Computer Physics Communications 57 (1989) 321–324
North-Holland

THE ZEUS THIRD-LEVEL TRIGGER SYSTEM

ZEUS CDAQ Collaboration

S. BHADRA, M. CROMBIE, D. KIRKBY and R.S. ORR

The Institute of Particle Physics and Department of Physics, University of Toronto, Toronto, Ontario, Canada M5S 1A7

The ZEUS experiment will have a three-level triggering system. The first two levels involve the local processing of component data. The third level will make a decision based on the global information from an event. It is designed to accept events at 100 Hz (10 Mbyte/s) and reduce this to around 3 Hz. Furthermore, we require that the third-level processors should be Fortran programmable and have a development environment that allows the utilization of the offline analysis code.

1. Introduction

The three levels of triggering in ZEUS are defined by the levels of decision time available. The 1st level has a decision time on the microsecond scale. It must reduce a primary rate of around 500 kHz to an output rate of 1 kHz. In general, 1st level trigger systems are fast, purpose built devices. They sacrifice flexibility, and the ability to correlate different event elements, to speed. For example, the ZEUS calorimeter 1st level trigger [1] is a pipelined "adder tree". Each process in this pipelined trigger repeats every 96 ns (the beam crossing time). The decision time is 5 μs, the length of the pipeline. The trigger algorithm is based on a calculation of jet quantities such as E_T, p_T, etc.

In general terms, a 2nd level trigger is one implemented in terms of programmable processors. The Zeus calorimeter 2nd level trigger [1] is an example. It is designed to reduce the trigger rate from the 1st level to around 100 Hz, which implies a decision time of a few milliseconds. The hardware implementation is a network of transputers [2]. Although the transputers are quite powerful processors, there is insufficient time to exploit fully the global features of the event data, or to use iterative algorithms such as are necessary in track fitting.

Whereas the 1st and 2nd level triggers use pipelining and some parallelism within an event, the third-level trigger exploits the concept of "event parallelism". As described in ref. [2] the full events are built into the third-level trigger crates. This is the first point at which all the information on a given event has been assembled in one processor. Each processor can work on a different event in a completely independent fashion. If there is sufficient computing power, this "farm approach" allows one to conceive of filtering the data through the full gamut of algorithms normally associated with the offline processing of data. The third-level system is designed to reduce the 100 Hz of event data from the 2nd level trigger to around 3 Hz.

2. Requirements on the system

The major design requirements on the system are:

1. The input bandwidth should be sufficient to accommodate the expected data rate (event rate × event length). From the design of the rest of the experiment a reasonable maximum for this is (100 Hz × 200 kbytes), or 20 Mbytes/s.

2. The system must have sufficient processing power to reach the requirement of reducing the

2nd level rate of 100 Hz to 3 Hz. Since iterative offline code will be run, the system should be able to provide several seconds of, for example, VAX 11/780 equivalent processing time per event.

3. Finally, and perhaps most importantly, these requirements must be achieved while providing a software environment which is easily accessible to the typical "offline user".

The input bandwidth is satisfied by single board VME-bus based processors; our design comprises six VME crates. The events are built via a transputer network [2] into a dual port memory (DPM) in each crate. This DPM acts as a source of events for the processors in the crate. It can provide data at around 10 Mbytes/s to the processors. Six crates then give a total data input rate well in excess of the design requirement.

As will be discussed shortly, the processors chosen are second generation Fermilab ACP processors. This choice leads us to choose the Fermilab branch bus [3] as a means of outputting the processed events. The hardware architecture, at crate level, is shown in fig. 1a. The events flow from the event builder into the card labeled 2TP [4]. This has two INMOS T800 transputers communicating with the DPM, which is interfaced to the VME-Bus. The VBBC/BVI [5] combination allows any processor in the crate to become master of the branch bus, and thus output data. The VBBC has been tested, and shown to be capable of driving the branch bus at 20 Mbytes/s.

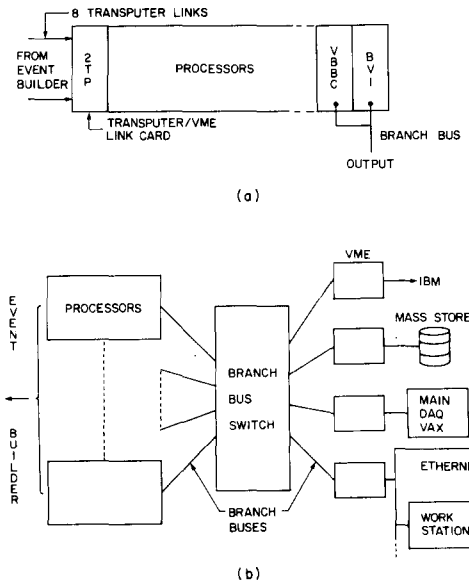

Fig. 1. (a) The hardware arrangement at the single VME crate level. (b) Block diagram showing the overall hardware arrangement of the system.

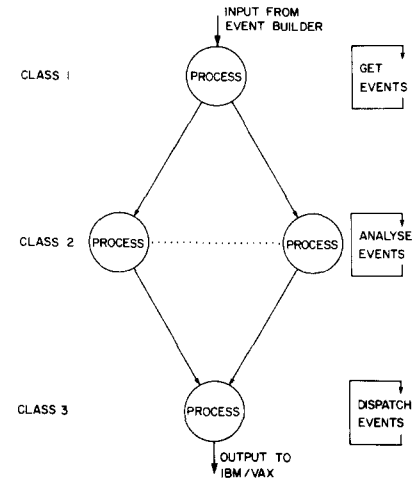

Fig. 2. Diagrammatic model of the software structure within a single crate.

The event transfer software is modeled in fig. 2. We intend to utilize the Fermilab co-operative process software [6], which is under development for the ACP-II. The software model for this system comprises several co-operating processes. These processes may all run on one physical processor, or they may be distributed over many pieces of hardware, a feature which will be extremely useful in the development environment. Figure 2 shows the processes divided into three classes: input, output, and multiple copies of the analysis code.

In each crate a single processor is devoted to running the class 1 and class 3 I/O processes. The remaining processors in the crate run the analysis code. Each of the analysis processors must exchange 8 messages per event with the processor running the I/O. At an estimated 250 μs per event, and 15 processors taking 0.5 s per event, this occupies 0.06 s out of each second; a negligible proportion of the VME-bus bandwidth. The data-transfer time budget per crate is equally comfortable. Assuming a maximum of 200 kbytes/

event at 100 Hz, distributed over 6 crates, the input rate per crate is 3.3 Mbytes/s.

The choice of processor board was partly decided by the fact that our original proposal was to use 100 Motorola MC68020 based ACP-I boards [7]. This technology is already somewhat outmoded. By the start-up time of ZEUS it would have been woefully lacking in cost effectiveness. The hardware requirements of the system are:

- total system CPU power of 1000 MicroVax equivalent units of processing (MVUP);
- total system cost of order $300 000;
- I/O bandwidth per processor of 20 Mbytes/s;
- support of a full operating system, allowing a workstation environment.

The second generation Fermilab ACP-II [8], which is based on the MIPS R3000 chip set, satisfies these requirements:

- MIPS R3000/R3010 chip set giving a power of 20 MVUP. This is from Fermilab benchmarks on the R2000 extrapolated to the R3000. It is confirmed by benchmarks run on the MIPS R2000 DECstation 3100. Fifty such processors would satisfy the total CPU power requirement of 1000 MVUP.
- The ACP-II has a full VME interface. This allows the board to be either a VME master or slave, with block transfers at 20 Mbytes/s.
- The board has at least 8 Mbytes of RAM, which should be sufficient to accommodate the offline code. The RAM is extensible to 24 Mbytes.
- The hardware configuration of the board allows it to support a full UNIX operating system. This allows the support of task to task communication via TCP/IP, and disk access from the nodes via NFS. These features are central to the implementation of the co-operative process software. A workstation like environment is supported in terms of Fortran-77, and telnet, which allows a user to login to an individual node.
- The cost, although not yet finalized, should be substantially less than $10 000 per board.

The requirements on the software environment that the system provides are as important as the actual performance itself. It must be such that one can realistically expect the utilization of the processing power. The algorithms for finding and matching event elements will be central to the performance of the third-level system. Presumably they will represent tens of man-years of software development in the offline environment. The third-level must be able to use this code essentially "as is". To assume that one could duplicate this code for the third-level, or embark on major conversion work, would be naive.

The division of the I/O and analysis into autonomous co-operating processes has great advantages for the software development environment. The analysis code must be developed offline. The software model shown in fig. 2 allows one the replace the online class 1 and class 3 processes by "offline" processes which read and write from mass storage, rather than from the online event stream. The analysis code then sees the same software interface during the development and online running. A further feature of the model is hardware independence of the software. Analysis code can be developed on any platform which supports the co-operative process software environment. The "offline" class 1 and class 3 processes can run on the same machine as a single copy of the class 2 process under development. In the online environment the only change from this is that the class 1 and class 3 processes change, but the software interface to the class 2 process remains unchanged. The class 2 process than runs as multiple autonomous copies on the analysis processors.

3. Overall system architecture

The overall system architecture is shown in fig. 1b. The data is transported from the third-level processors, via a Fermilab branch bus switch [9], to multiple destinations. The switch is a true cross bar switch, and thus allows several concurrent data paths to run at the full 20 Mbytes/s. Different processors can be concurrently transferring data to the IBM (for long term archiving), to local mass storage (for short term archiving), to the DAQ VAX (for small sample data monitoring), and to online workstations. In addition to allowing these concurrent paths, the switch automati-

cally provides arbitration between processors in different crates accessing the same destination.

The switch architecture also satisfies a further design requirement. During software development, the third-level system must be available as a truly multi-user system. This requirement is satisfied by the branch bus protocol. However, the third-level system should be available for offline development even when the experiment is running. The use of the switch isolates the data path to a crate being used for development from the main online event stream. This should ensure a more robust system.

Acknowledgements

We wish to acknowledge the helpful advice and comments of our many colleagues in the ZEUS collaboration. We are very grateful to the Fermilab ACP group for their help, advice, and freely provided information. This work is supported by a grant from the Natural Sciences and Engineering Research Council of Canada, through the IPP of Canada.

References

[1] The ZEUS Detector, Status Report 1987, The ZEUS Collaboration, DESY PRC 87-02 (September 1987).
[2] L.W. Wiggers and J.C. Vermeulen, Comput. Phys. Commun. 57 (1989) 316.
[3] Branch Bus Specification, Fermilab ACP (21 September 1987).
[4] H. Boterenbrood et al., Transputer based developments at NIKHEF, ZEUS-Note-88-002 (January 1988).
[5] VBBC Manual, Fermilab ACP (1988).
[6] ACP Co-operative Processes User's Manual, Fermilab (21 February 1989).
[7] ACP 68020 CPU Users Manual, Fermilab ACP (13 August 1986).
[8] T. Nash, Comput. Phys. Commun. 57 (1989) 47.
[9] Bus Switch Users Manual, Fermilab ACP (1 March 1988).