chips being set too low. The error will occur if more than approximately 80-90 strips are read out from one ϕ layer. The worst case would be all strips of the seven ϕ chips on layer four are read out which would take $36\mu s$ to transmit on the G-link (and another 36μ to complete the z readout). The event could take significantly longer to get out of the merge stage. When this error occurs, the downstream components of the SVT would sit and wait for the hit finders to finish. L2 would then wait for the SVT result, creating deadtime if necessary. Note that the hit finder should also have the capability of truncating its output if a long event is encountered.

The companion error is "output too long". Since 15 input streams are compressed into one output stream there is the possibility of the input being in time, but the output causing a major delay. The input will not be the limiting factor if less than 80-90 channels are on a stream. It is possible that every one of these hit strips will be output as a cluster in which case 10×85 clusters will be output at 30MHz for a total time of $28\mu s$. In this case the downstream components will again wait for the hit finders, creating deadtime if necessary. Again the hit finder may truncate this case.

The third error is "backed up". This occurs when, for whatever reason, the downstream components of the SVT are taking too long and are asking the hit finders to wait. The worse case here is that three more L1 accepts can come and the data forced into the hit finder. Since L2 can't make a decision until the SVT is finished the first event, L2 cannot free a L2 buffer and therefore L1 accepts will be held off. Again deadtime is generated.

Unless the hit finder turns out to be a pipeline, these errors can occur in miniature between the hit finder stages too.

References

[1] R. Culbertson and J. Berryhill, *Interfaces to the SVT Hit Finder Board.*

[2] J. Dusatko, H. Sanders *The G-link Fiber Receiver Transition (GRT) Board Design Specification.*

[3] SVT group, *SVT Technical Design Report.*

[4] R. Culbertson *Occupancy in the SVX-II* December 12, 1995.

[5] R. Culbertson *A Clustering Engine for the SVT Hit Finder Board.* September 23, 1996.

[6] S. Belforte *et al Specification of the XTRP, SVT, and Level 2 Interfaces*

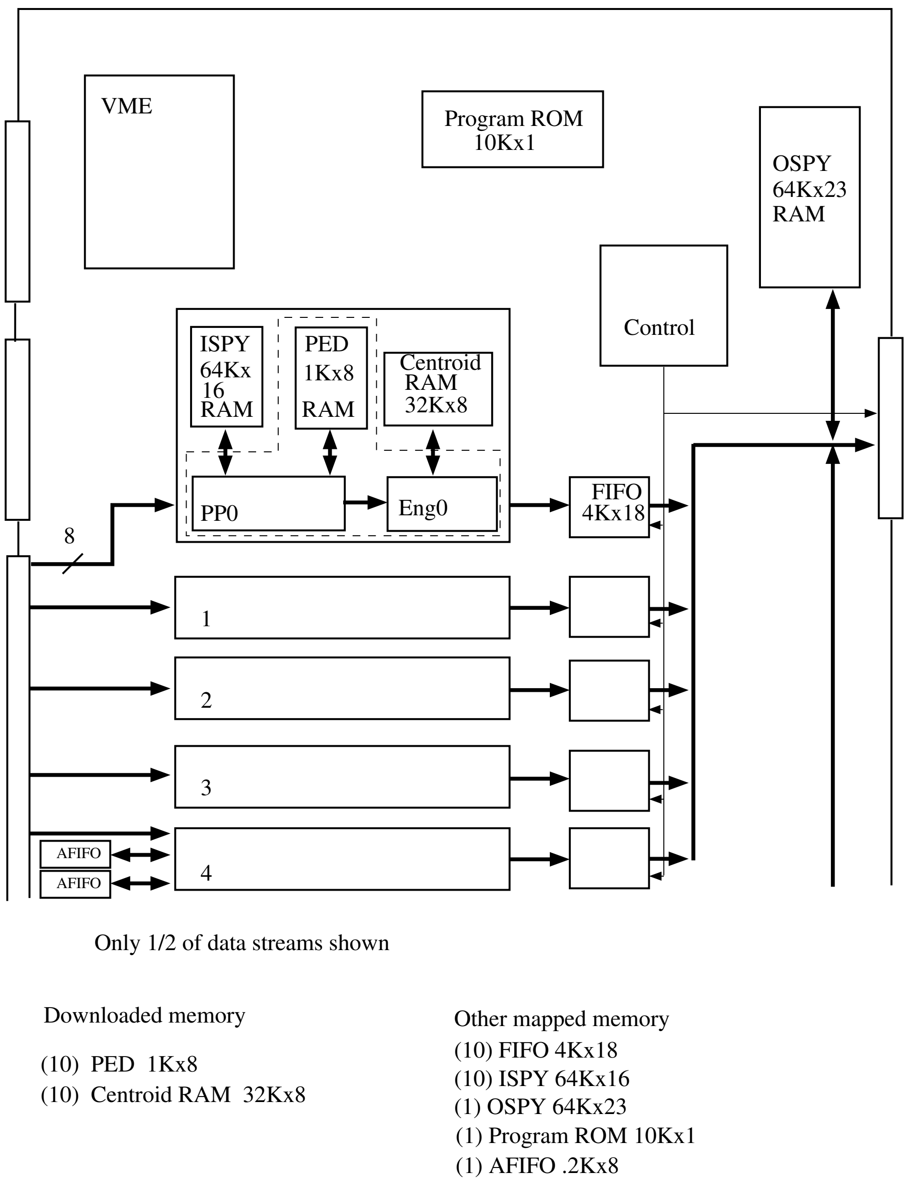

Figure 1: The top level diagram for the Hit Finder board.

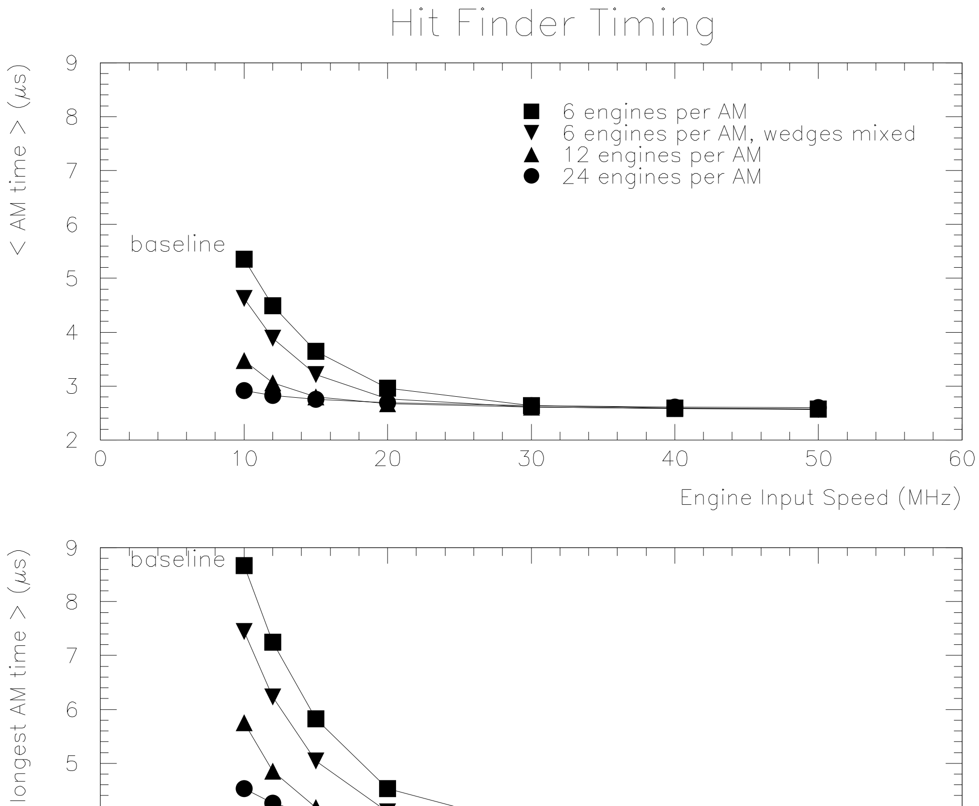

Figure 2: The average time between the the first data in and the last cluster out for the average AM (top) and the busiest AM in the event (bottom). The data is tuned Pythia Monte Carlo. The AM system is assumed to be running at 30MHz.

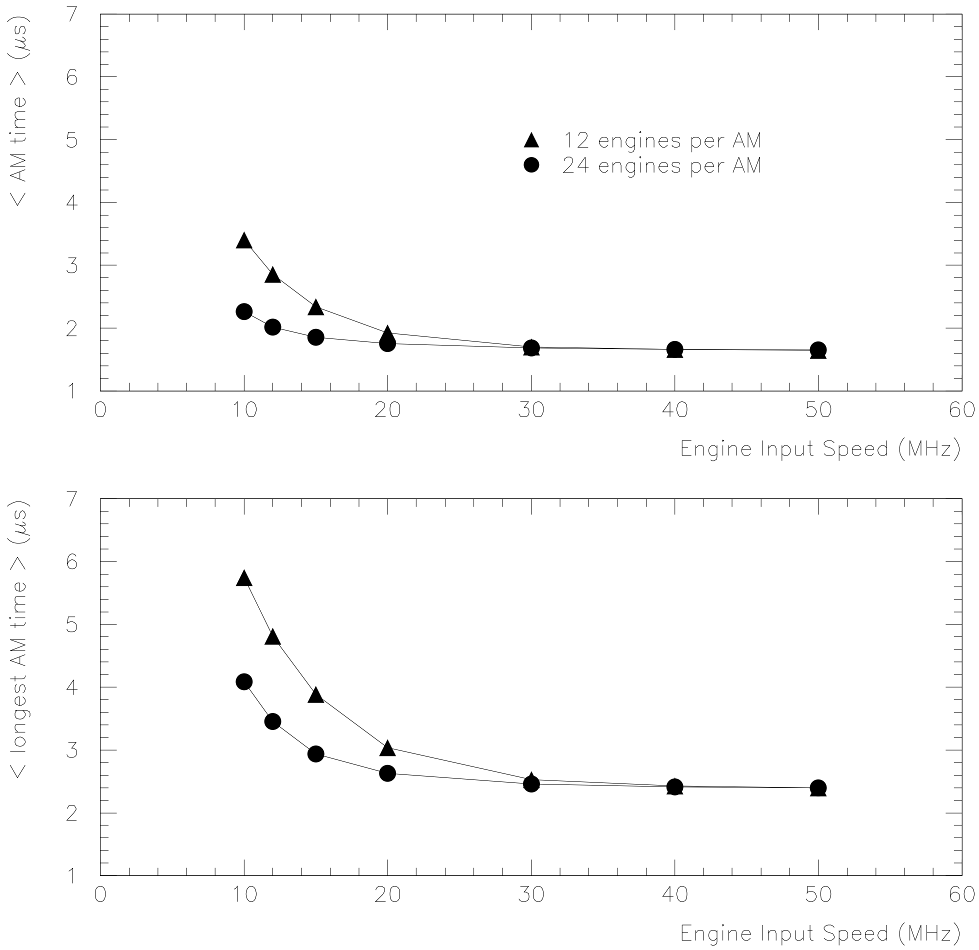

Figure 3: The average time between the the first data in and the last cluster out for the average AM (top) and the busiest AM in the event (bottom). The data is tuned Pythia Monte Carlo. The AM system is assumed to be running at 50MHz. Note that the scale is different than Figure 2.

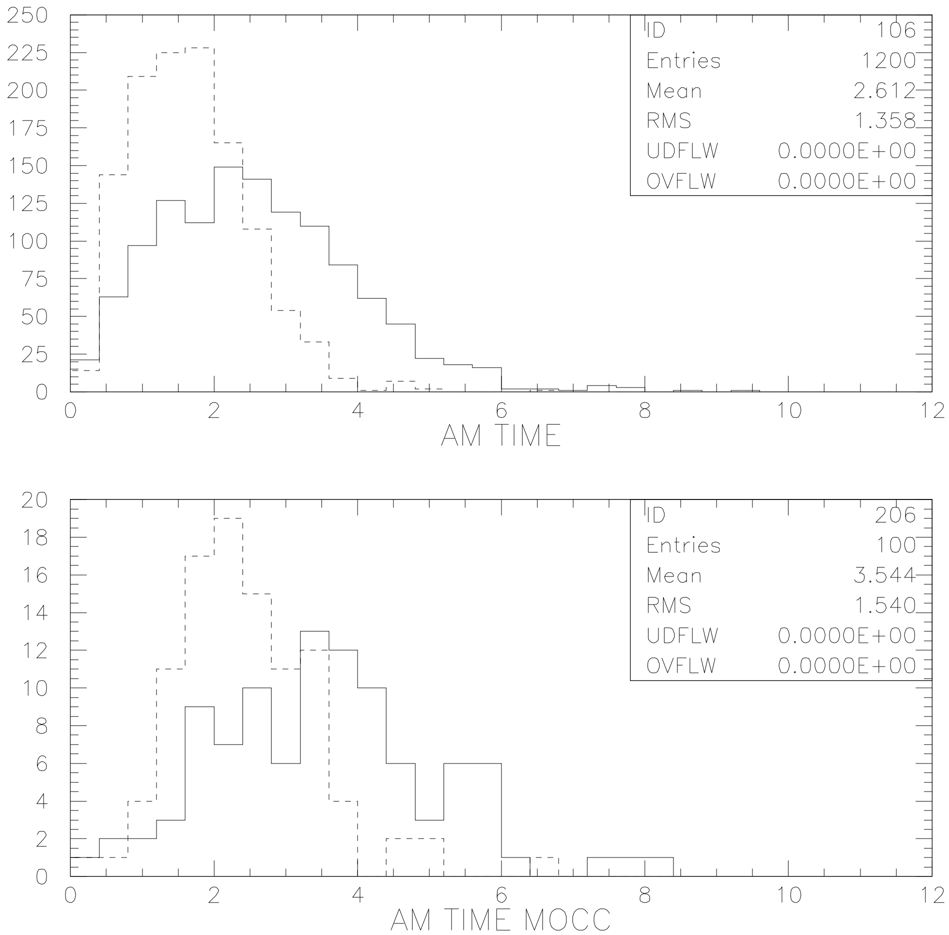

Figure 4: The time between the the first data in and the last cluster out for the average AM (top) and the busiest AM in the event (bottom). The six engine, wedges mixed, 30MHz design is shown as the solid histogram, the 24-engine, 30MHz design, with the AM system running at 50MHz, is shown as the dashed histogram. The data is tuned Pythia Monte Carlo.