STOCKHOLM UNIVERSITY

DEPARTMENT OF PHYSICS

A PROGRAMMABLE SYSTOLIC TRIGGER PROCESSOR FOR FERA BUS DATA

G. APPELQUIST, B. HOVANDER, B. SELLDÉN and C. BOHM

USIP Report 92- 04 September 1992

A Programmable Systolic Trigger Processor for FERA bus data

G. Appelquist, B. Hovander, B. Selldén and C. Bohm

Institute of Physics, University of Stockholm
Vanadisvägen 9, S-113 46 Stockholm, Sweden

ABSTRACT

A generic CAMAC based trigger processor module for fast processing of large amounts of ADC data, has been designed. This module has been realised using complex programmable gate arrays (LCAs from XILINX). The gate arrays have been connected to memories and multipliers in such a way that different gate array configurations can cover a wide range of module applications. Using this module, it is possible to construct complex trigger processors.

The module uses both the fast ECL FERA bus and the CAMAC bus for inputs and outputs. The latter, however, is primarily used for set-up and control but may also be used for data output. Large numbers of ADCs can be served by a hierarchical arrangement of trigger processor modules, processing ADC data with pipe-line arithmetics producing the final result at the apex of the pyramid. The trigger decision will be transmitted to the data acquisition system via a logic signal while numeric results may be extracted by the CAMAC controller.

The trigger processor was originally developed for the proposed neutral particle search experiment at CERN, NUMASS. There it was designed to serve as a second level trigger processor. It was required to correct all ADC raw data for efficiency and pedestal, calculate the total calorimeter energy, obtain the optimal time of flight data and calculate the particle mass. A suitable mass cut would then deliver the trigger decision. More complex triggers were also considered.

INTRODUCTION

Due to the large amounts of data involved, hierarchical trigger processing is a necessity in most large scale particle and nuclear physics experiments. Fast trigger processors are used to estimate the relevance of each event to the questions that are addressed by the experiment, so that the small sample of events that can be stored for future off line analysis contain pertinent data.

Hierarchical trigger processing uses repeated estimations of different relevant parameters to successively eliminate the least likely events. The decision from the first level trigger is sufficiently fast so that only a small amount of data, preferably none, is lost while the system is deliberating. The accepted events are then passed on to the second level trigger where data is subjected to a more thorough investigation. Larger decision latencies will be acceptable at this point since only a small fraction of the events are involved. Three trigger levels are often used where the first level may use analogue electronics, the second based on special purpose digital circuitry while the third uses general purpose computers. In general, higher level trigger processors process a larger fraction of the event at higher precision using more detailed models but requiring substantially longer processing time.

THE NUMASS EXPERIMENT

The development of the programmable systolic trigger processor (PSTP) module started to fulfil the requirements of a second level trigger processor for the proposed NUMASS experiment at CERN/SPS [1]. The aim of this experiment (fig. 1) was to search for new long-lived massive neutral particles produced in relativistic heavy ion collisions. The mass of the particles should be determined from measurements of the particle energies with a high resolution calorimeter and their time of flight from the production target to the calorimeter.

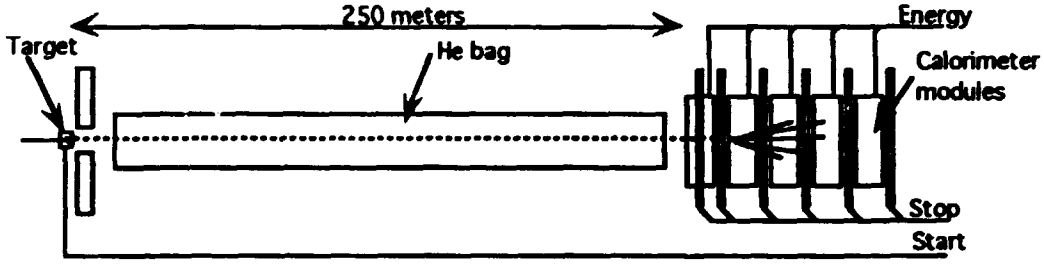

Figure 1. Outline of the NUMASS experiment

After the target, the forward directed particles produced in the collision travel along a ~250 m beamline and end up in a uranium/scintillator calorimeter (a test calorimeter built by the Zeus collaboration at DESY in Hamburg [2]). This calorimeter is made up of 5 modules in the beam direction (fig. 2) which allows for the insertion of layers of fast scintillator hodoscopes between the calorimeter modules for the time of flight (TOF) measurements. In addition, two extra TOF planes have been inserted in the middle of the first two calorimeter modules making a total of six TOF planes. The start signal for the TOF measurements is derived from a small scintillation counter at the production target.

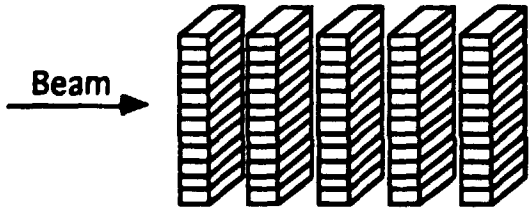

Figure 2. Calorimeter segmentation

The NUMASS second level trigger was designed to use the energy information from the calorimeter together with timing information from five of the six TOF planes. The scintillators in each calorimeter module would be segmented in 12 horizontal strips that are read out with a photomultiplier at each end. Four of the TOF planes would have 8 vertical scintillator strips and the other 16. All of them would be read out with PM tubes at each end. This gives 96 timing channels plus 120 calorimeter channels. In addition, the pulse height information from the timing channels would be used, giving another 96 channels. In total 312 channels form the input to the second level trigger.

The task of a second level trigger is to calculate the mass of the particles so that a masscut can be applied. The algorithm can be based on the relation:

$$M = \frac{E}{c^2}\sqrt{1-v^2/c^2} \quad \text{or} \quad M^2 \approx \Delta t E^2 \frac{2}{Lc^3} \qquad \ldots\ldots\ldots\ldots (1)$$

where M is the rest mass of the particle, Δt is the time of flight difference with respect to a particle that travels with the speed of light, L is the flight path and E is the total energy of the particle.

The total energy is obtained by summing the signals from the calorimeter and the time of flight can be calculated by combining the timing information from the TOF planes within the calorimeter with the start time from the target area. For a more detailed description of the algorithm used see below (PSTP applications).

DESIGN PRINCIPLES

If a single fast computational element is not sufficient, there are two ways to achieve the required processing speed: to use highly parallel independent computational structures or to use systolic (pipelined) processors. In the latter case the data is fed sequentially into the processor which consists of processing stations for each partial computation. These accept input data at each transition of a special processor clock delivering results at the following transition. Pipelined processing allows a high through-put with reasonable latencies. However, they must be designed to fit the special requirements of the experiment. Thus the algorithm used will, to a large extent, determine the hardware implementation of the trigger processor. This means that a modification of the algorithm may demand a complete redesign of the trigger processor. Also it means that such a trigger processor is dedicated to the experiment it was designed for and can therefore seldom be moved to other applications.

When designing the second level trigger processor for NUMASS one of the goals was to try to design a more general trigger processor that could be modified and that could be reused in other environments and other experiments as well. But, by necessity, a general trigger processor must be programmable. It should also be possible to increase the processing power since it is impossible to foresee future needs. Finally, it should be easy to thoroughly test the functionality of this rather complex system. This meant building a programmable and cascadable module with built in test facilities.

Programmability

For many trigger algorithms, and especially the one considered here, the processing power of standard microprocessors is not sufficient. Thus, the programmability must be implemented in some other way. One of the most interesting developments in digital electronics during the last decade is the field programmable gate arrays, e.g. the LCA circuits from XILINX [3]. These circuits consist of several thousand gate equivalents and the connections between these gates are stored in RAM memories so that different logic can be implemented by modifying the content of this RAM. With these circuits it is now possible to design reconfigurable hardware. Instead of using a microprocessor with

the algorithm implemented in software, which is slow, it can be implemented in hardware as a programmable systolic array.

Using field-programmable logic gives almost the same flexibility as microprocessor based designs, provided the algorithms are not too complex. The hardware can be modified on-line to adapt to changed requirements. It can also be reused in new, similar, environments by reconfiguring the gate arrays.

Cascadable

To be able to adapt the processing power to different needs, a general trigger processor should be cascadable so that several modules can be used together when required. A cascadable module should have more inputs than outputs so that the data streams can be concentrated within the module. Also, the input and output data streams should be of the same type so that the output from one module could be used as input to another.

Testable

When designing complex systems, testability must be introduced early in the design process. Otherwise testing the system will be, if not impossible, very time consuming. When designing with field-programmable logic, testability is automatically provided since the gate arrays can be configured with special test designs exercising different parts of the hardware.

DESIGNING THE PSTP MODULE

The algorithms considered are very well suited for pipeline processing. To achieve the processing power needed, several pipelines must be processed in parallel. Thus a suitable structure is the systolic array. The systolic array maps easily onto the cascadable hardware discussed above. The processing power can be increased by increasing the number of inputs used, given that data can be processed independently in the early stages of the processing pipeline. Or, fewer inputs can be used if the pipeline depth is increased, giving longer processing time but less hardware.

The PSTP module

A basic unit that can be used to build up the structures discussed should be powerful enough to operate independently. However, too large modules are not as flexible as several smaller. There are also physical limitations such as that the hardware should fit into a reasonably sized module box. Thus the choice of size and functionality of the basic module involves making a trade off between several conflicting requirements.

The ECL FERA bus, developed by LeCroy for ADC data transfers, was chosen for the main data link, since it was found adequate for the performance needed and since ADCs are a common input source. The control is conveniently handled by CAMAC.

The chosen structure (fig. 3) consists of 4 input modules, which obtain input data via the FERA bus, connected to one output module. The latter can send its data via FERA bus to a new layer of modules for more global processing. The output module is also connected to the CAMAC bus which is responsible for initialisation, programming and control. Data can be sent via CAMAC for monitoring and test purposes and the module can make an interrupt request by asserting the CAMAC "look-at-me" signal.

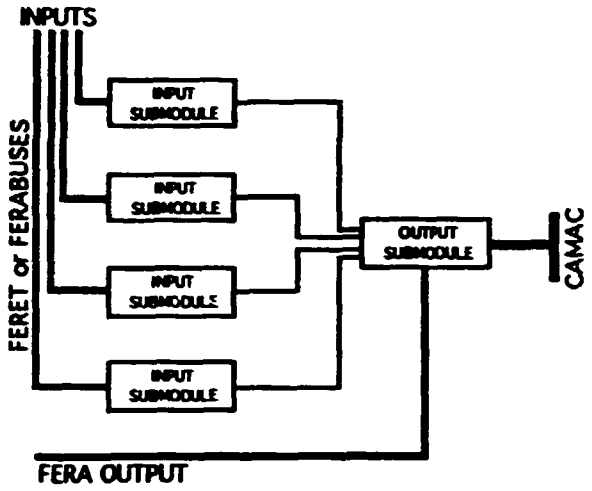

Figure 3. The PSTP module

The FERA bus is, in principle, capable of transferring 16 bit data words at the maximum frequency of 10 MHz. In this application, however, a transfer rate of 5 MHz was chosen to be on the safe side. The modules on the bus are read in a daisy chain fashion, which fits very well with the pipeline concept. However, the signals must first be converted from ECL to TTL levels before they can be processed in the input module.

Each of the five units contain an LCA (XC3064). The LCA is used for processing but also for controlling the rest of the hardware. There is no need for external "glue" logic since the LCA is sufficiently flexible to adapt to its environment.

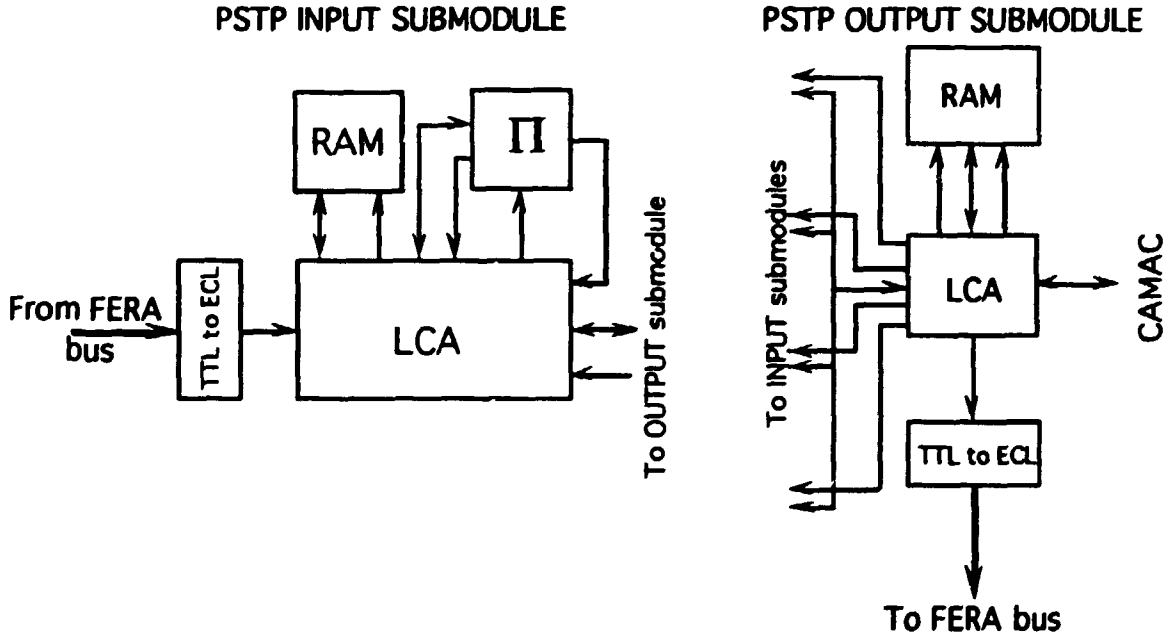

Figure 4. Input and Output sub modules.

In many cases it is desirable to subtract pedestals and multiply by a correction factor before the data is used in any further processing. To efficiently implement these operation the input module is supplied with a coefficient memory and a hardware multiplicator. The latter is capable of one multiplication per clock cycle. Figure 4 shows the hardware in the input and output sub modules.

In the output sub module the data from the four input sub modules is merged and further processed. This module also contains a RAM memory that can be used as a buffer memory or as a look up table to perform non-linear operations on the data. After completed pipeline processing, the results are available at the FERA bus output. From there the data can be transferred to other PSTP modules or be readout as a final result.

The FERA bus interfaces are good for high speed data transfers. But there is also a need for sending low speed control commands to the module. The CAMAC bus is simple but powerful enough for the purposes of control and slow data transfers. Through the CAMAC interface partial results can be read for monitoring purposes or full results if they are not produced at a too high rate. The CAMAC interface is implemented in the output LCA circuit. Although the LCA, in principle, can drive the CAMAC bus directly, it was decided to insert protective buffers between the LCA and the bus.

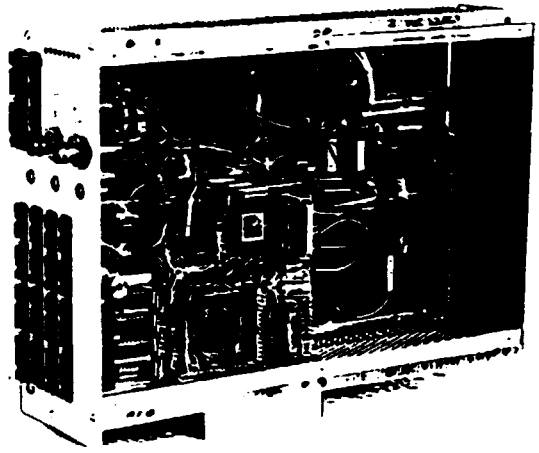

Figure 5. Physical layout of the PSTP module

Before using the module, all LCAs must be configured. This is done through a special interface connected to the CAMAC bus, containing a number of PAL circuits. Configuration data is sent to the LCAs one nibble at a time. Four LCAs can be programmed in parallel (4x4 = 16 bits) with different data at the full speed of the CAMAC bus (1 MHz). The time to program one LCA is typically a fraction of a second depending on the computer used to drive the CAMAC crate.

If any of the memories are used as constant memories, these have to be initialised as well. This is done by configuring the LCAs with special configurations for memory initialisation before giving them their final configuration.

DEVELOPMENT METHODS

Simulations were used extensively in the development of the PSTP module. The algorithms were first described at a high level in the VHDL [4] hardware description language to verify that a pipeline realisation of the algorithms was possible. This description was then refined to determine how it could be mapped into a hierarchy of PSTP modules and into their sub modules. In the last step, schematic descriptions of the operations performed inside the different gate arrays within the modules were made. These descriptions served as specifications when designing the gate array configurations.

The LCA circuits configurations were designed using the tools from XILINX. The configurations were then verified by hardware simulations where the PSTPs were described in terms of their hardware components and where the different LCA hardware models were initialised with their respective configurations. The Logic Modeling corp. LCA models [5] contain facilities for examining the internal states of the gate arrays. This

gives invaluable help when debugging the gate arrays since it is often difficult to monitor internal signals when working with the real hardware.

Test programs and LCA test configurations were developed to exercise different parts of the hardware. This made it possible to verify the module functionality after each hardware modification.

A PSTP APPLICATION

A typical application of the PSTP module is the second level trigger for the NUMASS experiment described above. Two different ways of implementing the trigger algorithm using PSTP modules were considered. One where speed was the primary goal and another where simplicity/price was given higher priority. The first solution required 6 PSTP modules and would generate the trigger decision after 16 μs. In the other solution only one PSTP module was required but then the trigger decision would be ready after 36 μs. The trigger decision will be based on a masscut and the requirement that the center of the energy distribution is well within the calorimeter. The algorithm to calculate the mass of the particle was based on (1) above and the basic steps were:

- Sum all the calorimeter signals to get the total energy the particle deposited
- Find the first (smallest) time from the TOF planes and take the difference between this value and the start time from the target area to get the time of flight .
- Use these results in (1) to get the mass of the particle and compare against limits.

In order to use PSTP modules to perform this calculation, a pipeline description of the algorithm must be found.

A fast parallel solution

In this solution each FERA module is connected to one PSTP input. In the experiment there are 21 FERA modules totally, each of them containing 16 channels. These modules can thus be served by 6 PSTPs (fig. 6), giving 3 extra inputs that can be used to combine the results from the different PSTPs (see figure 8 in the appendix for a description of how the PSTPs are arranged).

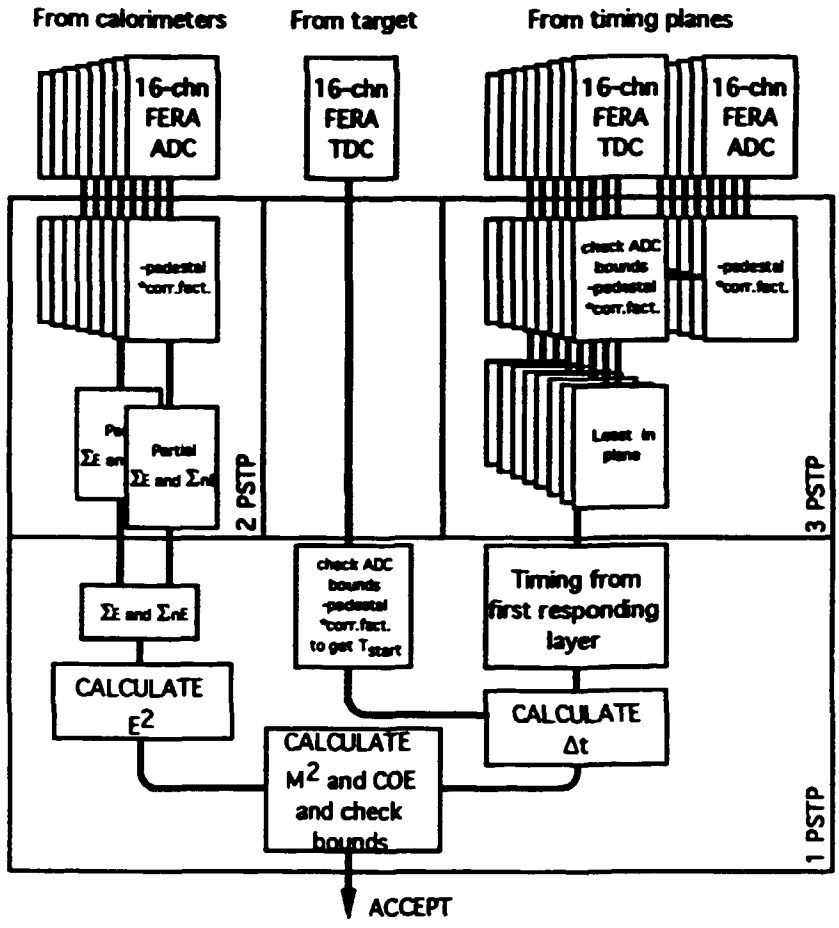

Figure 6. The NUMASS second level trigger algorithm implemented with 6 PSTP modules

A slower sequential solution

In the simplest case, the whole trigger algorithm is processed in one PSTP. To be able to do that, several FERA modules have to be connected in series on each PSTP input. Here all calorimeter ADCs were placed on one input, all TDC channels on one input and all the ADCs with pulse height information for the TDC channels on one input. The fourth input is used with a feedback loop from the FERA output, to utilise the full processing capacity of the PSTP.

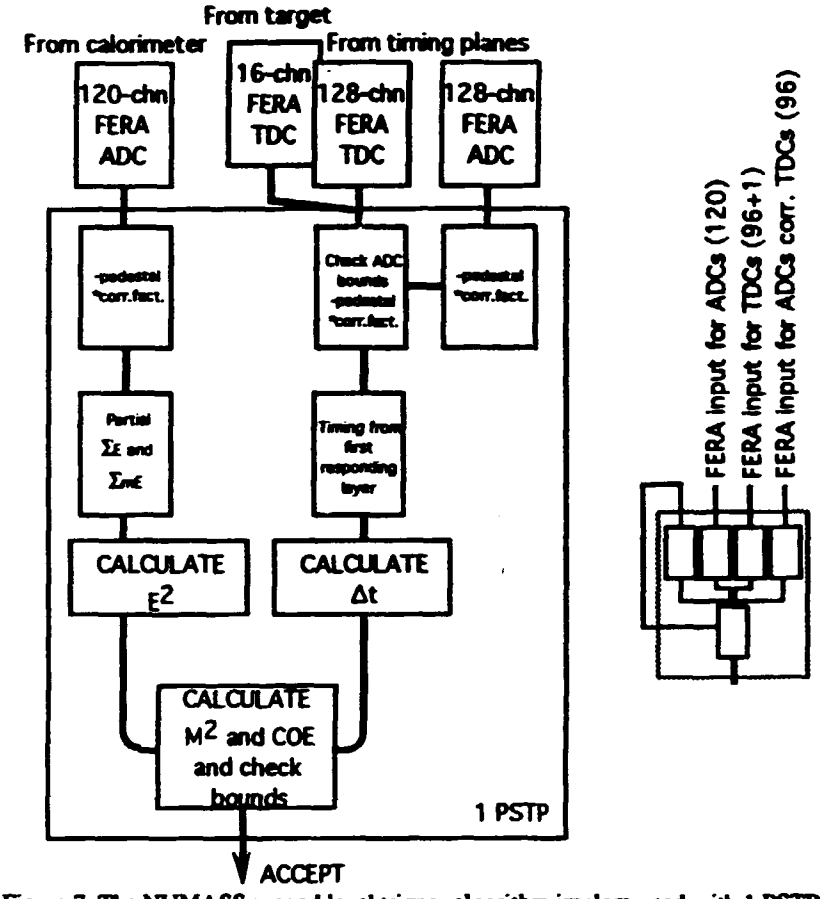

Figure 7. The NUMASS second level trigger algorithm implemented with 1 PSTP module

In this solution (fig. 7) the algorithm will be the same but the pipeline processing will take longer time since data is processed sequentially, to a larger extent, than in the previous solution. The trigger decision will now be available approximately 140 clock cycles after the conversion is completed or 36 μs.

Experiment status and prototype

During the development of the PSTP module the proposal for the NUMASS experiment was withdrawn. Therefore only one prototype PSTP module has been built (see figure 5). This module has been tested with LCA configurations that perform all the basic operations needed for the algorithm outlined above. Especially time critical operations like multiplication and using constants from the memory have been tested. A set-up where four FERA ADCs were read out and the total sum calculated, as in figure 9 in the appendix except for the last adder, was tested at CERN during a test run. Some errors were detected and later corrected by minor modifications of the LCA configurations.

An application of the PSTP concept to a nuclear physics experiment is currently being considered.

USER ENVIRONMENT

Test and configuration program

To test and configure the PSTP modules a program and a number of test designs were developed. The program runs either on OS-9 systems or on Macintosh with the MAC-UA1 software from CERN [6]. It is possible to automatically run tests of the system

where first a test design is downloaded and the program then sends suitable data to the module and checks the results. The test designs were made during the development process to test the module incrementally. These tests can also be used for module diagnostics and to localise and identify errors.

CONCLUSIONS

One PSTP prototype has been built and tested both testbench and in a experimental environment during a NUMASS test run at CERN. The OS-9 software environment developed for the test run worked well and the reliability errors discovered in the PSTP hardware have now been taken care of. An updated version is being planned which will include some changes suggested by the experiences from the prototype. This version will also include the last generation LCA circuits (the 4000 series) which will allow higher complexities and faster operation.

The general experiences from the applying the PSTP concept to an experiment like NUMASS show the great advantages with a flexible design. New alternative trigger schemes could be considered without changing hardware. Different triggers could easily be used during the same experiment using different configurations.

The development of PSTP configurations benefited greatly from the use of the advanced simulations tools from Mentor Graphics and Logic Modeling corp. Special PSTP development tools could be designed to simplify this process using a library of standard designs.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the stimulating collaboration with the Bern particle physics group, headed by Prof. Klaus Pretzl. The discussions with Prof. Kurt Borer and Dr. Daniel Frei and the OS-9 software guidance given by Dr. Fridolin Dittus were highly appreciated.

REFERENCES

[1] Proposal to Search for Neutral Strange Matter in Heavy Ion Collisions at CERN, CERN-SPLC/91-16, March 9, 1991, P 259.

[2] NIM A274 (1989)134.

[3] XILINX The programmable Gate Array Data Book, P. Alfke et al. 2100 Logic Drive, San Jose, California 95124.

[4] IEEE Standard VHDL Language Reference Manual, IEEE Std 1086-1987.

[5] Smart models library, Logic Modeling corp., 19500 Gibbs Drive/ P.O. Box 310, Beaverton, OR 97075.

[6] MAC-UA1, M68K-VME Macintosh Based Development System, CERN.

APPENDIX

This appendix contains a detailed description of the fast parallel algorithm outlined previously in this paper. The algorithm to calculate the mass of the particle is based on:

$$M^2 = \Delta tE^2 \frac{2}{Lc^3}$$

The basic steps to be performed were:

- Sum all the calorimeter signals to get the total energy the particle deposited
- Find the first (smallest) time from the TOF planes and take the difference between this value and the start time from the target area to get the time of flight.
- Use these results in (1) to get the mass of the particle and compare against limits.

Each FERA module is connected to one PSTP input, which means that each PSTP input will receive 16 values. The whole trigger system contains 6 PSTP modules connected in a tree structure (see figure 8).

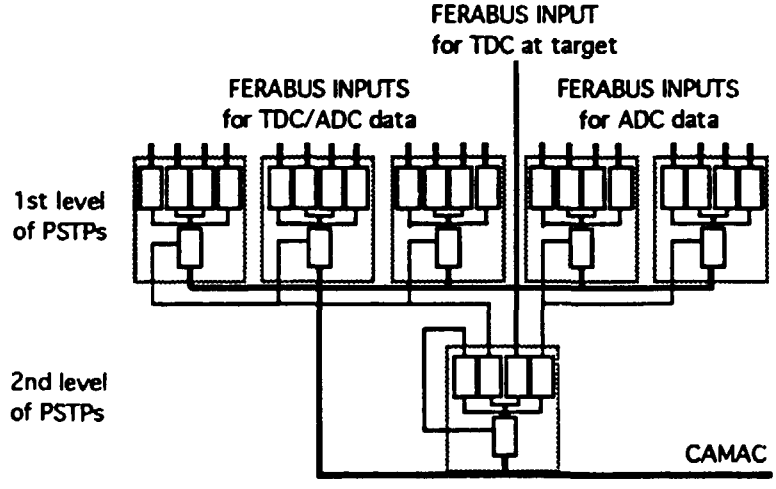

Figure 8. PSTP configuration for the fast parallel solution

Energy summation

The total energy is easily calculated in a pipeline by accumulating the values after an adder. To get the center of the energy distribution (COE) the weighted energy sum is also needed:

$$WE = \sum_{n=1}^{12} nE_n, \text{ and then } COE = \frac{\sum_{n=1}^{12} nE_n}{\sum_{n=1}^{12} E_n}$$

where n is the 12 horisontal strips in the calorimeter and E_n is the sum of the 5 strips at the same horisontal level but in different modules. Also the weighted energy sum can easily be calculated in a pipeline, an extra adding stage will provide the desired sum:

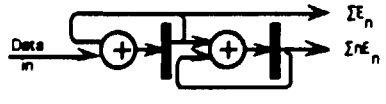

Figure 9. Pipeline processing to obtain the energy sum and the weighted energy sum

Given the configuration of the calorimeter, with 5 modules each containing 12 horisontal strips, it was found that the PSTP modules are best utilised if the calculation of the above sums is divided into several partial sums, each containing data from 4 horisontal strips (fig. 10). Each such partial sum is handled by the input sub modules of the PSTPs and then these partial sums are combined in the output sub modules to obtain the final results.

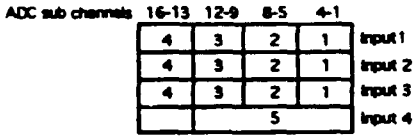

Figure 10. How data from the 5 calorimeter modules is assigned to PSTP inputs

Before data is used in any further processing, pedestals are subtracted from all incoming values and the results are multiplied by a correction coefficient . Figure 11 shows the processing in one input sub module that calculates the partial energy and weighted energy sums.

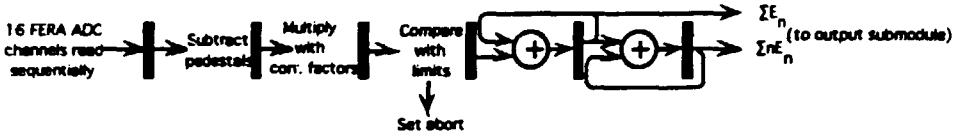

Figure 11. Pipeline processing in an input sub module

To split the energy summation into several partial sums that are combined is a trivial task but when it comes to the weighted energy sum, the partial weighted sums can not just be added. The expression for WE above can, however, be rewritten as:

$$WE_i = we_{i1} + we_{i2} + we_{i3} + 4e_{i2} + 8e_{i3}, \quad 1 \le i \le 5$$

where WE_i is the weighted energy sum for module i, we_{ip} is the partial weighted energy sums and e_{ip} are the partial energy sums.

Thus, channels 1 to 3 calculate partial sums as described above but in channel four, data from all 12 horisontal strips in module 5 is available and all the calculations for that module can be performed by the input sub module of channel four. The output sub module has then access to partial sums for module one to four and full sums for module 5. These values are combined in the output sub module to obtain the final results:

$$E = e_5 + \sum_{i=1}^{4}\sum_{j=1}^{4} e_{ij} \text{ and } WE = we_5 + \sum_{i=1}^{4}\{we_{i1} + we_{i2} + we_{i3} + 4e_{i2} + 8e_{i3}\}$$

The above equation shows the operations performed by one PSTP module. All scintillator strips are read out at both ends, thus another PSTP module is needed to process the values read out from the other end of all scintillator strips. The results obtained by these two PSTPs are combined in the next level PSTP (see below).

Time of flight calculation

For the timing values the algorithm is simpler. The problem is to find the timing value from the TOF plane with the first, i.e. smallest, value. This is done by first setting a register to the maximum value and then comparing all incoming data with the value in the register and storing the incoming value in the register if it is smaller than the current

value. Apart from this basic operation, three extra operations are performed: the timing values from the two ends of a scintillator strip are added, to get the mean value, compared against limits and vetoed if the corresponding pulse height values (coming from the ADCs) are too small. As before, pedestals are subtracted and all values are multiplied by correction coefficients before further use. The configuration of timing data easily maps onto the PSTP structure. There are 96 timing values and 96 corresponding pulse height values. If one FERA module (16 channels) is read by each PSTP input, three PSTPs are needed. Figure 12 shows the pipeline processing of timing data.

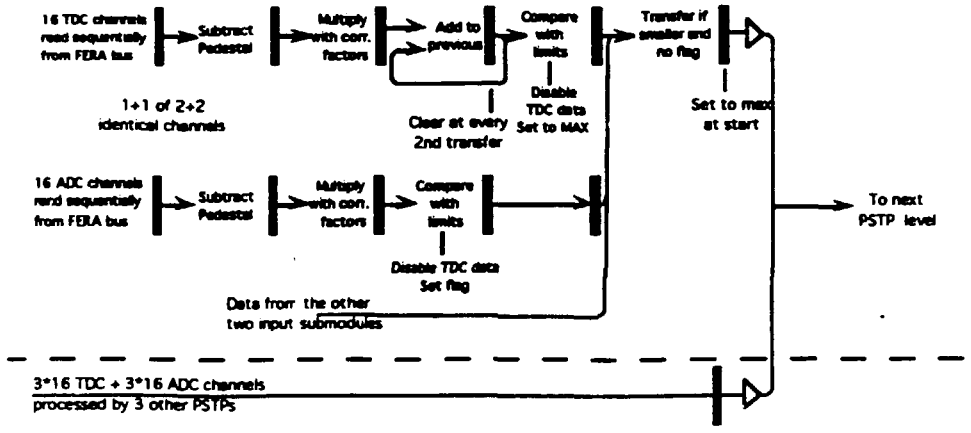

Figure 12. Pipeline processing of timing data in the first level of PSTPs

Calculation of particle mass

The results from the first 5 PSTPs, as described above, are combined in a final PSTP to obtain the desired particle mass. The two PSTPs that calculated the energies are connected to one of the inputs, the three timing PSTPs are connected to one input and the TDC that gives the start time is connected to one input. Finally, the fourth input is used by the output sub module, via a feedback from the FERA output, to perform the final multiplication of Δt with E^2. In the first input sub module the total weighted energy WE is calculated and a check that COE is within bounds is performed. Instead of dividing WE with E and comparing, E is multiplied by the limits and then the check is done, i.e.:

$$k_1 \leq \frac{WE}{E} < k_2 \quad \Leftrightarrow \quad k_1 E \leq WE < k_2 E$$

If this check is passed, the energy is multiplied by itself and E^2 is sent to the output sub module. In the second input sub module, the three timing values are read and the minimum value is selected. The start time, coming from the TDC at the target, is handled by the third input. Data that enters this PSTP module is of different types. Therefore the operations that have to be performed at a given step in the pipeline depend on the data type that currently is being processed. Such processing is not easily described in graphical format, therefore a table description is chosen:

Registers 1 Clocks		2	3	4	5	6
1	E_1				**First input sub module**	
2	E_2	E_1			**Processing of energy sums**	
3	WE_1	$E=E_1+E_2$				
4	WE_2	WE_1	k_1E			
5	E_1	---->	$WE=WE_1+WE_2$	k_1E	If $WE < k_1E$ Then Abort	
6	E_2	E_1		WE		
7		$E=E_1+E_2$		WE		
8		E	k_2E	WE	If $k_2E <$ WE Then Abort	
9			E^2			
10				E^2	----> Transfer E^2 to output sub module	
1	T_1	MAX	Transfer if register 1 is smaller		**Second input sub module**	
2	T_2	T_{min}	-"-		**Processing of TOF data**	
3	T_3	T_{min}	-"-			
4		T_{min}	----> Transfer T_{min} to output sub module			
1	T				**Third input sub module**	
2		T - Pedestal			**Processing of start time**	
3			$T_{start} = corr*(T - P)$		----> Transfer T_{start} to output sub module	
13	Δt				**Fourth input sub module**	
14	E^2	Δt			**Calculate** $M^2 = \Delta t * E^2$	
15		$M^2=\Delta t*E^2$	If $M^2 < M^2_{min}$ Then Abort,		---> Transfer M^2 to output sub module	
9	T_{start}				**Output sub module**	
10	T_{min}	T_{start}			**Accept event**	
11	E^2	$\Delta t=T_{start}-T_{min}$	----> Transfer Δt and E^2 to FERA output so that it can be read by			
12			the fourth input			
13						
14						
15						
16	M^2	-------> Activate readout				

Table 1. Pipeline processing in the final PSTP module

The trigger decision and the calculated mass will be available 38 clock cycles after the AD conversion is completed. The conversion takes 8 μs, giving a total processing time of approximately 16 μs.