EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

CERN-EP/86-157

October 10th, 1986

Test of a Data-Driven Trigger Processor for Experiment UA8

J. G. Zweizig, J. B. Cheze¹, J. Ellett, S. Erhan,
R. Jackson, M. Medinnis, P. E. Schlein, J. Zsembery¹
University of California^{*)}, Los Angeles, California 90024, USA.

(presented by J. G. Zweizig)

Geneva, 6 October 1986

Abstract: A fast trigger, based on principles of data-driven processing, has been designed and tested for CERN SPS-Collider experiment UA8. The electronics calculates the momentum vector of a track traversing the UA8 Roman pot spectrometer in 500 ns. When imbedded in the chamber TDC readout system, the total time between signal arrival in the counting room and result is 1.2 μ s. The calculations are performed by successive table look-ups and arithmetic operations executed by several examples of a single type of module configured in a data-driven architecture. The operation of the system is monitored with a package of control and diagnostic software. The trigger processor has been successfully tested at the SPS-Collider.

Invited Talk at the 3rd Pisa Meeting on Advanced Detectors
Castiglione della Pescaia, 2-7 June, 1986
(Proceedings to be published in Nucl. Instrum. & Methods)

¹ Centre d'Etudes Nucleaires-Saclay

^{*)} Supported by U.S. National Science Foundation grant PHY85-09175

1. Introduction

UA8 is an experiment at the SPS-Collider to study hard scattering effects in events produced by $p\bar{p}$ collisions with $\sqrt{s} = 630$ GeV which have an emerging quasi-elastic proton (or antiproton) in the final state. The final state proton is measured in a Roman pot spectrometer; the hard scattering is observed in the UA2 detector.

Such events, produced via inelastic diffraction:

$$p\bar{p} \rightarrow pX \quad (+ \text{ charge conjugate}) \quad (1.1)$$

result from the interaction of one beam particle with a soft ($x_F < 0.1$) color-singlet system called the Pomeron in the other beam particle.

Evidence of hard scattering observed in the final states of such processes would signal the occurrence of low order QCD processes between the partons in one beam particle with partons in a Pomeron of the other beam particle[1]. Ultimately, with sufficient data, the detailed study of such events should lead to an understanding of the internal structure and nature of the Pomeron, which plays a major role in high energy hadron scattering.

In the interaction region of UA2 (LSS4), we installed a spectrometer on the outgoing proton arm shown in Fig. 1, to measure recoil quasi-elastic protons and interfaced the readout of this spectrometer to the existing UA2 data acquisition system.

The search for jet structure in the diffractive system, X, of Reaction 1.1 requires a trigger consisting of coincidence between a recoil proton signature in the Roman pot spectrometer and total transverse energy in the UA2 calorimeter system, ΣE_T , above some required threshold. This trigger is logically OR'ed with all other UA2 triggers and initiates a readout of the UA2 detector and our spectrometers to the UA2 data tapes. Preliminary results from test data taken during the 1985 Collider run were presented at the 1986 Berkeley conference[2].

Our 1985 test run demonstrated that the use of scintillators alone to provide the recoil proton signature for the trigger leads to trigger rates several orders-of-magnitude larger than the desired physics. Partially this is because the rate of events of Reaction 1.1 with $x_F \approx 1$ dominates the more interesting events with smaller x_F (and larger mass of X). However, false triggers also result from other sources: (i) different low momentum particles traversing each of the Roman pots and creating undesirable coincidences; (ii) beam halo particles with the beam momentum traversing the spectrometer. These sources of uninteresting triggers must be eliminated in order to meet the rate requirements imposed by dead-time considerations.

To eliminate these sources of bad triggers, we proposed[3] to calculate (in real time) the momentum of a particle seen by the Roman pots and also to demand that its initial direction be consistent with having come from the interaction region. The data-driven[4] processor system described in this paper satisfies the above requirements and was tested successfully during the 1986 SPS-Collider run.

In a larger multi-particle spectrometer experiment, a more elaborate version of this type of processor[5] will serve as a "front end" in our VIRTUS parallel processor system[6]. VIRTUS is a system of programmable parallel processors in FASTBUS intended for complete on-line processing of events.

1.1 The UA8 Mini-Drift Chamber System

Each chamber in the Roman pot spectrometer shown in Fig. 1 comprises 6 MWPC wire planes with 2 mm spacing between wires. The system is operated with a TDC on each wire ("Mini-Drift") and yields a spatial resolution of $\sigma = 130 \mu\text{m}$ per plane. There are 2 planes with horizontal wires (y and y' , respectively) shifted half a cell from one another to solve the intrinsic left/right ambiguities. There are two planes (u and u') with $+7^\circ$ stereo and two planes (v and v') with -7° stereo. The six planes together provide about $55 \mu\text{m}$ point resolution in the vertical (bending) plane.

2. Momentum Algorithm & Implementation with Data Flow Architecture

Fig. 2 shows a sketch of a track traversing an SPS quadrupole magnet with chambers at positions 1, 2 and 3.

2.1 Momentum Algorithm

The starting point for the momentum calculation is the equation¹ for the vertical (y) position of a track propagating along a beam line from position i to position f:

$$y_f = R_{33}y_i + R_{34}y_i' + \Delta(T_{336}y_i + T_{346}y_i'), \quad (2.1)$$

where Δ is the fractional momentum difference from a central value (e.g., the beam momentum p_0), y_i and y_i' are the starting y position and slope, y_f is the final y position, and the coefficients R_{33} , R_{34} , T_{336} , and T_{346} are calculated from the SPS magnet parameters in LSS4 for a particle of momentum p_0 .

We solve this equation for Δ using the pot 2 position for the track's initial position, pots 2 and 3 to determine the slope, and the pot 1 position for its final position (the equation is invariant to reversals of the direction of the particle) and find the following result:

$$\Delta = \frac{p - p_0}{p} = \frac{y_1 - R_{33}y_2 - R_{34}y_2'}{T_{336}y_2 + T_{346}y_2'}. \quad (2.2)$$

Using the fact that $\Delta \ll 1$ and that $e^\Delta \approx 1 + \Delta$, we rewrite the equation as:

$$\begin{aligned} \Delta &\approx \ln(1 + \Delta) \\ &= \ln \frac{y_1 + (T_{336} - R_{33})y_2 + (T_{346} - R_{34})y_2'}{T_{336}y_2 + T_{346}y_2'} \\ &= \ln \{ y_1 + (T_{336} - R_{33})y_2 + (T_{346} - R_{34})y_2' \} - \ln \{ T_{336}y_2 + T_{346}y_2' \}. \end{aligned} \quad (2.3)$$

Since y' is a linear combination of the measured y positions in chambers 2 and 3, Eq. (2.3) may be rewritten as:

$$\Delta = \ln(y_1 + a \cdot y_2 + b \cdot y_3) - \ln(c \cdot y_2 + d \cdot y_3) \quad (2.4)$$

where a, b, c and d are known constants. The calculation of Δ may thus be reduced to a set of four calculations, each of which is the sum of two single variable functions:

$$A = a \cdot y_2 + b \cdot y_3 \quad (2.5)$$

$$B = c \cdot y_2 + d \cdot y_3$$

$$C = y_1 + A$$

$$\Delta = \ln(C) - \ln(B).$$

2.2 Calculation of y-intercept at Bunch Crossing Center

To calculate the y-intercept at the center of the beam-beam crossing region, y_0 , we use the thin lens approximation. This approximation uses the two track y positions measured after the magnet (y_2 , y_3) to determine a point at the center of the magnetic field (y_m), which is then used with the hit position in the first pot (y_1) to approximate the initial track trajectory. Specifically,

¹ The notation used here is that used by the TRANSPORT beam optics program [7].

$$y_m = \frac{z_3 - z_m}{z_3 - z_2} \cdot y_2 + \frac{z_m - z_2}{z_3 - z_2} \cdot y_3 \quad (2.6)$$

$$y_0 = \frac{z_m}{z_m - z_1} \cdot y_1 - \frac{z_1}{z_m - z_1} \cdot y_m$$

Eqs. (2.6) are of the same simple type as Eqs. (2.5).

2.3 Implementation

Each line of Eqs. (2.5) and (2.6) can be executed with the use of a module that has 2 independent digital inputs, a table look-up for each followed by a summation of the two table outputs. Moreover, since the calculations of A and B in Eqs. (2.5) may be performed simultaneously, the entire calculation can be made in 3 module delays. A module with these properties (called DIMSUM, for Dual Input Memory-Lookup and Summation) has been designed and built and is described in Chapter 3.

Eqs. (2.6) can also be evaluated using DIMSUM modules. This requires only 2 additional modules which function in parallel with the above momentum calculation, therefore taking no additional time.

A second new type of module (called EDSEL for Event Data Selector) is also needed to interface the raw wire chamber hit information (wire numbers and drift times from y and y' planes) to a DIMSUM module which finds the accurate y coordinate. The technical implementation of these modules[8,9] is discussed in the following chapters.

Symbolically, the algorithm for momentum calculation and selection is executed as shown in Fig. 3, which is a data-driven type of computer architecture[4]. The track y coordinates in the three chambers serve as input (by way of the EDSEL modules) to the first of four layers of DIMSUM modules, the last of which produces the Δ of Eq. 2.5. This quantity is input to one final DIMSUM module which selects Δ in some desired range. Since, as discussed below, the time spent in one DIMSUM module is approximately 100 nsec, the total time from EDSEL module output to trigger availability is about 500 nsec.

3. Dual Input Memory-Lookup and Summation Module (DIMSUM)

3.1 Physical Description

The DIMSUM module is housed in a single width CAMAC module. The modules are interconnected with standard 26-connector flat cables via the front panel, thus allowing arbitrarily complex calculations to be made in a multi-layered configuration. Input cable termination resistors are installed in sockets and are consequently easily removed to allow connection of a single output to more than one input.

3.2 Functional Description

The basic features of the DIMSUM module are shown in Fig. 4. The DIMSUM module has two operational modes. In the first of these, the on-line mode, each 12-bit input word serves to address a 4096 x 14 bit RAM. The first 12 bits of the contents of each addressed word are summed and the result is output on a 12 bit bus. In the second mode, the test mode, the RAM addresses are supplied by a sin-

Figure 4: Block diagram of DIMSUM Module

gle CAMAC controlled address register, and data can be entered to or read from either table over the CAMAC interface.

Module operation is switched between the two modes through the CAMAC interface. Mode changes may be accomplished either by setting a bit in the control register, or by setting or clearing the test mode bit using CAMAC Enable and Disable functions (F26·A2 and F24·A2 respectively). A back panel paired output (LEMO) is set by this test mode status and may be daisy chained with other modules to indicate that the system is not ready for use.

3.2.1 On-line Operations

In on-line mode, the output values are computed from the data on the two input cables as described above. All numeric data are transmitted as TTL compatible DC levels and therefore propagate asynchronously through the system.

Each interconnect cable contains a data valid flag in addition to the numeric data. This flag is originally set by each EDSEL module when data from an event has been latched. Each DIMSUM module asserts the data valid flag on its output cable approximately 100 nsec after the data valid flags on both of the input cables have been asserted. This delay is adjustable for each module by setting a potentiometer which is accessible through the front panel. The data valid signal is therefore tuned to propagate through the processor at the same rate as the calculation results. Presence of the flag at the output of the final module indicates that the calculation is complete.

The module contains parity and overflow checking circuitry for detection of hardware faults and illegal input data. The parity error and overflow signals are each available as Open Collector TTL lev-

els on a pair of LEMO connectors at the back of the module. The paired outputs enables daisy-chaining all the modules in the system to obtain a single system-wide error signal.

3.2.2 Test Mode Functions

The DIMSUM module is loaded, tested and monitored through CAMAC by way of a simple CAMAC interface on the board which is functionally compatible with both the CAMAC and the CERN REMUS standards. Control and loading of the module are accomplished through four registers accessed via CAMAC. Each register is assigned a unique CAMAC sub address for write access. However, since REMUS allows reading from only one CAMAC sub-address, a jumper option allows the internal address field of the control register to specify the register to be read in a CAMAC read operation.

In standard CAMAC, the data may be read or written either with single action calls, or, in the case of the memory data, with repeat mode block transfers. Since the Q response for valid commands is always 1, any such block transfer has to rely on LAM synchronization or a word count to end the transfer. ROMULUS/REMUS block transfers may be made using the LAM synchronized repeat mode. In accordance with usual ROMULUS procedure, a LAM is set by the Enable function (F26) and cleared after a successful read of the Status or Address register, or when an address overflow occurs on a read of one of the table memories. LAM response may be disabled by clearing the LAM mask bit in the Control register with a F24:A1 or by writing a zero to bit 14 of the Control register.

Software utilities exist for emulating a DIMSUM system as well as loading, testing, and facilitating the repair of such a system. These are described below in Chapter 5.

4. Technical Implementation of Chamber Readout and Trigger

Fig. 5 shows a block diagram of the TDC readout and trigger system in the configuration used for the on-line calculations in UA8.

4.1 The TDC Readout System

The trigger implemented for UA8 works in three stages. First, a signal synchronized with the bunch crossing (RF) generates a gate for the chamber TDC system and all CAMAC instrumentation. ECL logic signals from each "struck" wire of the chamber system which arrive at the inputs of the 32-channel TDC modules while the gate is asserted cause a 4-bit Gray-code clock signal to be latched. This constitutes the time digitization.

One 32-channel TDC receives the signals from one wire plane. All six TDCs for a given chamber are placed in a single crate such that the y and y' plane TDCs are read out first. Each crate also contains a readout module, which sends the gate and clock signals to the TDC modules over the crate backplane, and controls the sequential readout of the hit data into a memory module. All TDC crates are read out in parallel.

The second stage of the trigger starts 80 nsec after the arrival of RF, when all signals from the Mini-drift chambers have been time-digitized. At this time, the parallel readout from the TDC crates is started, causing all chamber hit data to be transferred to the memory modules at the rate of 75 ns per hit. During this readout process, the hit data from the y and y' planes are picked up and latched by the EDSEL modules in the EDSEL-DIMSUM crate shown in Fig. 5.

There is one memory module per chamber crate, which can store up to 31 16-bit hit words. All the memory modules are located in a single central memory crate. The data in the memory modules are accessible by the UA8 on-line computer (for diagnostic and control functions) via a Control module located in the memory crate.

The final stage of the trigger logic is provided by the chamber hit consistency check and momentum calculation of the EDSEL-DIMSUM system, as described in the following section.

Transfer of data to the UA2 on-line computer takes place through a so-called TBRC module[10] which resides in the memory crate and interfaces the crate bus to a CERN standard ROMULUS VBUS. Upon receipt of a Prepare and Go signal over the ROMULUS VBUS, the TBRC scans through all the memory modules and transfers the hit data over the VBUS (under internal control).

4.2 EDSEL-DIMSUM Operation

The final stage of the trigger commences when the three EDSEL data latch modules determine that the hit pattern in the selected planes (y, y') is consistent with having resulted from the passage of a single track through each chamber. When this occurs, Data Valid signals are set at the outputs of the EDSEL, and the hit information and accompanying validity indicator begin to propagate through the DIMSUM-based computing structure.

The precise y and y' coordinates for each chamber are determined by one EDSEL module followed by a DIMSUM module. As shown in Fig. 5, the EDSEL module connects to the cable which runs from the TDC readout module for a given chamber to the memory module. During the readout from TDC crates to Memory crate, an EDSEL module counts the number of hits with valid drift times in the y and y' planes and latches the hit address and time code for one of the hits from each of these planes. A Data Valid signal is generated only for events with one or two adjacent hits in both the y and y' planes. In the case that either of these planes contains two hits, the information from the hit with the lesser time code is stored, and the other hit is ignored.

The latched hit addresses, time codes, and ambiguity variable (which indicates the drift direction) are sent over two cables to a DIMSUM module for calculation of the y coordinate for the chamber. The first of these cables carries the address of the hit y -plane wire. This is converted to a y position by table look-up in the DIMSUM board. The other cable contains the time codes and ambiguity variable from which is computed a signed drift distance. The sum of these two quantities — which is the y coordinate — appears on the output of the DIMSUM module.

4.3 Limit Comparison and Final Trigger Calculation

A single trigger signal is derived from the results of the DIMSUM calculations by comparing the momentum variable and the y intercept position to one or more sets of limits² and logically combining the results of these comparisons. This step is implemented using a further DIMSUM module. In this scheme, the y intercept position (calculated using Eqs. 2.6 as described in Chapter 2), and the momentum variable are each sent to one input of a DIMSUM module. Each of the multiple triggers are assigned a pair of adjacent bits in the DIMSUM data word. The tables are set such that the low order bit of each trigger pair is set on for all input values which are valid for the associated trigger. The paired hits at the output of the adder then contain the logical AND and exclusive OR of the two range selections. One or both of these logical combinations are used by the trigger electronics to indicate the existence of a valid track.

5. Diagnostic Software

5.1 The System Definition Files.

All modules and interconnections are specified in an On-Line Definition File which is used by the following software packages:

1. Table Loader / Verifier
2. Connection Verification Routine
3. Module Diagnostics
4. System Emulator.

5.2 Online Definition Files

This file describes the processor configuration completely. Fixed format records describe the CAMAC address of each EDSEL or DIMSUM module, the contents of the DIMSUM tables and all interconnections between the EDSEL and DIMSUM modules. It therefore provides a functional description of the system which can be compared with the system itself.

² Several sets of momentum limits are foreseen to provide separate triggers (which are individually selectable by the fast electronics) for elastic scattering, inclusive diffraction, high mass diffraction and double-Pomeron exchange.

5.3 The Table Loader/Verifier Program

The table loader and verifier program fills the DIMSUM tables and then reads them back to verify that they were properly loaded. All DIMSUM modules listed in the configuration file are loaded and verified in turn. The table is built up in memory based on the table description in the configuration file and loaded into the DIMSUM with a single DMA transfer. The contents of the tables are then read back and compared to the original. If any errors are found, an error message is printed and the program terminates.

5.4 Connection Verifier

The Connection Verifier insures that all interconnections between EDSEL and DIMSUM modules are installed correctly. The program sets each EDSEL module to produce an alternating 1/0 bit pattern and then follows the pattern through all DIMSUM modules downstream of the EDSEL. It then reverses the pattern and repeats the verification procedure to insure that all signals in each of the cables are making good contact. Any bad connections are reported by the program.

5.5 Module Diagnostics

A diagnostic routine exists for testing any DIMSUM or EDSEL module inside the system. All functional units and data paths in the modules are tested. If an error is detected, it is reported to the operator and the program optionally enters a loop repeating the failing test to allow the repair of the module.

5.6 System Emulation

A library of routines exist for facilitating the emulation of the system. Fortran callable routines are available to:

1. Build the system configuration data base
2. Locate functional elements using symbolic names
3. Emulate a DIMSUM module.

The following routines are also planned:

1. Cycle hits through an EDSEL module
2. Examine the contents of an arbitrary cable.

6. Results of Test Run – May 1986

The last two weeks of the Spring 1986 collider run afforded us our only opportunity to test the trigger system before the ACOL startup scheduled for the Fall of 1987. The UA2 detector had by this time already been removed so that it was not possible to trigger simultaneously with the trigger processor and the transverse energy deposition in the UA2 central calorimeter. Nonetheless, we were able to test the trigger processor and study backgrounds under conditions very similar to those we will encounter during the data taking stage.

Detectors were installed in the four pots above the outgoing proton beam as well as the four pots below the outgoing $\bar{p}$ arm. At that time, we had enough DIMSUM and EDSEL modules available for one of the four-pot spectrometers. We chose to equip the outgoing proton arm with the processor since the background from beam halo is more severe in this arm and is therefore closer to what we expect starting in 1987.

Our first level trigger consisted of a three-fold coincidence of the scintillators in the first three pots of the proton arm and a veto formed from the scintillators in the $\bar{p}$ arm. The timing of the veto was set to cut out protons in the beam halo which traverse the $\bar{p}$ arm spectrometer in the opposite direction from tracks from valid interactions and are then focussed into the proton arm spectrometer.

A second level trigger was formed by the EDSEL modules and demanded valid hit combinations in the first three of the chambers of the proton arm spectrometer, as described in Chapter 4. The detector in pot 4 served mainly to provide redundant information for the off-line analysis and was not used either in the trigger or in the analysis presented here.

Data from events satisfying the second level trigger were written on tape and further analyzed off-line. In addition to the raw chamber information, each event record also included the output data from every DIMSUM module, including the momentum estimate and DIMSUM trigger decision (although this was not used in the on-line trigger).

Figure 6: y -coordinate of track at $z = 0$. The measured track extrapolated to the bunch-crossing center at $z = 0$ separates those arising from beam-beam interactions from background tracks.

Fig. 6 shows the distribution of the vertical coordinate (y_0) of reconstructed tracks projected to the bunch-crossing center. The tracks with y_0 greater than zero are mainly from proton beam halo which escaped the halo veto. The narrow peak centered at $-.15$ cm is from valid interactions. Although we did not calculate y_0 on-line during the Spring run, such a calculation could be made by the method described in Chapter 2. Thus, we will be able to considerably improve our halo veto by rejecting those tracks not originating from the interaction region.

The DIMSUM momentum spectrum for those tracks in the peak of Fig. 6 is shown in Fig. 7(a). This spectrum was also available on-line (except for the y_0 cut) where it served as a powerful diagnostic tool. Fig. 7(b) shows the momentum spectrum from tracks reconstructed off-line from the same events used for Fig. 7(a). The peak in Fig. 7(a) is broader than that of Fig. 7(b), and also displaced from the nominal beam momentum of 315 GeV. This is because the correct geometry constants were not known at the time of the test run. We expect the on-line resolution to be nearly as good as the off-line estimate when the correct constants are loaded into the DIMSUM tables. Nevertheless, the on-line resolution for the test run would already have been adequate for triggering purposes.

When the data of Fig. 7(b) were recorded, the processor was set to trigger on tracks with momentum larger than 265 GeV/c (even though the processor trigger was ignored during the test run, its status was written into the event record). The shaded region of Fig. 7(b) is the spectrum of those events which satisfied the processor trigger. Except for a small region near the momentum cut, there is no trigger inefficiency and furthermore, there are no events in which the off-line and on-line momentum estimates grossly disagree.

We thank the SPS Division for the effort required to make the UA8 1986 Collider run possible. The support of the CERN administration is gratefully acknowledged. We thank David Plane for his work on the momentum algorithm. Two of us (PES and JGZ) particularly wish to thank Bruce Knapp and Bill Sippach for many helpful and friendly discussions about their work with data driven processors.

References

- [1] G. Ingelman and P. Schlein, *Phys. Letters* 152B (1985) 256.
- [2] R. Bonino et al. (UA8 collaboration), contribution to the XXIII International Conference on High Energy Physics (Berkeley, 16-23 1986).
- [3] S. Erhan et al., Proposal to the SPSC P208, CERN-SPSC/84-82.
- [4] W. Sippach, G. Benenson, B. Knapp, *IEEE Trans. Nucl. Sci.*, NS-27, 578 (1980); H. Cunitz, Y. Hsiung, B. Knapp, W. Sippach, "Data Driven Processing", in *Proceedings of the International Conference on Instrumentation for Colliding Beam Physics*, SLAC Report 250, June 1982.
- [5] For large-scale applications of these methods, see: L. Borten et al., "A Data Driven Parallel Pipelined Hardware Reconstruction Processor" (page 45) and D.M. Kaplan, "A Parallel, Pipelined Event Processor for Fermilab Experiment 605" (page 31); *Proceedings of the Symposium on Recent Developments in Computing, Processor and Software Research for High-Energy Physics* (Guanajuato, Mexico, May 8-11, 1984).
- [6] J. Ellett, R. Jackson, R. Ritter, D. Yaeger and J.G. Zweizig, "VIRTUS - A System of Parallel Processors in FASTBUS"; UCLA-VIRTUS/86-04.
- [7] K. L. Brown et al., TRANSPORT, A Computer Program for Designing Charged Particle Beam Transport Systems, CERN 80-04.
- [8] J. Ellett, R. Jackson and J.G. Zweizig, "The Dual Input Memory-Lookup and Summation Module, DIMSUM"; UCLA-VIRTUS/86-02.
- [9] J. Ellett, R. Jackson and J.G. Zweizig, "The Event Data Selection Module, EDSEL"; UCLA-VIRTUS/86-03.
- [10] J. Ellett, R. Jackson and J.G. Zweizig, "The TDC Buffer Readout Controller"; UCLA-VIRTUS/86-01.