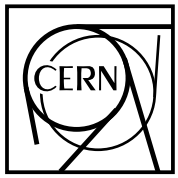
Available on CMS information server

CMS NOTE 1997/045

The Compact Muon Solenoid Experiment

CMS Note

Mailing address: CMS CERN, CH-1211 GENEVA 23, Switzerland



May 27, 1997

APV6 pipeline emulations

M. Millmore

Imperial College, London, SW7 2BZ, UK

Abstract

The data volume from the CMS inner tracker is large enough that data cannot be read out for every bunch crossing, so data are stored in the front end readout chips until a first level trigger signal is received, after which the interesting data are read out. This will reduce the data rate from 40 MHz to 100 kHz. For the silicon microstrips, the data are read out using the APV6 chip which holds the data in an analogue pipeline for up to 3.2 μs. Up to 6 events may be stored in the pipeline at any one time, and data are read out asynchronously.

In any system where data arrives with a random distribution in time, a finite sized memory can become full, causing data to be lost. Because of the complex nature of the APV6 pipeline logic, a true estimate of the proportion of data which will be lost can only be achieved by running a computer emulation of the pipeline logic, with a poisson distribution of trigger signals. The emulation has also been modified to study the effect of other possible logic designs.

1. Introduction

The APV6 is the first complete implementation of a radiation hardened front end chip for readout of silicon microstrips in CMS. It has been designed by the Rutherford Appleton Laboratory and constructed in the Harris AVLSIRA 1.2 µm bulk CMOS process. Work is currently underway to transfer the design to the DMILL process. These chips will also be adapted for readout of microstrip gas chambers (MSGC's).

The APV6 features 128 channels of analogue preamplifier-shapers with a 50ns CR-RC shaping time. Each is followed by an analogue pipeline, consisting of 160 storage cells per channel. The location in the pipeline to which data from the preamplifier-shaper are written is governed by a write pointer, which moves cyclically through the pipeline, one cell per bunch crossing. This is followed by a trigger pointer which lags the write pointer by the level one trigger latency (nominally 130 bunch crossings). The trigger pointer marks interesting cells in the pipeline and in future cycles the pointer will skip over those cells. Data are read out from the pipeline asynchronously, in the order in which the triggers were received. The addresses of the data to be read out are stored in a FIFO which may contain up to 18 addresses.

The readout of the chip may operate in two modes. The simplest mode is where one sample (the peak of the pulse shape) is stored in the pipeline per event. Since the shaping time of the preamplifier-shaper is 50 ns, this is only suitable for low trigger rates, when it will be possible to reconstruct which bunch crossing the data came from off-line. The second mode of operation is the deconvolution mode. In this mode three data samples are stored for each event, as shown in fig. 1. When these samples are summed with different weights applied to each one, the signal can be constrained to one bunch crossing. Since the process of reading the data from the pipeline is destructive, triggers must be separated by at least two bunch crossings.

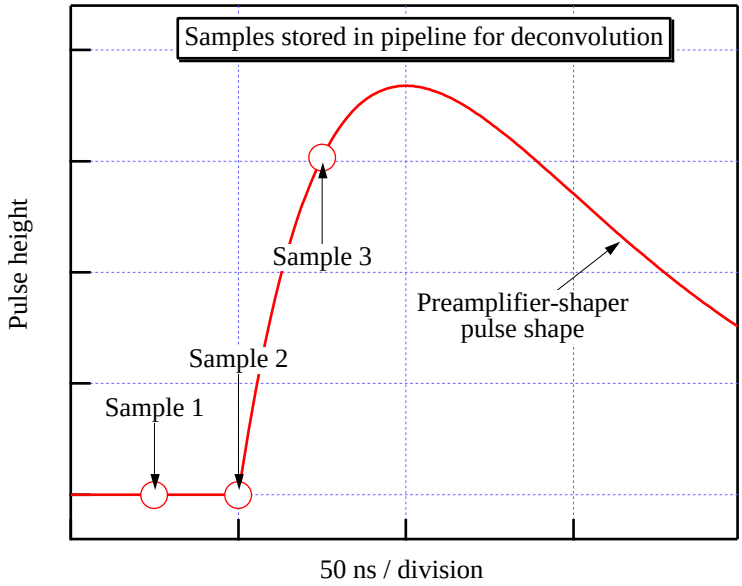


Figure 1. Timing of the three correct samples to recover an event.

After the data have been processed, they are read out through a 20 MHz multiplexer. This speed was chosen so that two APV6 chips may be read out through one analogue optical link with an external 40 MHz multiplexer. The total readout time for each event stored in the APV6 is 7 µs.

The maximum trigger rate in CMS is planned to be 100 kHz. Since it takes 7 µs to read each event from the APV6, and the triggers arrive with a Poisson distribution, it is inevitable that at some point the analogue pipeline will become full. This is true of any storage for events which arrive with this distribution in time. It is therefore important to study the operation of the APV6 to check that the volume of data lost is not large in comparison with the data lost due to the requirement for deconvolution that triggers cannot be received less than three bunch crossings apart.

2. Analogue pipeline emulation.

A study has been previously done of the rate of data loss, based on data queuing theory [1]. This was, however, based on several assumptions which are not correct for the case of the APV6;

- The pipeline has a fixed number of buffers.
- The next event buffer is filled as soon as a trigger is received.

- When the data have been read, the buffers are cleared for overwriting instantly.

This is not exactly the case with the APV6, where;

- Data are tagged in pairs, therefore data will take either four pipeline cells, or in the rare case when two triggers are received three bunch crossings apart, only six cells may be tagged for the two events.
- Cells are tagged as the trigger pointer increments, thus taking two to three clock cycles to complete marking.
- A pair of cells in the pipeline can only be cleared for overwriting after they have been read out and the trigger pointer jumps to the location directly after them.

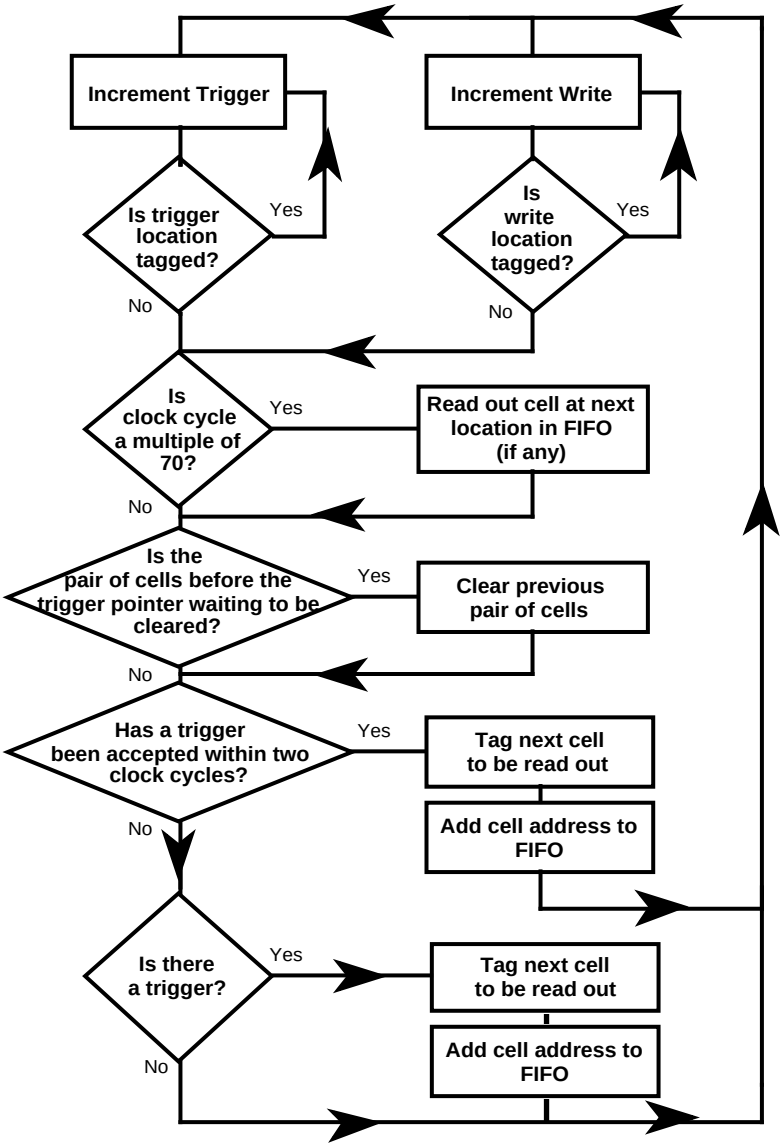


Figure 2. Flow chart of pipeline logic.

It is only possible to calculate exactly the data lost by writing an emulation of the analogue pipeline logic. This logic is shown as a flow diagram in fig. 2, and has been encoded using C++. The emulation was run with triggers arriving in a Poisson distribution (fig. 3) at various rates:

- Trigger rates varying from 50 kHz to 110 kHz at several latencies from 120 to 130 bunch crossings.
- Two extreme examples of 200 kHz and 400 kHz trigger rates at a latency of 130 bunch crossings.
- Trigger rates from 90 kHz to 110 kHz at a latency of 130, with a 40 MHz output multiplexer speed.

It is known that the pipeline will fill up after a finite time. The number of clock cycles until failure has been measured for the different trigger rates. To avoid overfilling the pipeline, some veto must be applied to triggers from the level one trigger logic. Various methods of providing this veto are discussed.

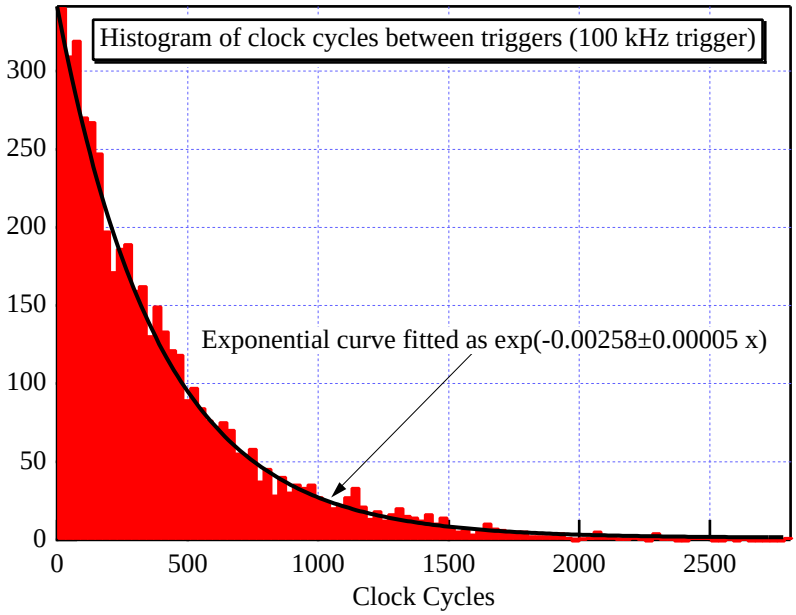


Figure 3. Poisson distribution of trigger signals at 100 kHz

3. Rates of failure

There are two ways in which data can be lost. Firstly the pipeline can become too full (the number of cells tagged is equal to the length of the pipeline minus the trigger latency). Secondly, the FIFO, which contains the addresses of the cells to be read out, can become too full. Fig. 4 shows a distribution of the number of clock cycles until data are lost at several trigger rates. It can be seen that, at 100 kHz, the most probable number of clock cycles before a buffer overflow, if not prevented by the trigger logic, is approximately 50,000.

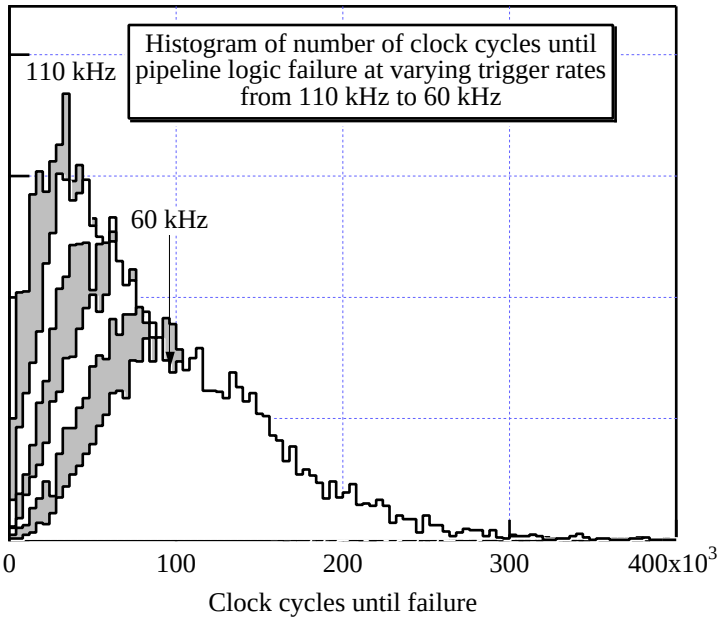


Figure 4. Histogram of number of clock cycles until pipeline or FIFO become full at various trigger rates.

4. Methods of generating a veto.

A simple assumption is that the number of cells tagged as interesting and the number of addresses in the FIFO may be measured by counting the number of triggers sent to the chip and the number of events read out. This, however, does not work because of the way in which cells are cleared for overwriting from the pipeline after they have been read out (fig. 5).

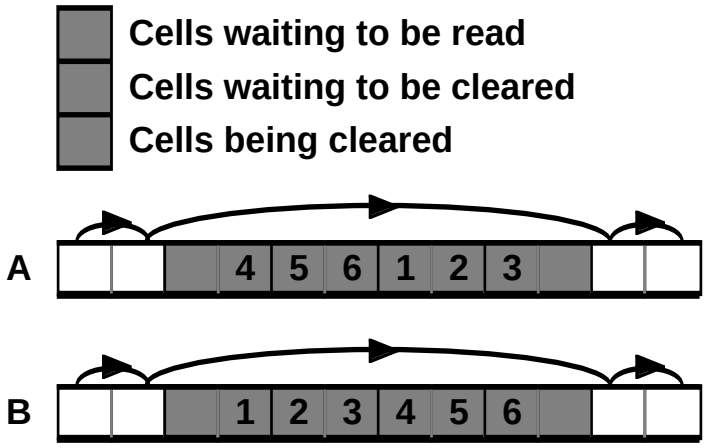


Figure 5. Two events adjacent in the pipeline. Numbers indicate the order in which the cells were tagged. In example A, the last cells from the first event will be cleared as the trigger pointer passes directly to the right of them. In example B, the first event cannot be cleared until the second event is read out and cleared since the trigger pointer cannot be directly to the right of the cells until then.

To preserve the correct latency, a pair of cells which have been read out and are waiting to be cleared for overwriting can only be cleared when the trigger pointer is in the cell directly after them. The consequence of this is that if two consecutive triggers are received then the cells from the first event will not be cleared until the second event has being read out and cleared, since only then can the trigger pointer be in the cells directly after the first event's cells. In this way, consecutive triggering can fill the pipeline, although there may only be one event waiting to be read out.

4.1 Using an APV6 to generate a veto

The next logical method of generating a veto is to use an APV chip externally to monitor the state of the other chips. Whilst this method has a certain simplicity, it also has several drawbacks. The only diagnostic outputs are two pads, one of which goes high when the trigger pointer passes the start of the pipeline and the other when the write pointer passes the start. This gives no information about the state of the FIFO and therefore the worst case must be assumed (every cell tagged requires a FIFO location). This information is only updated every cycle of the pipeline, and it can provide no other information about the state of the chip, for example to which location the data should have been stored in, which may be used to check that all chips are functioning correctly. The proportion of data lost using this method to veto triggers is shown in fig. 6, together with the data lost as calculated by the data queuing method.

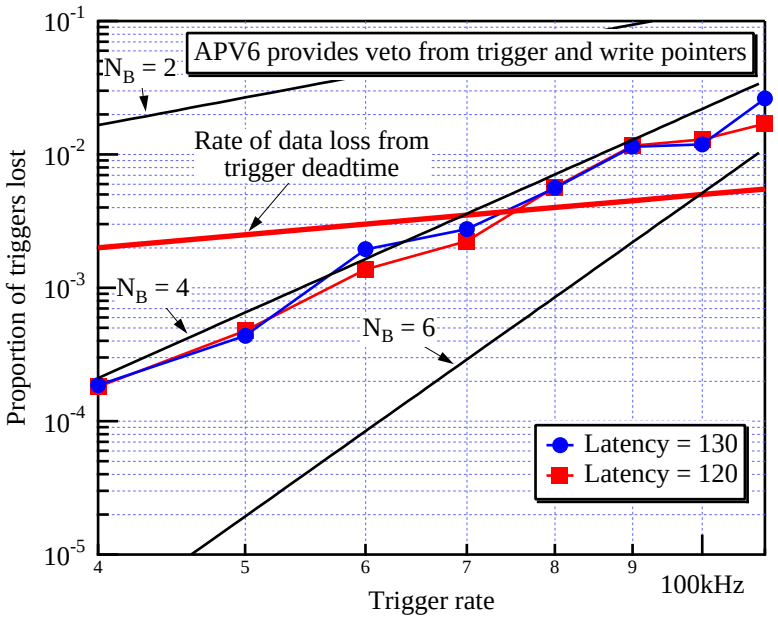


Figure 6. Proportion of data lost if an external APV6 is used to provide a trigger veto.

The proportion of data lost is similar to that calculated for four buffers from the data queuing model. This is to be expected because of the assumption that every cell tagged in the pipeline will have an address written to the pipeline. This would only happen in extreme circumstances, where consecutive triggers are constantly received. Under normal circumstances three addresses will be written to the FIFO for every four cells tagged in the pipeline, due to the fact that the cells are tagged in pairs. The effective size of the FIFO is therefore reduced from 18/3 = 6 to 18/4 = 4.5.

The solid line in fig. 6 shows the proportion of data lost due to the deadtime of two clock cycles after a trigger. Above 75 kHz trigger rate, the proportion of triggers lost from the veto is greater that the proportion of triggers lost from the deadtime. At 100 kHz trigger rate, the proportion of triggers lost is over 1%. This would be unacceptable for CMS.

4.2 Using an emulator to provide a veto.

A Field Programmable Gate Array (FPGA) may be programmed to contain an emulation of the APV6 which will run in real time. Since this would give exact information about the state of the pipeline and the FIFO, a veto signal for when the APV6 was unable to accept a trigger could be provided correctly on an event-by-event basis. The chip could also provide internal information about the current state of the APV6s, for example at which locations in the pipeline the next event should have been stored. Since the APV6 reads out this information with each event, this can be used to check for any chips which have come out of synchronisation with the rest of the system. The proportion of data lost with this scheme is shown in fig. 7.

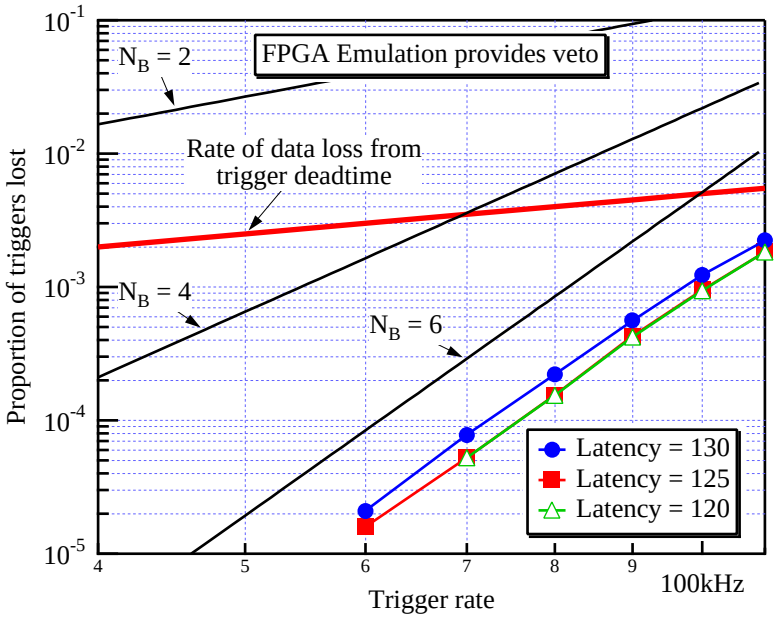


Figure 7. Proportion of data lost using an external emulation of the APV6 to provide a trigger veto.

The emulation was also run in the normal mode for 130 bunch crossings latency with a trigger rate of 200 kHz and 400 kHz to test the chip under extreme circumstances. At 200 kHz, 11% of the triggers are lost and at 400 kHz, 52% of the triggers are lost. These trigger rates would only be seen if there were an error in the level one trigger processor.

The proportion of triggers lost at 100 kHz trigger rate represent less than 0.1% of the total data. This is notably better than the data queuing model for 6 buffers in the pipeline. It is believed that this is due to the fact that data are not stored in fixed buffers, but in pairs of cells in which one pair of cells can be cleared before the other has been read-out, thereby creating extra space in the pipeline. This has been investigated by running slightly modified emulations:

- Three cells tagged individually but cleared in one block.
- Three cells tagged individually and cleared individually.
- Cells tagged in pairs and cleared in pairs (normal mode).

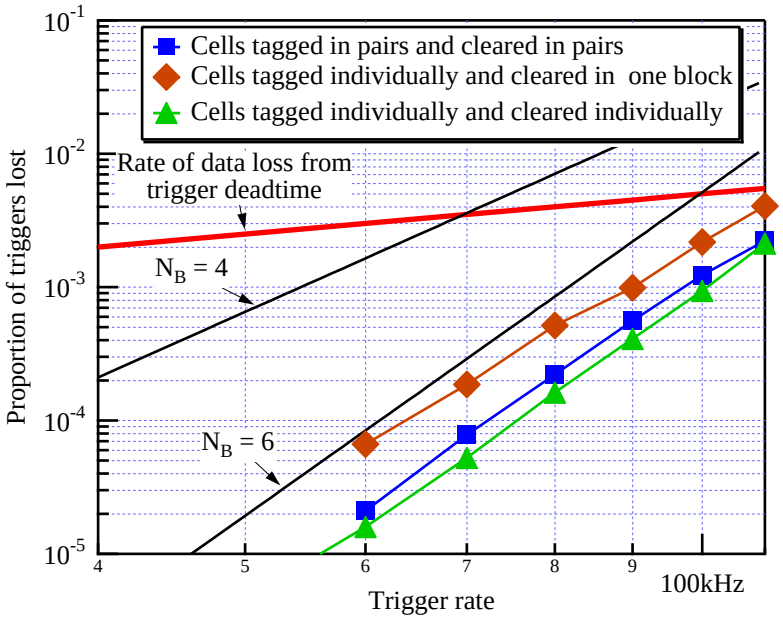


Figure 8. Emulation is altered to investigate pipeline efficiency. Latency = 130 bunch crossings.

The results of are shown in figure 8. When cells are tagged individually and cleared individually, which is the optimum condition, the proportion of data lost is approximately 25% lower than under normal operating conditions. When the cells are tagged individually (3 cells tagged, not 4) and cleared in a block, the performance is closer to that seen for the case of 6 buffers in the pipeline. It is slightly better since only 3 cells are tagged for each event, giving space for 10 events in the pipeline. The limiting factor is then the FIFO which by it's nature must clear one cell at a time, thus giving slightly more than 6 buffers.

4.2.1 Emulation of a 40 MHz multiplexer

It is also interesting to study the proportion of data lost if the APV6 were to run with a 40 MHz output multiplexer. This would require one optical fibre for every APV6 and is not therefore envisaged at present. To study this situation, the logic of the code, as shown in fig. 2, is changed such that the next pipeline cell is read from the pipeline every 35th clock cycle. 140 clock cycles are then required to process each event, during which time 128 channels of analogue data and 12 channels of digital header from the previous event are read out at 40 MHz. Fig. 9 shows the proportion of data lost under these conditions for a trigger latency of 130 bunch crossings. Note the change of scale from the previous graphs.

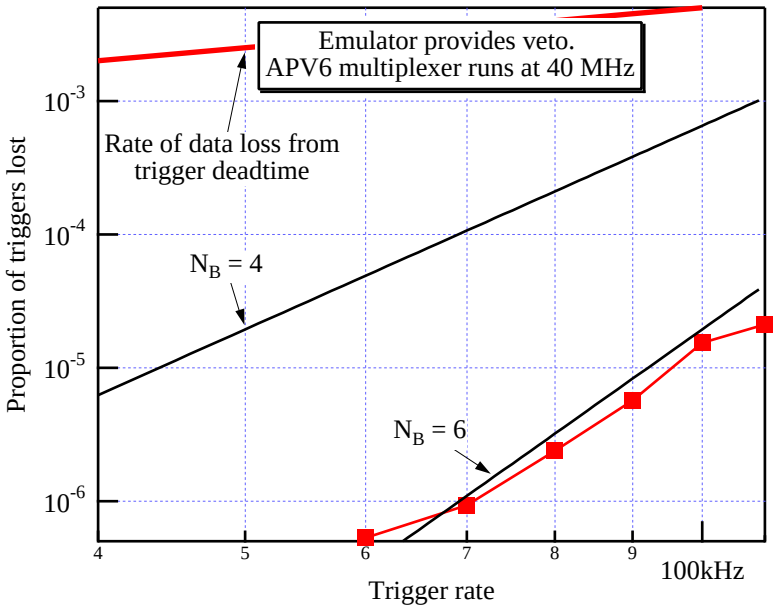


Figure 9. Proportion of data lost using an external emulation of the APV6 to provide a trigger veto. APV6 output multiplexer is run at 40 MHz.

The proportion of data lost at 100 kHz trigger rate is approximately 1.5×10^{-5}. This is comparable with a pipeline with six buffers as predicted by the data queuing model. The result is much closer to that of the data queuing model because the total number of clock cycles to read all of the cells is 140, which is less than the length of the pipeline, therefore the cells can start to be cleared on the next pass of the trigger pointer.

5. Physical location of a veto generator.

It has been shown that it will be required to veto some trigger signals so that the APV does not fail. There are several places in which this may be incorporated into the system:

- The APV may be redesigned to provide the veto.
- The logic may be in an FPGA on the Front End Driver (FED).
- The logic may be in an FPGA on the Front End Controller (FEC).

While all of these are physically possible, not all are desirable for the whole system.

5.1 Veto generation from the APV

It is not impossible to redesign the APV6 to include logic to ignore triggers when the pipeline or FIFO are too full to accept them. While this would appear to be a simple solution, it has several disadvantages;

- The logic to provide the veto is being replicated 100,000 times, leading to excessive duplication.
- Having the logic inside the tracker would provide no external monitoring of the state of the chips.
- Sending information about the number of triggers missed would increase the data packet size, thus increasing the proportion of data lost.
- The chip area would be increased, increasing the cost per channel.
- There would not be enough time to send a veto to the global triggering system if that were required.

5.2 Veto generation in the FED

It would be possible to generate a veto signal from an FPGA emulation inside a FED. This veto could be passed to the FECs to veto the actual triggers sent to the APVs. The FED would have the information about which pipeline cells the data were stored in to compare with the information sent from the APV to check that they were all in synchronisation. The disadvantages of this method are;

- Either an FPGA must exist in all of over 1000 FEDs or there must be some synchronised communication between them at 40 MHz.
- The FEDs and the FECs are not situated next to each other, therefore the signal to the FEC must be predicted in advance (6 bunch crossings for 50m separation). It must be assumed that triggers could occur during this time, thus reducing the effective number of buffers, and greatly increasing the proportion of data lost.
- It is desirable to keep the FED as a data driven device, not a data driving device.

5.3 Veto generation in the FEC

An FPGA could be situated in one or all of the FECs where the trigger would be vetoed at the first possible opportunity within the tracker electronics. Since only one crate of FECs is envisaged, it would be simple to send the signal to the other FECs. The FEC would have a link to the FEDs which would either transmit the information expected for each event to verify that each APV is correctly synchronised, or transmit a signal to say that a trigger had been vetoed so that the FED may send out a signal to the rest of the system. The FEC has at least 7μs to send the signal since the FED does not need the information until the trigger has been transmitted to the APV and it has returned its data.

5.4 The global triggering system

It is important to decide whether or not the veto signal will be sent to the global triggering system. If it were known that the veto would always be sent to the global triggering system then it could be more productive to place the emulator on a separate card. It would then also be desirable to put an emulator in the FEDs to check the state of the APVs against the expected state to check their synchronisation. This emulator would not veto any

triggers, but could be used as a double check. The FED need send no information other than the events to the global data acquisition system, and no information about the event need pass from the FEC to the FED.

If the veto is not guaranteed to always go to the global triggering system, then it is essential that the emulation is in the FEC to minimise that data lost as outlined above. The FED would either not send any data for that event, insert a dummy (e.g. all zero) event, or send a signal to say that an event had been vetoed.

6. Conclusions

Any system which uses a finite length memory to store randomly arriving data can overflow at some point. The data acquisition system in CMS will lose 0.5% of the interesting data at 100 kHz due to the deadtime in the system caused by deconvolution requirements. It is desirable to keep the proportion of data lost due to filling the memory small with respect to this.

There are several methods of detecting when the pipeline will fill. If the APV6 in its current design is used to detect when the pipeline is full, an additional 1% of data will be lost at 100 kHz. This figure is too high. If an FPGA emulation of the pipeline logic is used then only an additional 0.1% of the data will be lost, which is an acceptable figure.

The location of the emulator depends on whether the trigger veto signal first is passed to the global trigger or not. The most attractive location is in the FECs as this will minimise data loss and retain simplicity. A link between the FEC and the FEDs could also provide a check of the synchronisation of every APV in the system.

References

[1] Optimisation of the CMS tracker readout system. G. Hall. CMS TN/95-003