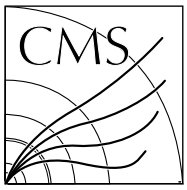
Available on CMS information server

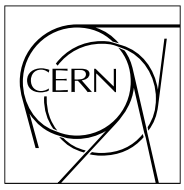
CMS NOTE 1999/028



The Compact Muon Solenoid Experiment

CMS Note

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



28 April 1999

APV logic simulations

N. Marinelli

Imperial College, London, UK

Abstract

The amount of data in the CMS inner tracker system at the LHC interaction rate is so large that it cannot be read out at each bunch crossing. A pipeline memory is then needed to store the data at the front–end level until a Level 1 Trigger accept signal marks the interesting data which will then be read out. Due to the random arrival of triggers with a maximum average rate of 100kHz a queue may develop in the pipeline which in the end can become full and cause errors. Some triggers must then be vetoed.

In the present version of the chip to be used in the Tracker read–out, the APV6, the memory (storage, marking of interesting data, read–out and clearing) is governed by a very complex embedded logic: precise predictions concerning the inefficiency due to vetoing the triggers can be done only by means of computer simulation.

This document is mainly intended to give an extensive description of the logic and to present the results obtained by running an updated version of the simulation.

A further generation of the chip, the APV25 (based on silicon submicron technology) is under development: its digital logic will be simplified with respect to the APV6. Results on the efficiency expected from the APV25 are also shown.

1 Introduction

The APV6 front–end chip is equipped with 128 analogue channels, each of which contains a preamplifier–shaper stage (CR–RC) with a peaking time of 50 ns followed by a pipeline memory 160 locations deep where the data can be written at the LHC 40 MHz rate so that the memory always contains a record of the most recent beam crossing the chip has sensed.

Writing and reading of data in the pipeline are governed by two pointers, WRITE and READ, which circulate in the pipeline jumping from cell to cell clocked by the LHC 40 MHz clock. The two pointers are separated in time by a Level 1 Trigger Latency, so that the READ pointer always falls in the cell where data have been written a Latency before. If a L1 Accept occurs the cell corresponding to the READ pointer value is marked as interesting.

The addresses of interesting cells are stored in a FIFO memory 18 locations deep: data from these cells will be read out asynchronously in the order the triggers were received.

Data read–out can be performed in two different modes: “peak mode” and “deconvolution mode”. In peak mode only the peak value of the signal coming from the preamplifier–shaper is sampled in the pipeline, so only one cell per event is needed.

In deconvolution mode, where the complex APV logic comes in, the signal from the preamplifier–shaper is sampled (see Fig. 1) off the peak. The three samples are combined by the APSP (Analogue Pulse Signal Processing) deconvolution filter which performs a weighted sum obtaining a signal with a peaking time of 25 ns [1].

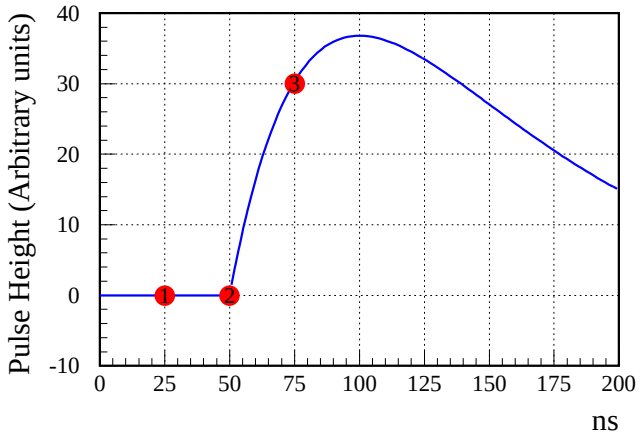


Figure 1: The three samples stored in the pipeline for deconvolution.

After the data have been processed by the APSP they are read out through a 20 MHz multiplexer, so that two APV6 can be read via one analogue link with an external 40 MHz multiplexer.

Since the total time required to perform the read–out of one event is 7μs and the maximum average trigger rate foreseen at CMS is 100 kHz with Poisson fluctuations, at some point the pipeline, or the FIFO, can become full causing an error condition with consequent loss of data. Such a condition is recoverable only by resetting the chip.

The occurrence of such failures must be avoided. The possibility of vetoing those triggers which could cause an error condition by simulating the chip in real time with an FPGA has been already considered and accepted in the past [2]. Such a state machine simulation would indeed provide exact knowledge about the current state of the pipeline and FIFO. The other possible strategy would be monitoring the state of the chips by using an APV6 external to the system. Unfortunately the diagnostic which can be performed in this way is not satisfactory, since no information about the state of the FIFO can be obtained nor information about which location the data should have been stored.

Vetoing the triggers leads of course to some inefficiency but it is certainly the best strategy to be adopted, in order to avoid dead time and loss of data due to the reset.

It is then mandatory to make a good estimate of the magnitude of such inefficiency; due to the complex logic

embedded in the APV6, this information is obtainable only by modelling the logic with a computer program.

2 APV6 pipeline logic

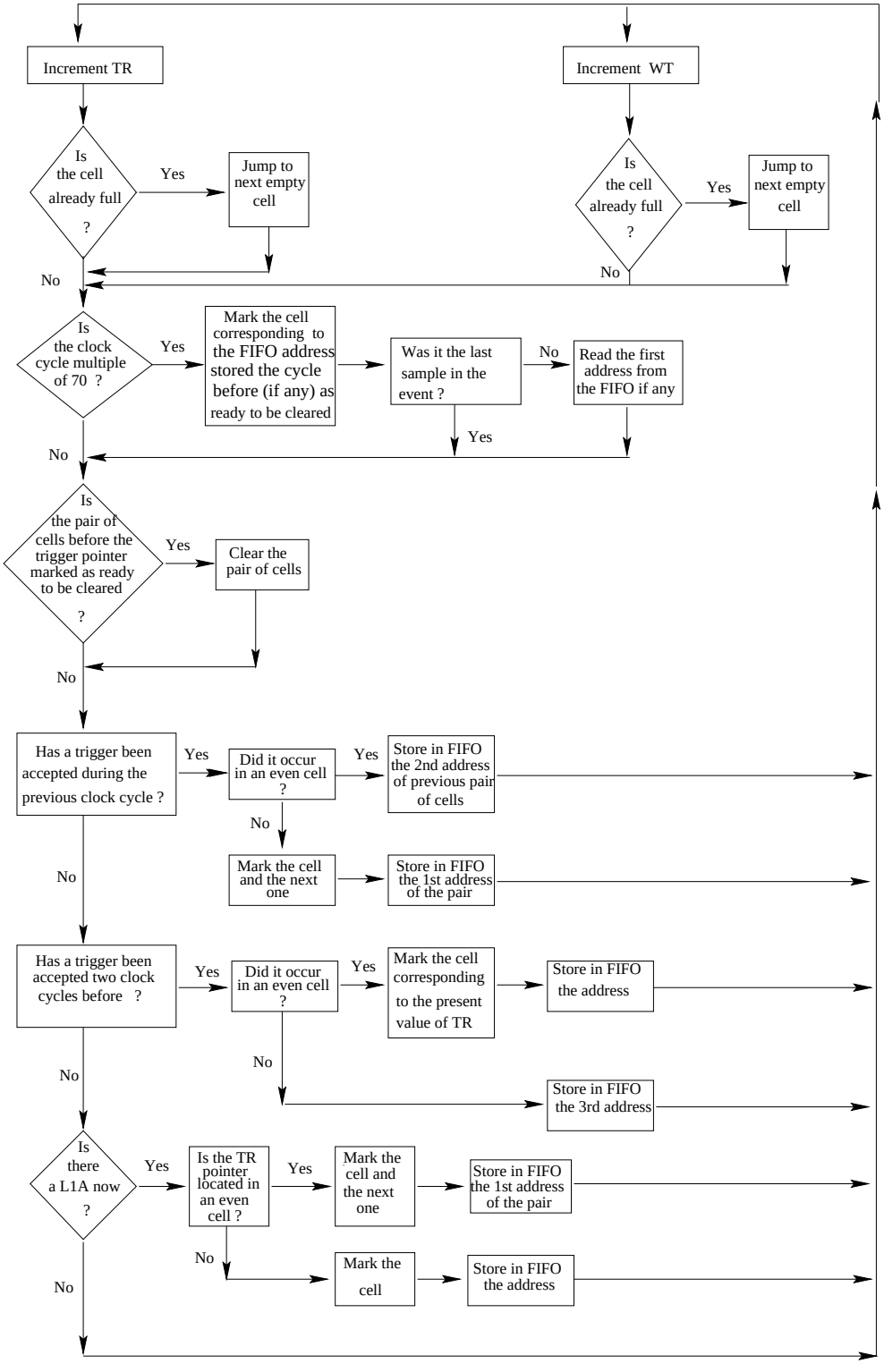


Figure 2: Flow chart of the APV6 logic. WT= write pointer, TR= trigger (read) pointer.

The flow chart shown in Fig. 2 is an attempt to describe in a schematic but complete way the logic the APV6 follows to write data in pipeline locations, mark the interesting cells, perform the read–out and finally clear the

cells which have been read out, so that new data can be stored.

The best way to read the flow chart is starting from a Level 1 Accept signal (bottom left hand side of the picture).

The WRITE and READ (or TRIGGER) pointers move in the pipeline every bunch crossing: to preserve the correct latency full cells are skipped by both the pointers.

On the occurrence of a L1 accept signal, cells are marked as interesting for later read–out: to reduce the amount of circuitry on the chip, cells are marked in pairs, so a single event takes four cells, one of which doesn't contain data. The behaviour of the chip is slightly different depending on where the READ pointer is located (even or odd cell) when the L1 accept occurs, as can be seen from Fig. 3, so that from two to three clock cycles are needed to complete the marking and three clock cycles to store the addresses of the marked cells.

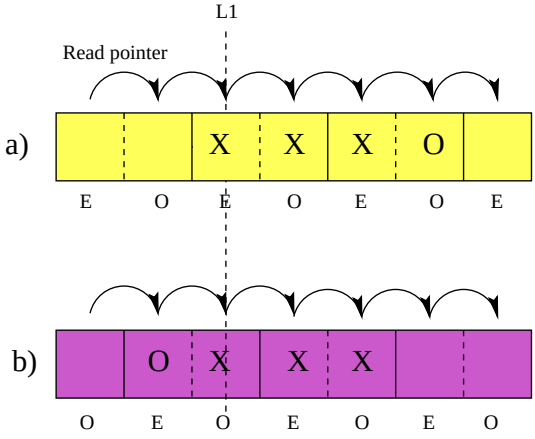


Figure 3: Marking of cells in pairs. L1 accept occurs: a) in an even cell, b) in an odd cell.

Every 1.75μs the chip checks if there are data waiting to be read out: if yes, the first address stored in the FIFO is read out. Reading the content of the corresponding cell takes 1.75μs. After the readout of the three samples have been completed (5.25μs), the APSP operation takes another 1.75μs to perform the deconvolution. Therefore each event is read out in a total time of 7 μs. If further cells are waiting to be read out another readout cycle begins immediately after. If no more cells are waiting to be read out, the chip reverts to checking every 1.75 μs.

Once the cells have been read out they are ready to be cleared: cells are cleared in pairs but this only happens when the READ pointer is located in the empty cell immediately on the right of a ready-to-be-cleared pair of cells. This action, needed in order to preserve the correct latency, has the following consequence: if two consecutive triggers (see next paragraph) are received, the cells belonging to the first event will be cleared only after the second event has been cleared. This happens because only in this way can the READ pointer be directly after the cells belonging to the first event (see Fig. 4).

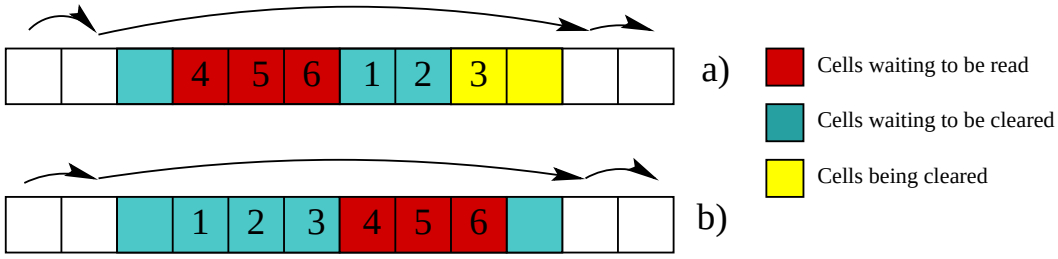


Figure 4: Adjacent events stored in the pipeline. Cells are numbered in the order they have been marked: a) Cells 1,2 from the first event will be cleared as soon as the Read pointer will, on the next cycle, be on their right; b) Cells 1,2,3 from the first event are ready to be cleared but they will wait until the second event will be read out and cleared.

This explains why bursts of consecutive triggers can fill the pipeline even if only one event is waiting to be read

out. This is why the behaviour of the APV6 running in deconvolution mode is not predictable without support of a simulation.

As already pointed out in the past [2], triggers coming less than three bunch crossings apart must be vetoed for two reasons: conceptually, if the chip is operated in deconvolution mode two consecutive triggers would mean overwriting data; also the chip has been designed to sense two consecutive triggers like a '11' signal on the trigger line and two triggers two bunch crossing apart like a '101' signal. These signals have respectively the meaning of a 'calibrate pulse request' and 'reset signal'. So 'consecutive' triggers must always be intended to be at least 3 bunch crossings apart.

The pipeline is 160 locations deep, but its effective depth is 160 minus the latency value. Up to a latency of 132 bunch crossings, up to 6 events can be stored in the pipeline, the number being limited by the number of FIFO locations. At higher latency the number of events is limited by the number of cells available.

The information concerning the address of the third cell (sample) is contained in the APV6 output data stream(Fig. 5). This allows to confirm the operation of the logic and, in the final system, to verify early that the synchronization has not been lost. Eight bits, after the digital header, are used to encode the address (Gray code).

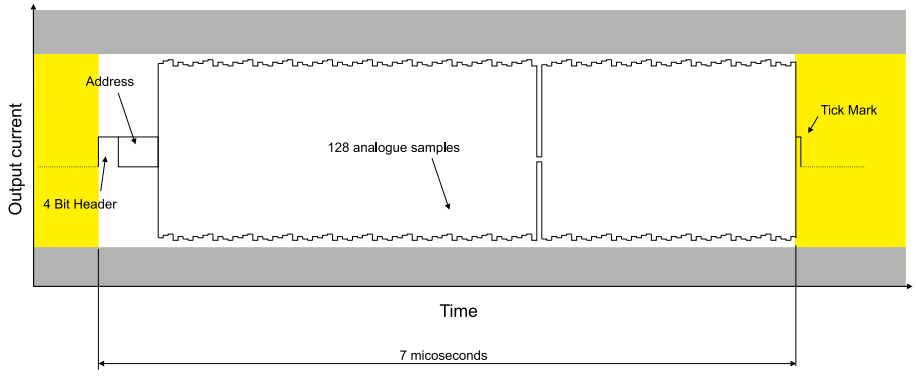


Figure 5: APV6 output data stream.

3 Simulation and related checks

The logic described in the previous sections has been encoded in the C language, updating the original simulation [2]. From the comparison between the simulated results and the behaviour of the chip in the laboratory, the new simulation appears to reproduce correctly the behaviour of the real APV6.

A very simple experimental setup has been used as a first attempt. A programmable multi–channel pulse generator, SEQSI (SEQuencer for Silicon Investigation) [3] has been used to trigger an APV6 with a short sequence of random triggers (100 kHz average rate). The APV output data streams have been observed on an oscilloscope and the addresses of samples have been inferred from the digital header.

The same sequence of about 50 random triggers has been used as input for the simulation program and a comparison between the addresses (real and simulated) of the marked cells has then been made. With the old simulation no agreement was found even with such a short sequence of triggers: the simulation could reproduce only the addresses of the first few events. The problem has now been solved and with the revised simulation the agreement is very good: the simulation predicts correctly the whole sequence of addresses produced by the chip and it predicts an error (due to FIFO full) exactly when the chip experiences it (this error can be detected by means of the internal watchdog logic which switches low the 3rd bit in the digital header of the data frame).

The main improvement introduced in the new simulation is related to the timing used to read out and clear the cells. Indeed in the old version cells were marked as ready to be cleared as soon as their addresses were read out from the FIFO, without taking into account that the actual reading time needed for the electronics to read the content of each cell is 1.75μs. This implied a too fast clearing of the pipeline and as a consequence the chip inefficiency was underestimated.

The time until the occurrence of a failure has been measured as a function of the mean L1 trigger rate and assuming a L1 latency of 130 bunch crossings, by counting the number of clock cycles (25 ns) between a reset and a failure.

At each failure the APV6 must be reset and the counting loop restarted. The distributions showed in Fig. 6 have been obtained by observing 10K failures each.

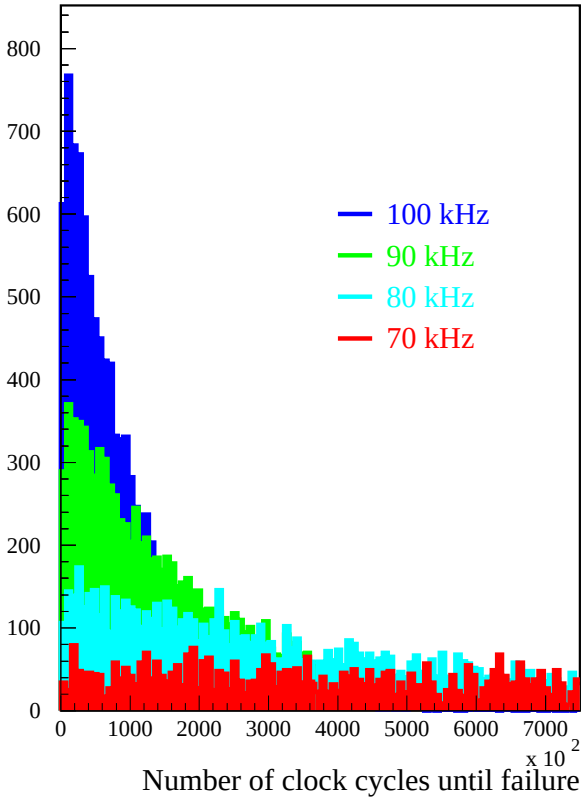


Figure 6: APV6 failure rate.

The distribution obtained at the average L1 trigger rate of 100kHz shows a fast rise at low values of the time followed by an exponential decay which can be fit with the following function (see also Fig. 7):

$$f(x) = p_1 \, e^{-x/p_2} \left[1 - e^{-(x/p_3)^2}\right] \tag{1}$$

Since each entry in the histogram is determined on the basis of an initially empty chip (indeed after each failure the chip must be reset), the shape of the distribution can be given the following statistical meaning: starting from the empty state, the chip takes some time to reach situation in which its memory has an "average occupancy". Not surprisingly this time is comparable to (and longer than) the average time of arrival of 6 triggers (6×400 clocks = 60 μs), 6 being the buffer length. Once the average state has been reached, the probability for the chip to fail becomes time-independent or, in other words, it is totally uncorrelated with the fact that the chip has been reset, being only related to the random occurrence of a bad sequence of triggers, where "bad" means with respect to the "average" condition of the memory. The decay constant of the exponential (p_2=85080±1003 from the fit, corresponding to 2.13±0.03 ms) can then be interpreted as the "average lifetime" of the chip before failing.

The fit has been performed also at 90 and 80 kHz, finding respectively decay constant 178240±3453 clocks (4.45±0.09 ms) and 434030 ± 27925 clocks (10.7±0.7 ms).

4 Trigger veto: inefficiency

The fraction of data lost by vetoing triggers which otherwise would cause an error condition can then be estimated. By means of the simulation the number of full cells in the pipeline and in the FIFO is known at each time so it is relatively easy to determine when a trigger must be discarded. An FPGA will simulate the APV in real time in

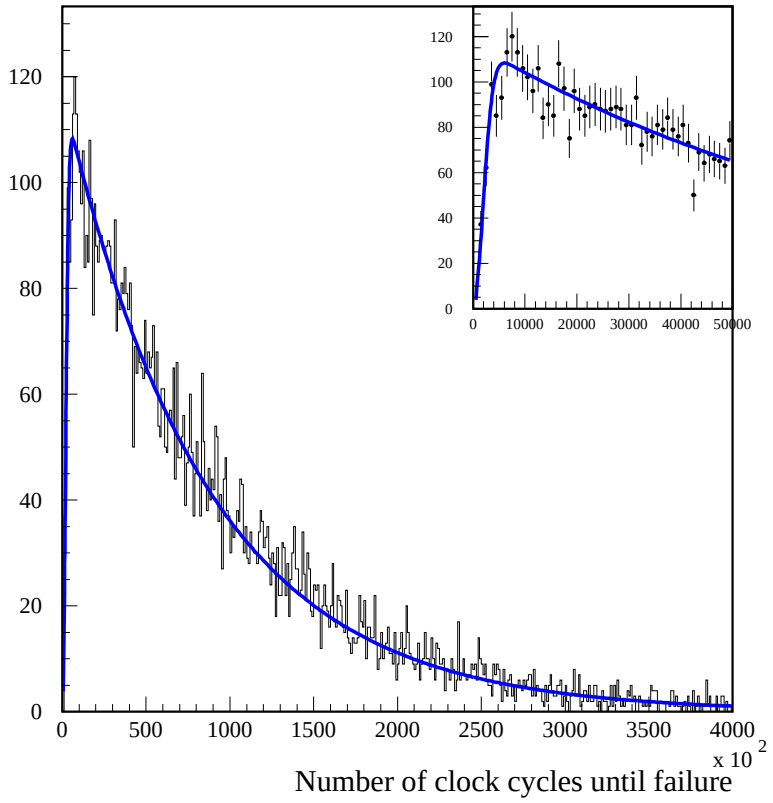


Figure 7: Rate of failures at 100 kHz. The smooth curve on the plot is the function [1], the value of the parameters obtained from the fit being p_1 = 117± 2, p_2 = 85081± 1003, p_3 = 2710± 188, with χ^2/d.o.f=0.88. The plot on the right upper corner shows a zoomed view of the rise.

the real system, so that an inhibit signal can be sent back to the global L1 trigger or locally to the unit (Front End Controller) which distributes the L1 signals to the front–end.

The new results are shown in Fig. 8 as a function of the mean trigger rate. At 100 kHz and for a latency of 128 bunch crossings the new estimate of the inefficiency is about one order of magnitude larger than was believed before (0.8% instead of about 0.08%) and it is now larger than the loss (0.5%) due to the inhibition of triggers in the two bunch crossings following each trigger. At larger latencies the fraction of data lost reaches values of the order of 1.5%. This implies an increase of the CMS L1 latency would incur a significant penalty. By using the old simulation it was envisaged that, by reducing the buffer depth to 5 events, a Latency of 136 bunch crossings could be afforded [5] with an acceptable loss of data (about 0.5% at 100 kHz). This of course is no longer the case. The results obtained with the revised simulation definitely tend to discourage this option.

5 Expectations from the APV25

The most appealing feature of devices built with the 0.25 μm process is their intrinsic high radiation hardness, which is mandatory, as is well known, in the CMS Tracker environment. The first prototype of APV25 is now under way.

The APV25 is basically structured as the APV6 but includes some new features in each of its stages [4]. In particular the size of the cells forming the pipeline are significantly smaller (now 35×13 μm^2) allowing saving of space to allocate circuitry.

The space saving is then partially used to make a longer FIFO (from 18 to 32 locations) and a longer pipeline (from 160 to 192 cells). On the other hand the cell pairing in the pipeline can be avoided. This constitutes a major improvment from the logic point of view, since the cells can be marked one by one so that each event takes

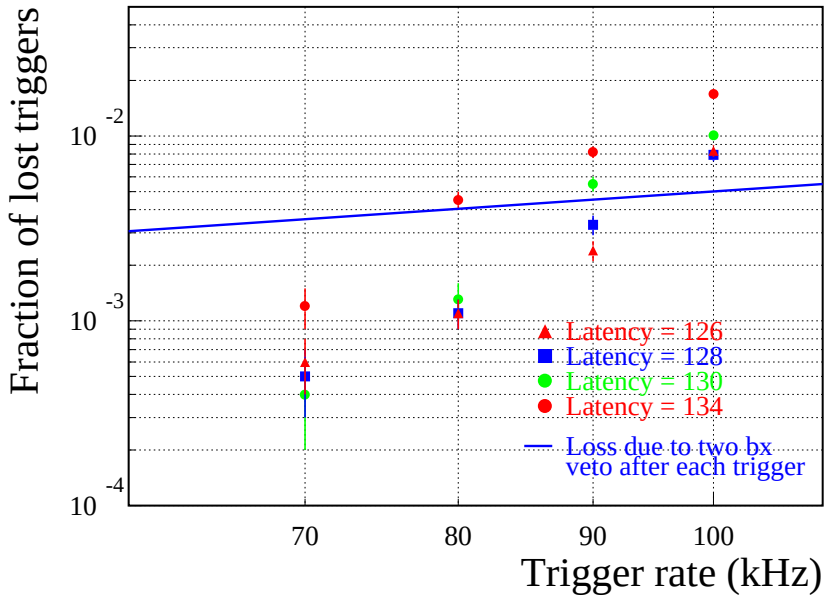


Figure 8: Fractional loss of events caused by vetoing triggers which would cause a memory overflow. The line on the plot represents the loss due to the rejection of triggers coming less than 3 bunch crossings apart.

three cells instead of four. They are also cleared individually, implying a drastic simplification of the logic and consequently of its description.

Within the new configuration, the maximum number of event which can be queued at all times is 10, the number being limited, of course, by the number of locations available in the FIFO for writing 3 addresses per event. With the pipeline 192 cells deep, 10 events can be stored up to a L1 latency of about 160 bunch crossings.

The simplified logic has been encoded in a C++ code and the results obtained confirm the expectations: the probability of failing because of full FIFO or full pipeline should be much lower and as a consequence, the inefficiency caused by vetoing triggers (following a real state machine emulation) is very low.

As in the APV6 case, the behaviour of the chip has been studied as a function of the mean L1 trigger rate and latency. At 100 kHz and with a latency of 130 bunch crossings the estimated average time until a failure is about 40ms.

If the state machine simulation is exploited to veto the triggers which would cause the failures, the loss of data is independent of the latency up to a value of about 146 bunch crossings since up to that value the limiting factor is the length of the FIFO and not the number of cells available in the pipeline. At higher values of the latency, the number of cells available in the pipeline (192 minus the latency) decreases and becomes critical. In the range of latencies where the APV6 can operate the loss measured in the APV25 is of course remarkably lower, as can be seen in Fig. 9. The results shown have been obtained on the basis of 100K triggers sent to the chip, corresponding to a running time of 1 sec.

At lower trigger rates, the situation is of course even better: at 90kHz the fraction of triggers vetoed is 4×10^{-5} (100K sent in 1.1 sec) and at 80 kHz, it has been necessary to run the simulation for a longer time ($\sim$351K triggers in 4.4 sec) to veto a trigger.

6 Future plans

The very simple experimental setup used to perform the comparison between the APV6 results and the simulation did not allow 1) feeding the real chip with long sequences of triggers, 2) working in a systematic way by acquiring more statistics.

It is indeed very important to understand what happens with longer sequences: the simulation simply implements the logic in the chip and the major timing features, but the real chip could, in the long term, show other subtle

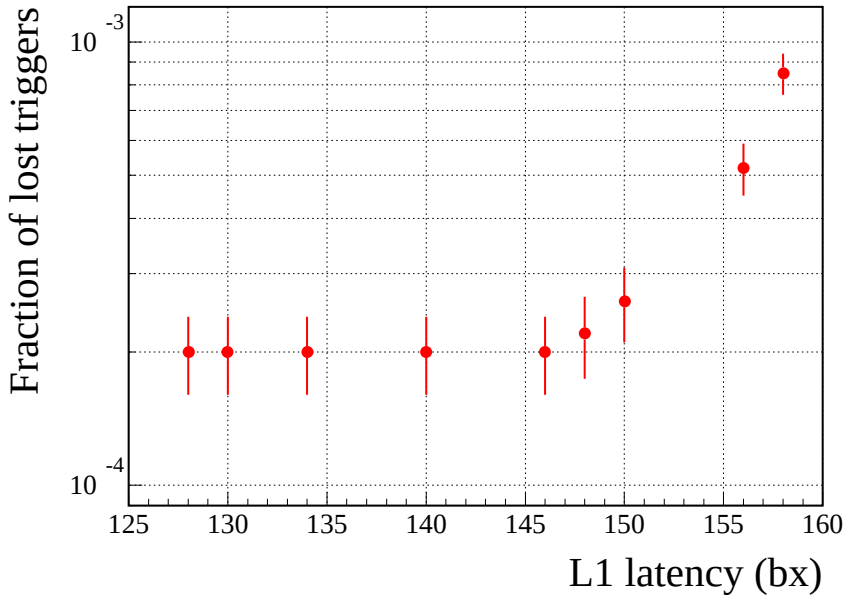


Figure 9: APV25: Fraction of triggers lost if a veto based on state machine emulation is applied as a function of L1 latency, at 100kHz average trigger rate.

features which are not implemented and that could be hard to implement, at least in the relatively simple C code available now. As an example, the time the pointers, especially the Read pointer, take to skip full cells, could be non–negligible if there are too many full cells to be skipped at one time. Another very interesting check could be testing the agreement between the simulation and the chip when triggers are vetoed. Indeed after having checked that the simulation reproduces the failures of the chip correctly, the next step is to see if they still agree after a veto. In such a test the veto would be simply performed by not sending that particular trigger which in the simulation causes the error.

Therefore the next big step to be taken is to build in the lab a full acquisition chain making use of the new PMC FED version [6]: data coming from a continuously running APV6 will be collected by the FED, analysed off-line and compared with the result coming from the simulation.

The final goal is to implement the logic in a Field Programmable Gate Array which will run in real time synchronous with all the APV chips in system. Not only can the FPGA be used to determine, on an event–by–event basis, when vetoing a trigger is needed, but it represents the main tool to monitor the global synchronization of the chips in the tracker system. To achieve both these purposes, an APV simulator could be located on each of the Front End Controller modules: trigger signals are indeed distributed from the TTC system via the FEC to the front–end. Therefore if a trigger needs to be vetoed, the FEC is the best place to perform this action. An intercommunication between FECs and the Front End Drivers must also be planned to allow information from the FPGA running the APV6 simulation to be sent to the FEDs to cross check the state of cells in the real chips.

7 Conclusions

An improved C++ code simulating the APV6 logic, predicts higher fractions of data lost when vetoing triggers to avoid buffer overflow. At 100 kHz and with a Level 1 Trigger latency of 128 bunch crossings the inefficiency foreseen now is 0.8%. The new figure is still acceptable if the latency is kept within 130 bunch crossings.

Further checks will be performed on the existing simulation before producing a new one suitable to be downloaded in an FPGA real time state machine simulation.

At the same time a first study of the behaviour of the next generation of the chip, the APV25, has been performed. A longer pipeline gives more freedom to choose the value of the L1 latency and the different hardware implementation of the pipeline, without paired cells, gives the optimal condition of marking and clearing cells. The overall result is an object which appears to be much more stable and efficient.

Acknowledgment

I would like to thank G. Hall, B. Macevoy and M. Raymond at Imperial College and to M. French and L. Jones at Rutherford Appleton Laboratory for the useful discussions we had about the subject.

References

[1] *"The deconvolution method of fast pulse shaping at hadron collider"*, S. Gadomski *et al.*, Nucl.Instrum.Meth.A320:217-227,1992;
"A novel technique for fast pulse shaping using a slow amplifier at LHC", S. Gadomski *et al.*, Nucl.Instrum.Meth.A326:112-119,1993

[2] *"APV6 pipeline emulations"*, M. Millmore, CMS Note 1997/045.

[3] *"SEQSI user's manual"*, M. Morrissey (RAL)

[4] *"0.25 μ m APV Development Status"*, M. French, talk given at the CMS Electronics Meeting, 01/02/1999.

[5] *"APV6 pipeline emulation"*, N. Marinelli, talk given at the CMS Electronics Meeting, 29/04/1998.

[6] CMS Front End Driver User's Manual, http://hepnts1.rl.ac.uk/CMS_fed/Default.htm.