Nuclear Instruments and Methods in Physics Research A 332 (1993) 253–262
North-Holland

NUCLEAR INSTRUMENTS & METHODS IN PHYSICS RESEARCH
Section A

The Eventbuilder of the ZEUS experiment

Ulf Behrens, Lars Hagge and Wolfgang O. Vogel [1]
Deutsches Elektronen-Synchrotron, Hamburg, Germany

Received 15 February 1993

The Eventbuilder of the ZEUS experiment is a real-time parallel data formatting and transport system. It combines data flows originating from various detector components and transfers them to the third level trigger processor farm. The Event builder is based on an asynchronous packet-switching transputer network and uses transputer links for bulk data transfer. A high-speed 64×64 custom made crossbar switch allows dynamic linking of any detector component to any branch of third level processor nodes, offering a bandwidth of more than 24 MB/s over a distance of 100 m. The use of structured system development techniques (SA/SD) resulted in a flexible and well-partitioned system structure and guaranteed that all requirements were met.

1. Introduction

HERA, the new electron–proton colliding facility at DESY, started operation for physics measurements in spring 1992. HERA provides electron–proton collisions at a cm energy of 310 GeV. The ZEUS collaboration constructed a detector for one of HERA's interaction regions.

Challenges for the ZEUS experiment are the short interval between beam crossings of only 96 ns, more than 250000 readout channels and an initial raw data rate exceeding 10 GB/s. The data rate has to be reduced by a factor of at least 10^4 before data recording. This imposes strong requirements on the trigger and data acquisition system.

The trigger and data acquisition system of the ZEUS experiment is a highly-parallel distributed real-time system. It consists of several independent readout systems and three trigger levels for data filtering. Its central part, the ZEUS Eventbuilder, combines and formats the data flows originating from the various readout systems.

The Eventbuilder is subject to the highest data rate within the ZEUS data acquisition system. Due to its connections to almost all parts of the data acquistion system, the Eventbuilder is also an important tool for system analysis and diagnosis. This article describes the development of the ZEUS Eventbuilder, gives a brief overview of its hardware and software architecture and reports first results on the performance of the Eventbuilder and the data acquisition system.

2. Overview of the ZEUS trigger and data acquisition system

The ZEUS detector comprises several independently operating detector components, each of them equipped with their own so-called component subsystem (CSS). Component subsystems contain the "front-end" electronics required for the component control and readout. They interface to two levels of global trigger processors and the Eventbuilder. The layout of the ZEUS trigger and data acquisition system and the data throughput at its components are shown in fig. 1. The component subsystems of the ZEUS experiment are listed in table 1.

Once a detector component has been read out, the data are stored in a 5.5 μs first level trigger pipeline and analyzed by a local first level trigger processor. The results of the different component subsystems referring to the same beam crossing are input to the global first level trigger (GFLT), which computes an overall first level trigger decision. The maximum rate of GFLT accept decisions is designed to be 1 kHz. Up to the GFLT both the trigger and readout are deadtime free.

On GFLT accept, data accepted for further analysis are copied to a second level trigger pipeline. A GFLT accept rate of 1 kHz and a "copy" time of 30 μs result in 3% deadtime. This is the only source of deadtime provided no buffer full states occur.

[1] Now at Blohm & Voss, Hamburg, Germany.

A second level trigger processor local to the component subsystem computes a trigger subdecision, which is forwarded to the global second level trigger (GSLT) and used to compute an overall second level trigger decision. The GSLT is designed to accept ca. 10% of all GFLT accepted triggers.

In case a component subsystem receives a positive GSLT decision, the corresponding data are assigned a "GSLT decision number" and transferred to the Eventbuilder. The Eventbuilder combines and formats all the component data carrying the same GSLT decision number into one data set. This data set is called an "event", and its GSLT decision number is also referred to as the "event number".

Once an event is complete, it is input to the third level trigger (TLT). The TLT is a processor farm consisting of six branches of a total of 36 processor nodes. It performs the global event reconstruction and a final filtering and is designed to accept up to 5 events/s.

3. Developing the ZEUS Eventbuilder

The performance of the Eventbuilder has strong influence on the output of the entire experiment, and a careful design of both the systems hardware and software is mandatory for optimum operation. The use of

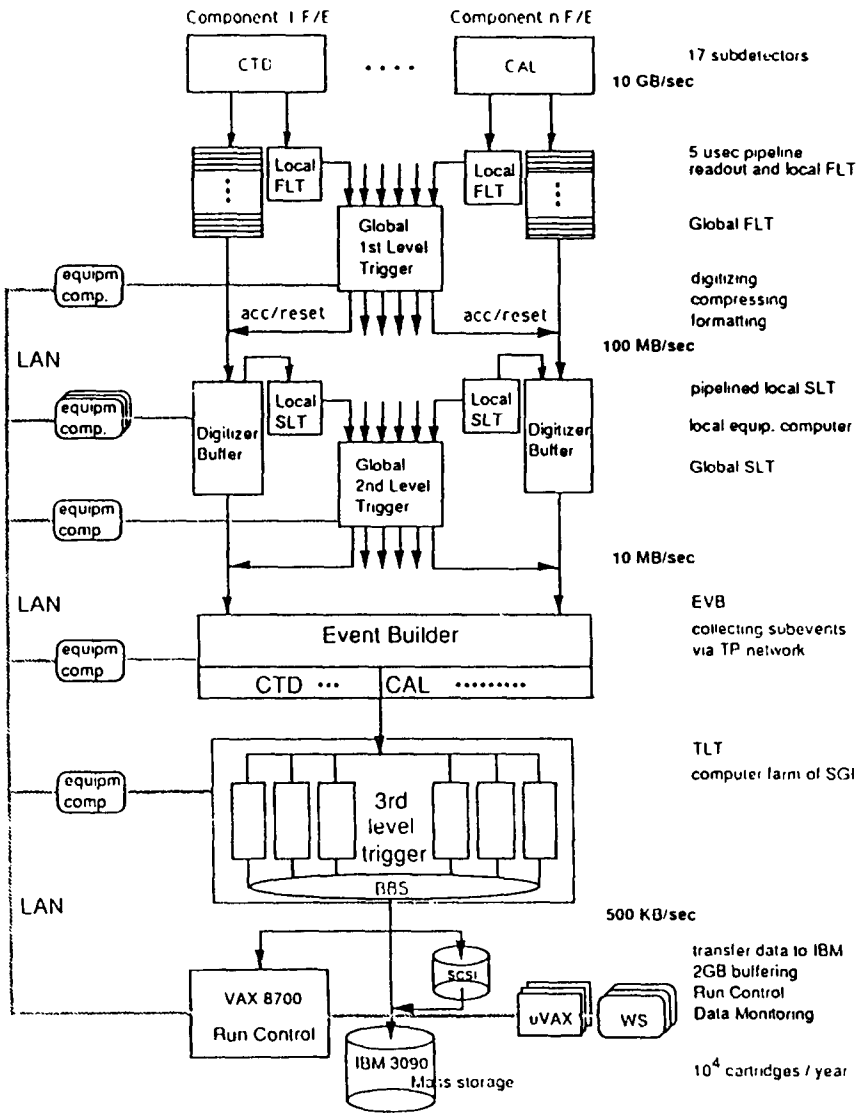

Fig. 1. Layout of the trigger and data acquisition system of the ZEUS experiment; on the right side, the data throughput at the different components of the system is shown.

Table 1
Specification of component subsystems in the trigger and data acquisition system [3]

Detector component	No. readout	Maximum Channels	Readout event length [kB]	Processor
Central tracking detector	CTD	4608	30	transputer
Forward/rear track. det.	FRTD	5778	15	transputer
Barrel calorimeter	BCAL	5184	20	transputer
Forward calorimeter	FCAL	4344	20	transputer
Rear calorimeter	RCAL	2336	10	transputer
Transition radiation det.	TRD	2472	10	transputer
Hadron electron separator	HES	37304	10	transputer
Backing calorimeter	BAC	40000	2	transputer
Vertex detector	VXD	832	2	68k-family
Beamline	BEAM		1	transputer
Barrel muon chambers	BMUO	62256	0.6	transputer
Forward muon spectrometer	FMUO	18948	0.5	68k-family
Leading proton spectrometer	LPS	52000	0.2	68k-family
Luminosity monitor	LUMI		0.2	68k-family
Vetowall	VETO		0.01	transputer
Global second level trigger	GSLT		20	transputer
Fast clear	FCLR		2	68k-family
	Σ:	258142	142	

structured development techniques (SA/SD) yielded a well-partitioned and flexible system structure and ensured all requirements being met.

The following sections list the requirements on the Eventbuilder and illustrate the analysis and design process. Additionally, the implementation of the Eventbuilder and its operation are also described. Finally, experience gained from system development and integration is summarized [#1].

3.1. Requirements

The requirements on the ZEUS Eventbuilder are defined by its position in the trigger and data acquisition system, the rate of positive GSLT decisions and the amount of data acquired from the component subsystems. The main issues are to:

- combine and format data from different components carrying the same event number into a single data record (event). Sufficient buffer space has to be provided to account for the asynchronously arriving component data. Complete events have to be transferred to the TLT, which involves a data transport over a distance of around 100 m;
- sustain a GSLT accept rate of at least 100 events/s. Taking into account the event sizes defined in table 1, this requires a total bandwidth of more than 15 MB/s and up to 3 MB/s at the interfaces to component subsystems;
- provide fault tolerance against failure of transmission lines to the TLT. The data transport to the TLT necessitates the use of serial transmission lines and a redundant hardware architecture;
- distribute the events over the TLT branches. By surveying the data throughput at the interfaces to the TLT, the load of its different branches of processor nodes can be estimated and used for load-balancing.

Further requirements include format checks of component data and generation of an index to the data objects inside an event record [#2], careful on-line monitoring for debugging and system analysis purposes, conceptual simplicity regarding maintenance and future upgrades, and low cost.

3.2. Essential model

The Eventbuilder has to support interfaces to the various detector components, the six branches of third level trigger processor nodes (TLT), the global second level trigger (GSLT) and the run control console (RC). The context diagram (CD, fig. 2) shows, how the Eventbuilder is embedded in the trigger and data acquisition system.

[#1] This article introduces part of the notation of SA/SD. However, for the modelling technique we refer to refs. [7,14,12].

[#2] The ZEUS collaboration stores their data in the ADAMO format [4].

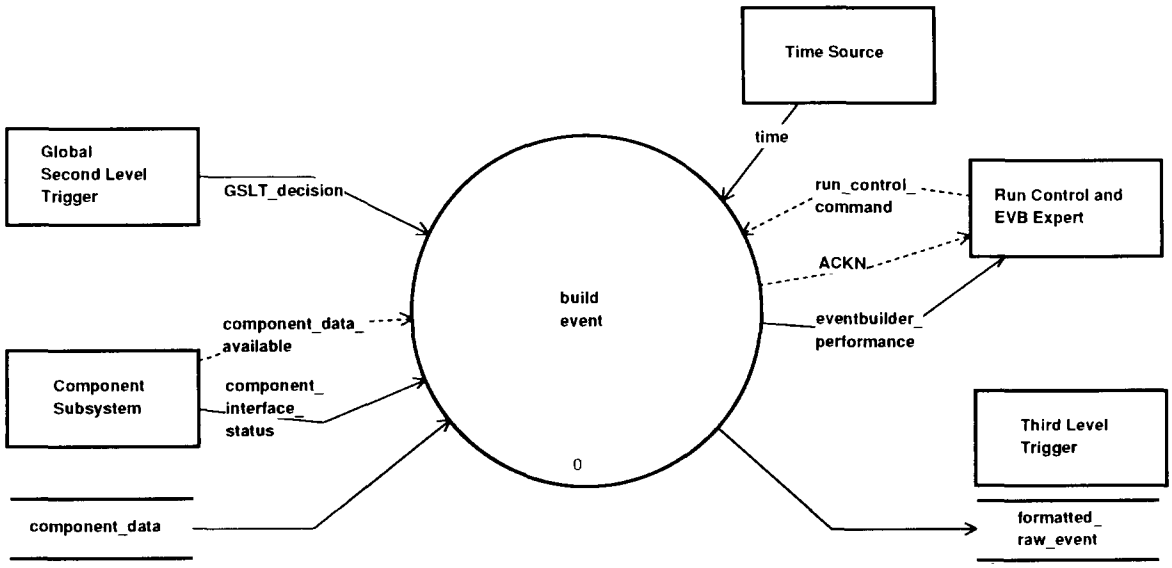

Fig. 2. The context diagram defines the interfaces between the Eventbuilder and other components of the trigger and data acquisition system. Boxes represent external systems, bars common memory areas and arrows the flow of data (solid) or control information (dashed). The Eventbuilder is shown as a bubble, representing a process.

Entity relationship diagrams (ERDs) show the data elements occurring in a system and highlight the relationships between them. In case of the Eventbuilder, an ERD can easily be derived from a description of the system's behaviour, where nouns refer to objects and verbs indicate relationships (fig. 3). Every detector component has to *respond* to a *GSLT_decision* by providing its *component_data*. *Component_data consists_of* several *component_data_banks*. Scanning the *component_data* reveals the *component_data_composition*. *Matching* this to the *readout_configuration* ensures only valid banks being built into the event. When all *participating_components* of a run have *responded* to the *GSLT_decision*, the set of valid *component_data_banks* and the *event_composition* can be *combined_into* the *formatted_raw_event*.

Tasks operating on the data elements and establishing the relationships are defined in a control and data flow diagram (CDFD, fig. 4). The diagram is an extension of the "build event" process in the context dia-

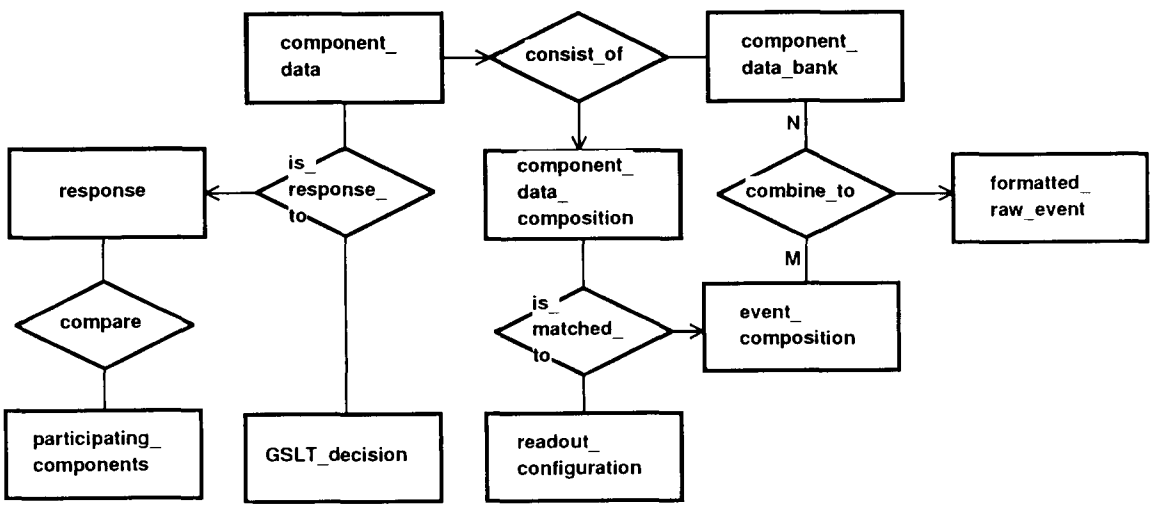

Fig. 3. The data objects occurring within the boundaries of the Eventbuilder are defined in an entity-relationship-diagram. Boxes indicate data objects, diamonds represent relationships between objects. The numbers classify the type of a relationship (one-to-one, one-to-many, ...).

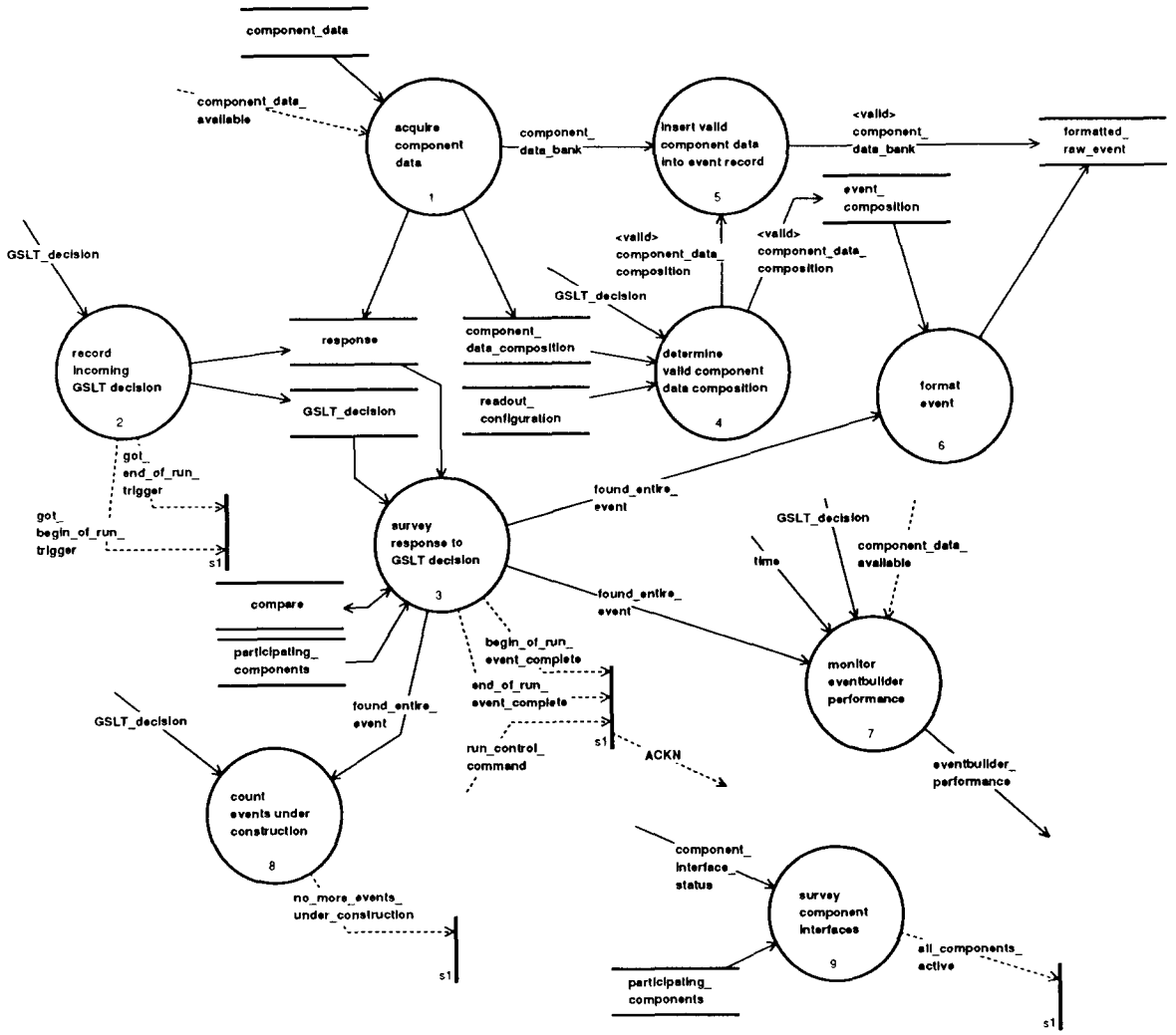

Fig. 4. The control and data flow diagram defines how the Eventbuilder processes objects and establishes relationships amongst them. The notation is similar to the context diagram. The vertical bar denotes the interface to the finite state machine which is synchronizing the processes.

gram. It has the same input and output flows, but gives a more detailed definition of how to build events. The processes of the CDFD are synchronized by a control unit (finite state machine, fig. 5) which analyzes the process states and reacts on external signals.

3.3. Hardware architecture

Transputers [#3] proved to be well suited processors for the ZEUS Eventbuilder. Standard VME transputer modules offering two T800 transputers with 4 MB of private memory each and a triple-ported memory (TPM) of 128 kB or 512 kB on a double-height VME-module [11] have been developed within the ZEUS collaboration. It was decided to use those modules wherever possible. Fig. 6 shows the layout of the Eventbuilder hardware.

Interfaces to component subsystems and to branches of TLT processor nodes connect the Eventbuilder with the trigger and data acquisition system of the ZEUS experiment. To keep the interfaces independent of the implementation of the external systems, common memory areas have been chosen for data input to or output from the Eventbuilder. The interfaces are implemented using the ZEUS standard transputer modules with the common memory areas being located in the TPMs. At the input side, one of the board's transputers is made available to the component subsystem. This

[#3] Transputers are single VLSI devices with processor, memory and communication links to other transputers [9]. Transputers are building blocks for real-time parallel systems as described in ref. [6]. Their links are designed for synchronization purposes inside distributed systems, but may also be used for data transport.

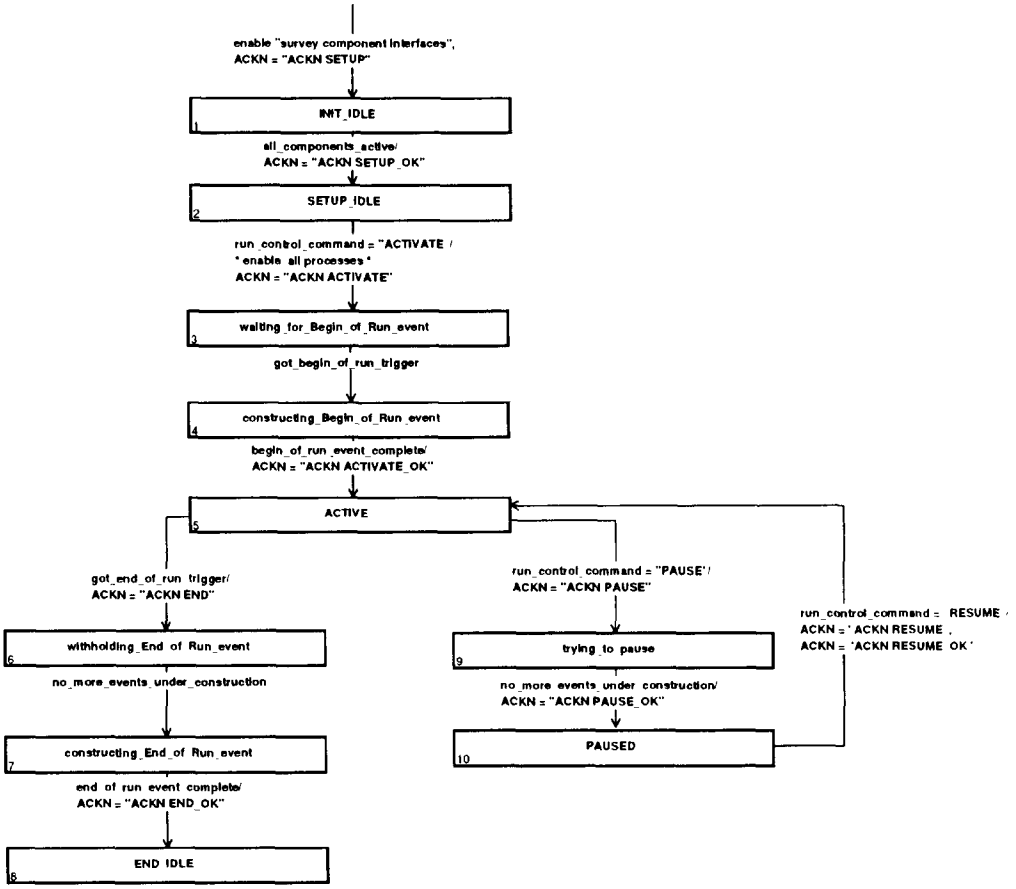

Fig. 5. The state transition diagram (STD) defines a finite state machine. Boxes denote system states, arrows show transitions between them. Labels on the arrow define the condition under which a transition occurs and list the actions to be taken.

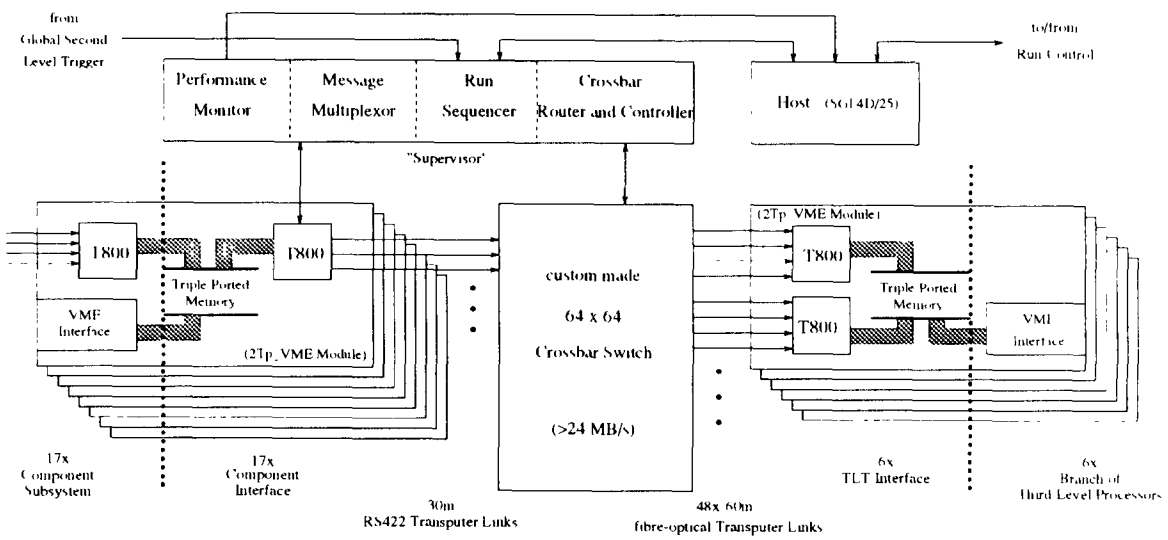

Fig. 6. Layout of the Eventbuilder hardware. Interfaces to component subsystems and branches of TLT processor nodes use ZEUS standard transputer modules, while supervisor and crossbar switch are custom made. For maximum performance fibre-optical transputer link connections have been developed for the data transfer between crossbar switch and interfaces to the third level filter farm.

way, components can access the memory via VME or transputer. The third level trigger obtains its data by VME accesses to the TPMs.

A network of data paths has to be foreseen in the Eventbuilder to transport data from every component subsystem to any branch of TLT nodes. A freely configurable network (crossbar switch) has proven to be the best solution [5]. Crossbar switches can connect any of their inputs to any of their outputs. In case of an $N \times N$ crossbar switch, N such connections can be established simultaneously. The Eventbuilder 's custom made crossbar switch for transputer links is based on Inmos C004 chips [10].

For maximum performance, fibre-optical link connections [8] have been developed for the long distance data transfer to the third level filter farm. The data transfer is limited by the handshake protocol on transputer links. Currently, a peak data throughput of 600 kB/s/link is achieved, limiting the total sustained bandwidth to 24 MB/s.

A control unit (Supervisor) provides the interface to Run Control and configures the crossbar switch according to the data arriving at the component interfaces.

3.4. System implementation and operation

To implement the Eventbuilder, the processes of the Essential Model were allocated to the different processor groups. Then the code for each transputer and the protocols between them were designed. The code is written in parallel C. An SGI 4D/25S unix workstation with a purpose built transputer interface served as host and development platform.

The Eventbuilder operation principle can be summarized as follows:

- Component subsystems provide their data in a common memory area and signal its availability to the Eventbuilder. The Eventbuilder then checks the component data for the correct format.
- As soon as the GSLT decision is available for an event, the Eventbuilder tries to transfer all the component data to one of its TLT interfaces. For this purpose, the component interfaces issue a "link request" to the crossbar router whenever they have data ready for transfer. Once they receive a "link ready"-message, the data transfer to the TLT interface is immediately started on the specified link. The availability of this link is again signalled by a "link release"-message.
- The decision, which event should be transferred to which TLT branch, is computed by the crossbar control task. It traces the buffer and I/O loads on each TLT branch to avoid new events being directed to busy branches. The connections between component and TLT interfaces are installed by a router which is tuned to minimize deadtime on the transmission lines.
- When all component data of an event have been transferred to a TLT interface board, the formatting of the event is triggered by the control unit. Formatted events are copied to the common memory area with the TLT.

3.5. Experience

The Eventbuilder of the ZEUS experiment has been developed, implemented and tested between 1988 and 1991, consuming about 11 man years. Most of the effort has been spent on software development (7 man years). Because of the extent of the Eventbuilder system (more than 50 transputers distributed over 24 VME crates) and its numerous interfaces, about half of this time went into system integration and verification.

The use of SA/SD techniques proved to be helpful in many situations. The software model is well partitioned and of a flexible structure, so that modifications of requirements usually affect only a single process. The encapsulation of processes enabled prototyping and partial implementation and supported system integration at an early stage. As all process interfaces were well defined, simulators of the different processor groups and external systems have been developed. Thus, very reliable performance estimates have been available at any stage of system development.

Transputers have shown to be easy-to-handle multi-purpose processors for real-time parallel systems. However, testing and debugging software distributed over several transputers turned out to be a difficult and very time consuming task as no tools were available which could analyze a transputer network without changing its real-time behaviour. For the tracing of synchronization problems, again the diagrams of the Essential Model were indispensable.

4. Eventbuilder performance

The Eventbuilder of the ZEUS experiment has seen successful operation for more than one year. During this time, the Eventbuilder performance has been carefully monitored and evaluated. This fact and its central position in the data acquisition chain have enabled the Eventbuilder to become an important diagnostic and analytic tool for the entire trigger and data acquisition system.

4.1. Monitoring concept

Eventbuilder operation involves several hundred processes which are distributed over more than 50 transputers and have to share limited system resources

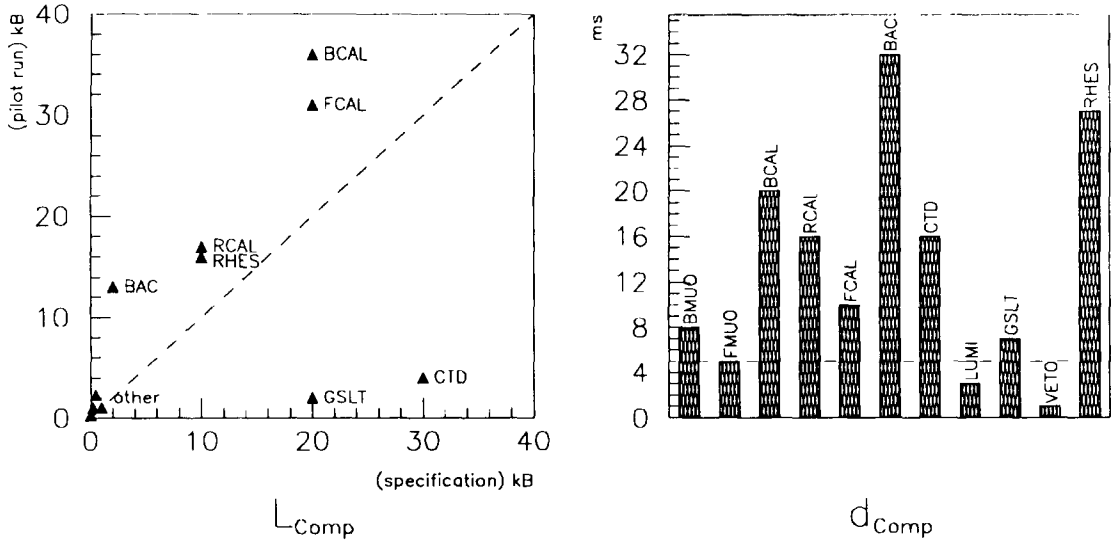

Fig. 7. Behaviour of component subsystems as observed during pilot run. Left: average amount of data acquired per event, compared to specification values. Right: average response time on trigger decision (specification: 5 ms).

like buffer space or transfer lines. A set of characteristic quantities is monitored during Event builder operation to trace the system performance.

Monitoring data are collected in different parts of the Eventbuilder, but have to be analyzed on a dedicated processor. By passing the data along with the

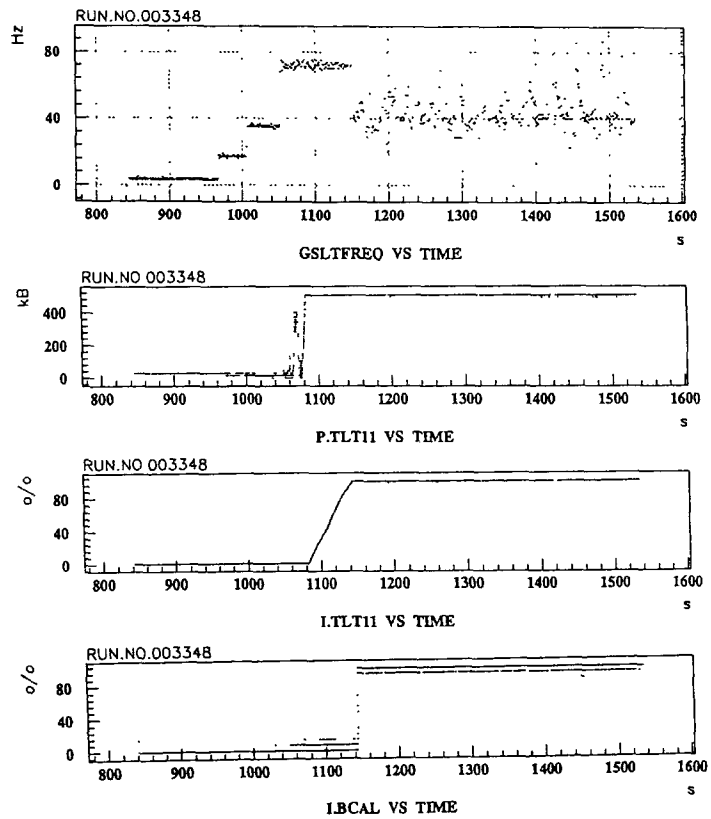

Fig. 8. Buffer loads reveal unusual system behaviour (explanation in the text).

synchronization messages, no extra traffic is introduced on the Eventbuilder network. To keep the extra load which monitoring imposes on the Eventbuilder processors as low as possible, monitoring data are collected while the events are transferred instead of being taken at fixed intervals. Time stamps allow tracing of the Eventbuilder performance. To allow for correlations of monitoring data acquired in different parts of the system (i.e. on different transputers), a "real time" is defined throughout the whole system [13].

4.2. Performance

Requirements on the bandwidth of the Eventbuilder arise from the GSLT frequency, f_{GSLT}, and the amount of data acquired from each component subsystem, L_{Comp}. Their nominal values are listed in section 3.1. The response time of a component subsystem to a GSLT decision, d_{Comp}, defines the minimum buffer capacities required at the component interfaces.

During the first year of operation, the mean GSLT decision rate, f_{GSLT}, was kept below 20 events/s. Therefore, the limit of the Eventbuilder has not been reached. Measurements have shown the total bandwidth to be at least 24 MB/s. The Eventbuilder can construct up to 72 events in parallel. Its buffers can accommodate at least 75 more events, depending on the event size. Fig. 7 shows data sizes and response times for components as observed during the pilot run and compares them with the specification.

4.3. On-line monitoring and system analysis

For on-line monitoring purposes it is sufficient to simply survey the load of the buffers inside the Eventbuilder. Any unusual system behaviour can be detected, sometimes even predicted from heavy buffer load. As an example, fig. 8 shows how the Eventbuilder's buffers fill when the accept rate of the second level trigger (GSLT) exceeds the speed of the third level trigger (TLT). As the TLT is located downstream from the Eventbuilder, buffers are expected to start filling at the backend. Indeed, the common memory areas with the TLT fill up first (P.TLTn), followed by the internal buffers of the interfaces to the TLT (I.TLTn). Finally, the buffers inside the private mem-

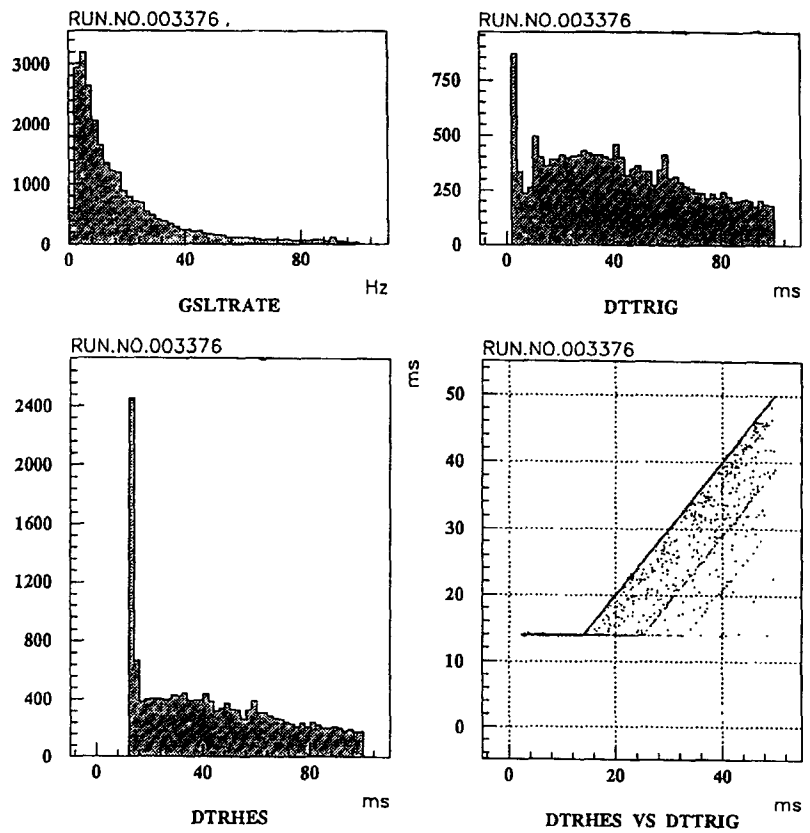

Fig. 9. The maximum event rate which can be handled by component subsystems can be determined from monitoring the GSLT accept rate and the data flow into the Eventbuilder (see text).

ory of the interfaces to the component subsystems (I.Comp) fill. The figure shows, that when all buffers in the Eventbuilder were filled, the data acquisition system stabilized at a GSLT accept rate of 44 events/s.

Monitoring the GSLT accept rate and the data flow into the Eventbuilder allows the determination of the maximum event rates which can be handled by the different component subsystems. Even at low GSLT accept rates, consecutive positive GSLT decisions may be separated only by a few milliseconds. Fig. 9 shows for a run with an average GSLT rate of 18 Hz the interval between two consecutive GSLT decisions, DTTRIG, going down to 2 ms (upper left and right). Component subsystems should have a constant response time on GSLT decisions, therefore the interval between two consecutive component data sets, DTComp, is expected to equal the corresponding DTTRIG. However, plotting DTComp against DTTRIG shows DTComp to saturate (lower right). Obviously, the component subsystem cannot keep pace if the trigger decisions are coming in too fast, and the corresponding data start piling up in the system's buffer. Only when DTTRIG increases beyond the minimum of DTComp the component subsystem can start emptying its buffers, and DTComp < DTTRIG. Determining the minimum of DTComp allows the derivation of the maximum event rate which can be handled by a component subsystem.

The last issue shows that monitoring Eventbuilder operation may also be used to survey the performance of those components interfacing the Event builder. This way, the Eventbuilder has become an important diagnostic and analytic tool for the entire data acquisition system. Currently, the Eventbuilder environment is used for the construction of a prototype expert system [2,1] which can survey and analyze the monitoring data. Its goal is to provide on-line diagnostics and guidance for the shift crew running the experiment.

5. Conclusion

The Eventbuilder of the ZEUS experiment is a transputer-based real-time parallel data formatting and transport system with a total bandwidth of at least 24 MB/s. It has seen successful operation for more than one year now.

The Eventbuilder has been developed making extensive use of structured system development techniques. Application of structured analysis and structured design (SA/SD) yielded a well-partitioned and flexible system structure and ensured that all requirements were met.

Its central position has enabled the Eventbuilder to become an important diagnostic and analytic tool for the entire trigger and data acquisition system of the ZEUS experiment. The full potential of the Eventbuilder diagnostics will be achieved when the expert system [2] becomes fully available.

References

[1] U. Behrens, M. Flasinski and L. Hagge, ZEXP – The ZEUS Expert System, DESY Report 92–141 (1992).
[2] U. Behrens, M. Flasinski, L. Hagge and K. Ohrenberg, Prospects of an Expert System for ZEUS, ZEUS Note, in preparation.
[3] ZEUS Collaboration. The ZEUS Detector, Status Report 1989. Technical report, DESY (1989).
[4] S.M. Fisher and P. Palazzi, The ADAMO Data System, 3.1st ed. (1990).
[5] L. Hagge, Diplomarbeit, University of Hamburg (1990).
[6] C.A.R. Hoare, Commun. ACM. 21(8) (1978) 666.
[7] D.J. Hatley and I.A. Pirbhai, Strategies for Real-Time System Specification, (Dorset, New York, 1987).
[8] We would like to thank Holger Leich and associates of IfH Zeuthen for the development of the fibre-optical transputer links.
[9] Inmos Ltd., Bristol, UK, The Transputer Databook, 2nd ed. (1989).
[10] We would like to thank Frank O. Lohmann for the construction of the crossbar switch.
[11] NIKHEF, Electronic Department, Amsterdam, The Netherlands, Short Hardware Description of the 2TP_VME Module (1990).
[12] M. Page-Jones, The Practical Guide to Structured Systems Design (Yourdon, New York, NY, 1980).
[13] T. Schlichting, Diplomarbeit, University of Hamburg (1992).
[14] E. Yourdon. Modern Structured Analysis, Yourdon Press Computing Series (Prentice–Hall, Englewood Cliffs, NJ 1989).