ISTITUTO NAZIONALE DI FISICA NUCLEARE

Sezione di Trieste

INFN/AE-93/13

22 giugno 1993

G.L. Girotto, L. Lanceri, F. Scuri and E. Zoppolato

TRANSPUTER NETWORKS FOR THE ON-LINE ANALYSIS OF FINE-GRAINED ELECTROMAGNETIC CALORIMETER DATA

Servizio Documentazione
dei Laboratori Nazionali di Frascati

Transputer networks for the on-line analysis of fine-grained electromagnetic calorimeter data

G.L. Girotto[1,3], L.Lanceri[2,3], F. Scuri[1,3] and E. Zoppolato[1,3]

June 19, 1993

Abstract

Transputer networks, designed to perform parallel computations, are well suited for data acquisition, on-line analysis and second level trigger tasks in high energy physics experiments. Some simple algorithms for the analysis of fine-grained electromagnetic calorimeter data were implemented on two types of transputer networks and tested on real and simulated data from a silicon-tungsten calorimeter. Results are presented on the processing speed, measured in a test set-up, and extrapolations to a full size detector and data acquisition system are discussed.

[1]Istituto di Fisica, Universita' di Udine, Via Fagagna 208, I-33100 Udine, Italy
[2]Dipartimento di Fisica, Universita' di Trieste, Via A. Valerio 2, I-34127 Trieste, Italy
[3]INFN, Sezione di Trieste, Via A. Valerio 2, I-34127 Trieste, Italy

1 Introduction

Transputers are microcomputers with built-in communication capabilities and specially designed software tools that make them particularly suited to perform parallel data processing.

These features make them attractive for on-line data processing and high level digital triggers in nuclear and high energy particle physics experiments [1]. One existing application [2] is specialized to the treatment of data from tracking devices. In another case [3] transputers are used as building blocks of a complex read-out and second level trigger system.

In this paper we investigate the performance of transputer networks in the execution of algorithms for the on-line analysis of data produced by fine-grained electromagnetic calorimeters. The aims are:

- reducing the large amount of calorimeter data by calibration and zero suppression algorithms;
- computing track or shower parameters, such as position, angle, width, shower maximum, total energy;
- performing a preliminary classification of showers, to obtain a relevant background reduction before data recording.

This investigation was motivated by the needs of the Wizard experiment [4] whose detector is planned to operate in a satellite with stringent constraints on power consumption and data recording rates. The data taken on a test beam with the Wizard calorimeter prototype provided a realistic testing ground for the proposed ideas.

The paper is organized as follows: after a brief summary of transputer characteristics in section 2, we describe our approach to the reduction of calorimeter data and the computing algorithms in section 3; two possible ways of programming these algorithms in processes to be executed in transputer networks with the largest possible degree of parallelism are discussed in section 4. In section 5 the transputer hardware set-up used for our tests is described, together with the implementation of the processes in two different network configurations. Section 6 is devoted to the results of measurements of computation speed, with some final comments on how they can be extrapolated to a real experimental context.

2 The T800 Transputer

Transputers, manufactured by INMOS, are microcomputers with their own internal local memory and with links for connection to other transputers.

In our tests we used the INMOS transputer T800 whose block-diagram is shown in Figure 1. In the same VSLI chip the following components are packaged:

- a 32 bit integer processor (CPU), running at a clock frequency of up to 30 MHz;
- a 4 Kbytes fast memory (static RAM);
- a 64 bit floating-point coprocessor (FPU);
- an interface to external memory (4 Gbytes addressing space);
- four bidirectional intercommunication serial links, operating at up to 20 Mbit/s transfer rate, for the connection to other transputers.

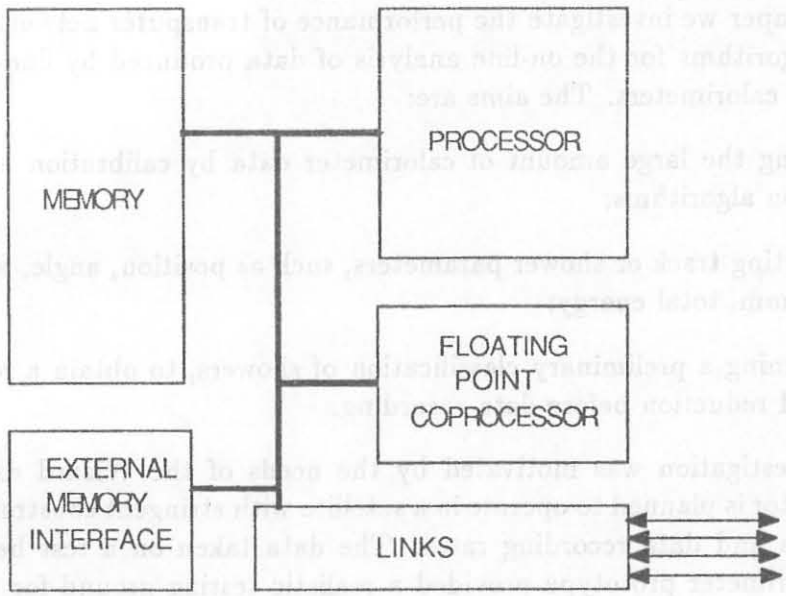

Figure 1: INMOS T800 Transputer block diagram.

The peak instruction execution rates that can be achieved are 30 Mips for the CPU and 3.3Mflops for the FPU. Transputers can be programmed in standard high level languages. However, the Occam [5] language allows the most efficient use of the architecture of transputers and transputer networks, supporting multiple parallel processes.

Processes are program building blocks that can be executed concurrently on one or more transputers. Processes communicate through channels, which can be established softwarewise between processes being executed on the same transputer, or may correspond to hardware links for processes running on different transputers.

Transputers provide hardware support for the Occam model of concurrency and communication between processes, through a microcoded scheduler that enables concurrent processes to be executed together, sharing the processor time.

Transputer networks can be easily built by interconnecting *TRAnsputer Modules* or TRAMs [6]. A TRAM is a small subassembly including one or more transputers,

a few discrete components, some external RAM and/or application specific circuits, mounted on a small board with standard size and pinout for a few control signals and the transputer links. Networks are obtained by plugging TRAMs on motherboards providing the required link interconnections.

3 On-line analysis of electromagnetic calorimeters data

In sampling calorimeters, layers of passive absorber material are alternated with active detector layers, that sample the ionization produced by charged particles, and provide information both on the amount of deposited energy and on its position.

A typical example of such a detector is the prototype of the fine-grained calorimeter of the Wizard experiment (Figure 2). It consists of 20 layers of silicon detectors, alternated with 19 plates of Tungsten, 1.75 mm (0.5 X_0) thick. The sensitive silicon elements in each layer are made of two Silicon Detectors, segmented in perpendicular strips, with 3.6 mm pitch, providing both x and y coordinate measurements. The total detector thickness is equivalent to 9.5 X_0. The final calorimeter for the Wizard experiment will cover a larger area ($60 \times 60\ cm^2$) with the same technique.

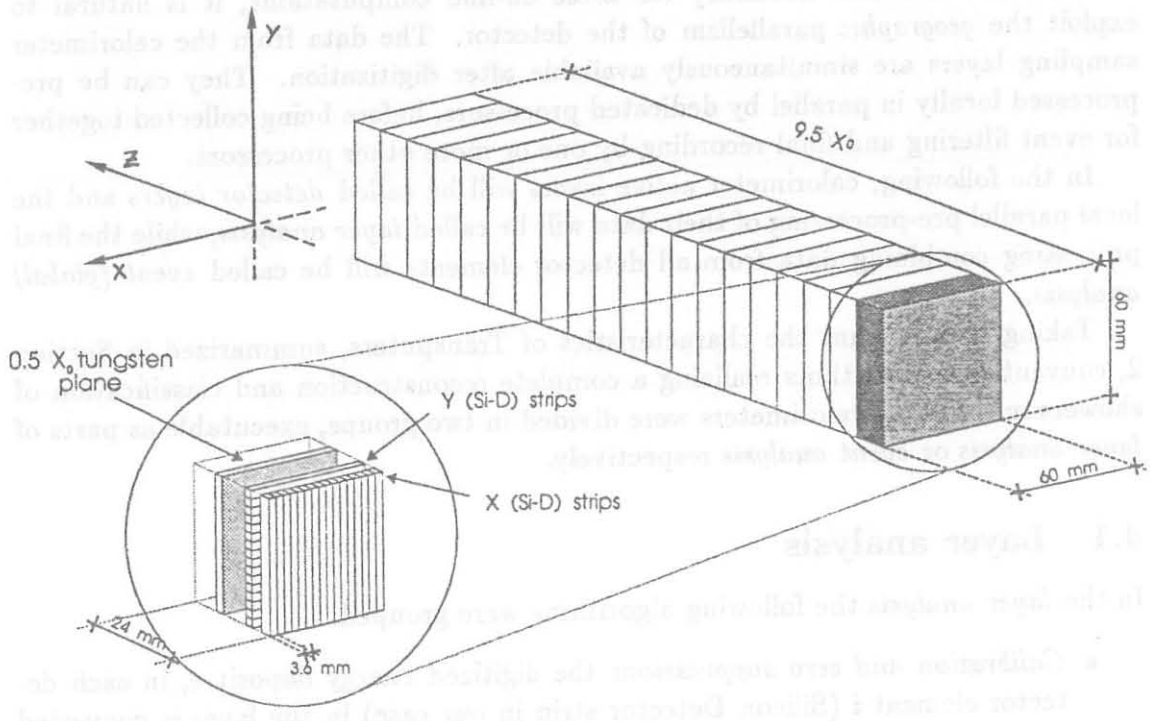

Figure 2: The prototype of the Silicon-Tungsten electromagnetic calorimeter of the Wizard experiment.

The data acquisition system of such detectors must be able not only to record the raw data, but also to process them on-line, in order to:

- apply calibration corrections to the data and suppress empty detector channels;
- reconstruct the position and direction of the incoming particle; this is relevant specially for high-level triggering on photons, for which no information is available from other detectors in the experimental set up;
- exploit the detector granularity for shower imaging and particle identification.

The last two features become essential whenever a reliable on-line filtering is necessary to reject a large fraction of backgrounds before data recording. In the case of Wizard, this is required by the operating conditions of the experiment, that will be borne on a satellite, with limited data transmission rate to the earth. In general, in all experiments this kind of pre-processing makes the following off-line analysis easier and cheaper.

4 Implementation of algorithms in parallel processes

To minimize the time necessary for these on-line computations, it is natural to exploit the *geographic* parallelism of the detector. The data from the calorimeter sampling layers are simultaneously available after digitization. They can be preprocessed locally in parallel by dedicated processors, before being collected together for event filtering and final recording by one or more other processors.

In the following, calorimeter active layers will be called *detector layers* and the local parallel pre-processing of their data will be called *layer analysis*, while the final processing combining data from all detector elements will be called *event (global) analysis*.

Taking into account the characteristics of Transputers, summarized in Section 2, conventional algorithms realizing a complete reconstruction and classification of showers in sampling calorimeters were divided in two groups, executable as parts of *layer analysis* or *event analysis* respectively.

4.1 Layer analysis

In the *layer analysis* the following algorithms were grouped:

- *Calibration and zero suppression*: the digitized energy deposit e_i in each detector element i (Silicon Detector strip in our case) in the layer is converted to mip (minimum ionizing particle) units using a set of pre-calculated calibration constants; zeros are suppressed by comparison with suitable thresholds (typically: 0.5 mip);

- *cluster variables*: in each layer, non-zero energy deposits in adjacent elements (strips) are grouped in *clusters*; for each cluster k, the following variables are computed: total energy deposit E_k in the cluster; energy-weighted centre $\overline{x}_k$, $\overline{y}_k$ of the cluster; *dispersion* of energy in the cluster k, defined in the x projection as:

$$\sigma_{x_k} = [\sum_{i=f_k}^{l_k} [e_i(x_i - \overline{x}_k)^2] / \sum_{i=f_k}^{l_k} e_i]^{1/2}. \quad (1)$$

  where f_k and l_k are the first and last strip in cluster k, and similarly σ_{y_k} in the y projection.

- *layer variables*: in each layer j, the following variables are computed: total energy deposit E_j in the layer; energy weighted centre $\overline{X}_j$, $\overline{Y}_j$ of the total deposit in the layer; *dispersion* of energy deposit in the layer, defined in the x projection as:

$$\delta_{x_j} = [\sum_{k=f_j}^{l_j} [E_k(\overline{x}_k - \overline{X}_j)^2] / \sum_{k=f_j}^{l_j} E_k]^{1/2}. \quad (2)$$

  where f_j and l_j are the first and last cluster in layer j, and similarly δ_{y_j} in the y projection.

The data from the Wizard prototype as well as those expected in the experiment typically have only one incident particle per event, originating one shower or track in the calorimeter. The *layer variables* in this case describe the lateral profile of an individual shower or track. These algorithms can be generalized to the case of multiparticle events. One possible strategy is to find the parts of the detector showing relevant energy depositions and to define locally calorimeter sub-volumes and detector sub-layers, matching in lateral size the typical shower width, in which the above described algorithms can be applied. This generalization was not of immediate interest in this study, and was left to future developements.

4.2 Event analysis

In the *event analysis* the following algorithms were grouped:

- Computation of *global variables*: Total energy E_{tot} deposited in the calorimeter, *global dispersion* of the energy deposit, defined as the sum Σ_{tot} of all σ_k for both projections, or alternatively as the sum Δ_{tot} of all δ_i, and the sum S of the differences between the deposited energies in each layer and those expected on average from a minimum ionizing particle. These three variables are related to the transverse and longitudinal shower developement respectively. Normalized to E_{tot}, they can be used for event classification.

- Reconstruction of *shower parameters*: by the least squares method, straight lines are fitted to the energy-weighted centres x_j, y_j of the total energy deposited in the detector layers, both in the xz and the yz projections, z being the longitudinal calorimeter axis (Figure 2). The parameters of the fitted lines give an estimate of the incoming particle direction and impact point on the first detector layer.

- *Event classification and filtering*: the *global variables* and the results of the fit can be used to classify the event according to the nature of the shower, and to filter out events in which an electromagnetic shower cannot be recognized or the fitting procedure fails. Both a combination of simple cuts on the global variables and a more refined algorithm based on feed-forward neural networks were tested [8].

4.3 "Pipeline" and "tree" networks

The two simplest networks considered in the literature [1] for parallel processing by transputers are called *pipeline* and *tree* respectively (Figure 3). The algorithms described in paragraphs 4.1 and 4.2 were implemented and tested in both types of networks. In both cases, the intrinsic geographic parallelism of the detector layers was exploited by devoting one process to each *layer analysis*. On the other hand, the processes dedicated to data collection and *event analysis* were different in the two approaches.

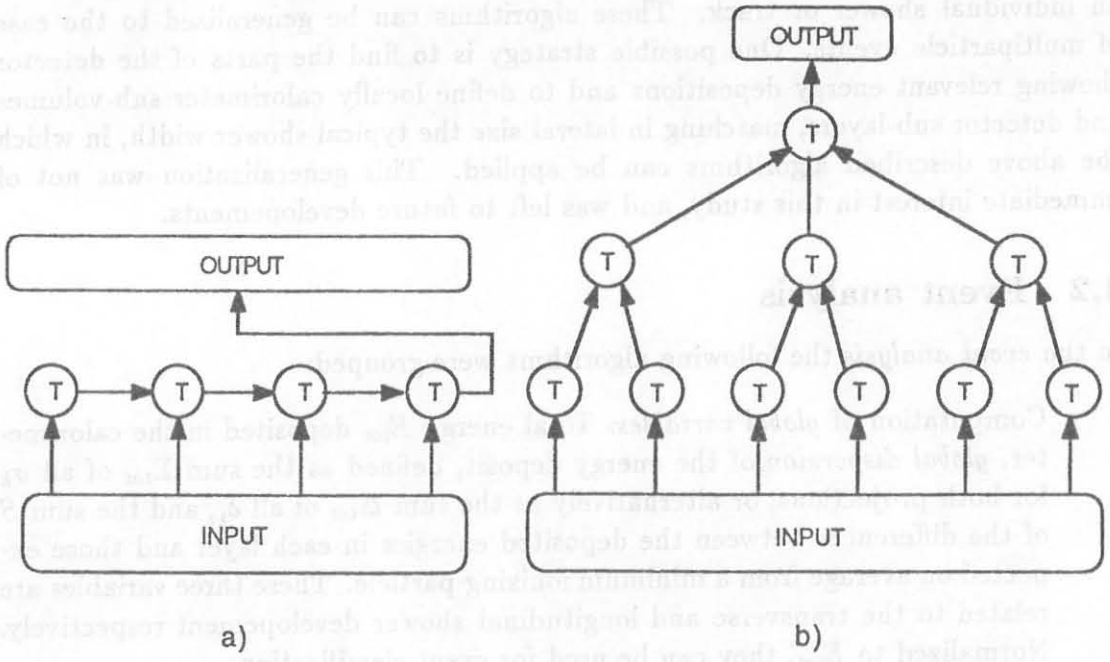

Figure 3: The two simplest transputer network architectures: (a) "pipeline" and (b) "tree".

During program developement and part of the performance tests, all processes, connected by software *channels*, were executed on the same transputer. The detector data, both real and simulated, were kept on a host computer and distributed to the processes devoted to layer analysis by the interface process *inout* (Figure 4 and Figure 5), simulating the detector. The same process was also used to collect the results of the event analysis and transmit them to the host, and at initialization to receive and distribute calibration constants.

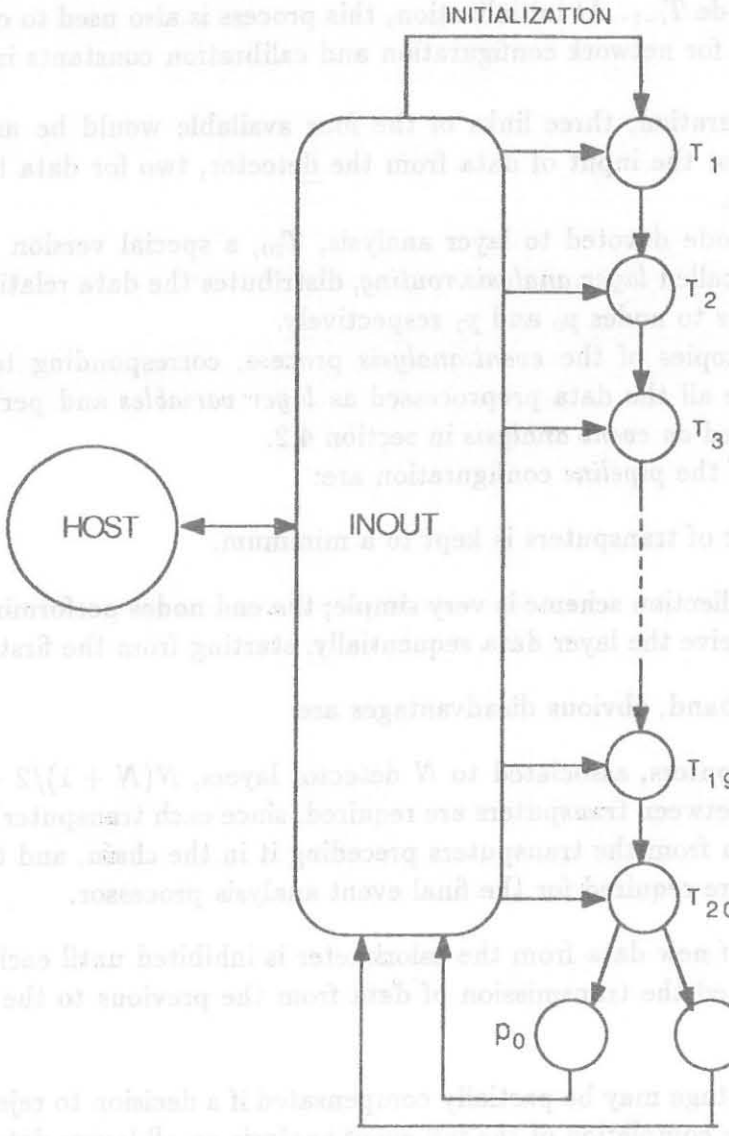

Figure 4: "Pipeline-like" configuration of Transputer Modules (TRAMs) for 20 detector layers.

4.3.1 "Pipeline-like" configuration

The *pipeline-like* network is shown in Figure 4. Nodes T_i $(i = 1,...20)$ and p_0, p_1 represent sets of processes to be finally executed on separate transputers.

Two processes correspond to each node T_i:

- *layer.analysis*: it receives the raw data of one detector layer and performs all the computations described as "layer analysis" in section 4.1.

- *pipe*: when *layer.analysis* is completed, it sends the results (if required, also the raw data) to the following node T_{i+1}, and receives the data transmitted by the previous node T_{i-1}. At initialization, this process is also used to communicate parameters for network configuration and calibration constants initialization.

In this configuration, three links of the four available would be used for each transputer: one for the input of data from the detector, two for data transmission along the *pipeline*.

On the last node devoted to layer analysis, T_{20}, a special version of the layer analysis process, called *layer.analysis.routing*, distributes the data relative to the xz and yz projections to nodes p_0 and p_1 respectively.

Finally, two copies of the *event.analysis* process, corresponding to the nodes p_0 and p_1, receive all the data preprocessed as *layer variables* and perform all the computations listed as *event analysis* in section 4.2.

Advantages of the *pipeline* configuration are:

- The number of transputers is kept to a minimum.

- The data collection scheme is very simple; the end nodes performing the *event analysis* receive the layer data sequentially, starting from the first layer.

On the other hand, obvious disadvantages are:

- for N transputers, associated to N detector layers, $N(N+1)/2+2$ message exchanges between transputers are required, since each transputer must transmit the data from the transputers preceding it in the chain, and two message exchanges are required for the final event analysis processor.

- The input of new data from the calorimeter is inhibited until each transputer has completed the transmission of data from the previous to the next in the chain.

This last disadvantage may be partially compensated if a decision to reject the event is taken before the completion of the full event analysis on all layers data. In several cases, this is a real possibility because approximate but sufficient information to reject an event can already be obtained from the first few detector layers.

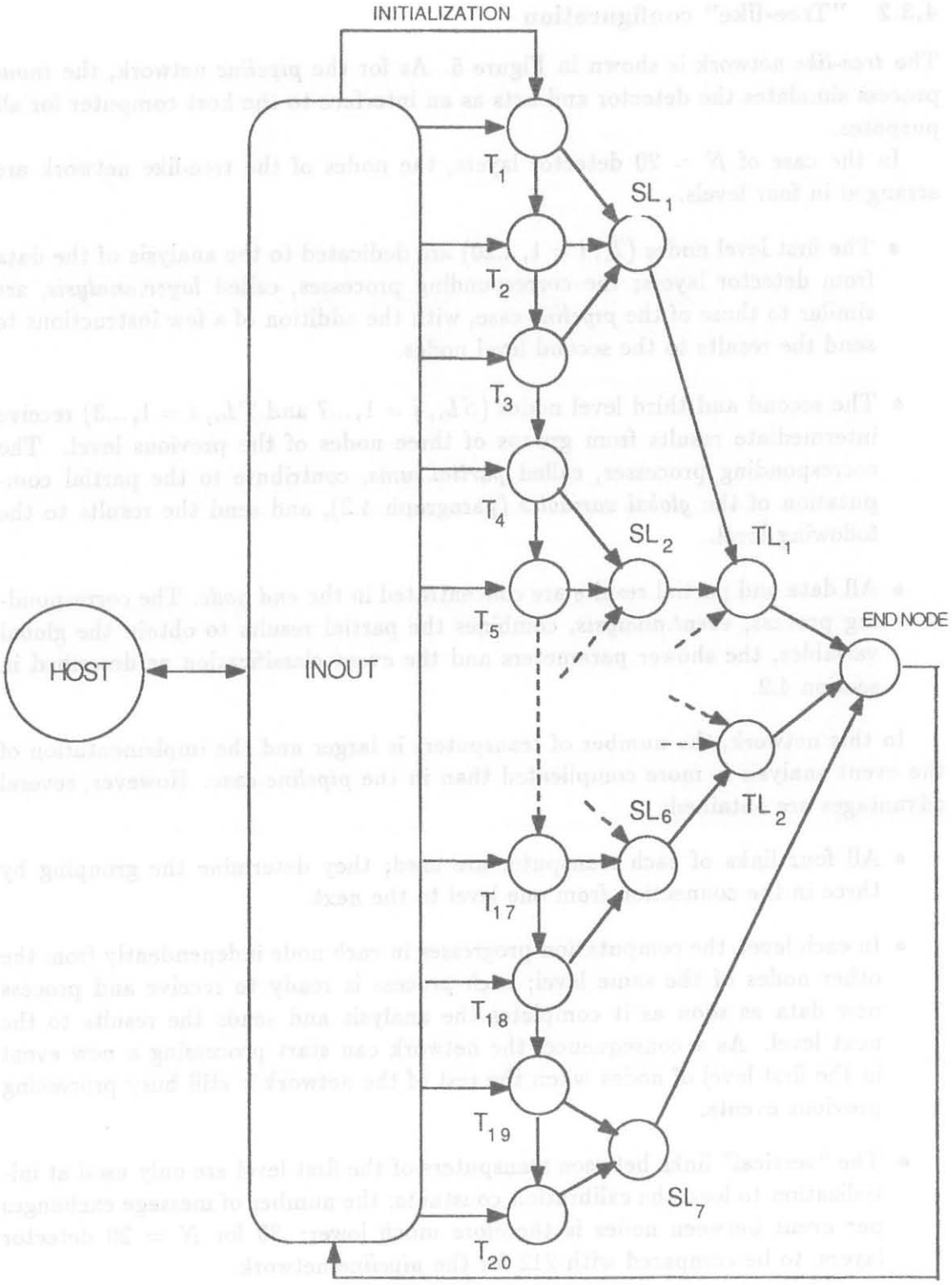

Figure 5: "Tree-like" configuration of Transputer Modules (TRAMs) for 20 detector layers.

4.3.2 "Tree-like" configuration

The *tree-like* network is shown in Figure 5. As for the *pipeline* network, the *inout* process simulates the detector and acts as an interface to the host computer for all purposes.

In the case of $N = 20$ detector layers, the nodes of the tree-like network are arranged in four levels.

- The first level nodes (T_i, $i = 1, ...20$) are dedicated to the analysis of the data from detector layers; the corresponding processes, called *layer.analysis*, are similar to those of the *pipeline* case, with the addition of a few instructions to send the results to the second level nodes.

- The second and third level nodes (SL_i, $i = 1, ...7$ and TL_i, $i = 1, ...3$) receive intermediate results from groups of three nodes of the previous level. The corresponding processes, called *partial.sums*, contribute to the partial computation of the *global variables* (paragraph 4.2), and send the results to the following level.

- All data and partial results are concentrated in the *end node*. The corresponding process, *event.analysis*, combines the partial results to obtain the global variables, the shower parameters and the event classification as described in section 4.2.

In this network, the number of transputers is larger and the implementation of the event analysis is more complicated than in the *pipeline* case. However, several advantages are obtained:

- All four links of each transputer are used; they determine the grouping by three in the connection from one level to the next.

- In each level, the computation progresses in each node independently from the other nodes of the same level; each process is ready to receive and process new data as soon as it completes the analysis and sends the results to the next level. As a consequence, the network can start processing a new event in the first level of nodes when the rest of the network is still busy processing previous events.

- The "vertical" links between transputers of the first level are only used at initialization to load the calibration constants; the number of message exchanges per event between nodes is therefore much lower: 30 for $N = 20$ detector layers, to be compared with 212 for the *pipeline* network.

5 The test set-up

The Occam programs realizing both the *pipeline* and *tree* approaches were developed and tested in the laboratory on an INMOS developement system. An IBM PS/2 PC was the host computer.

For program developement and some performance tests, a reduced set of two T800 transputers was used. They were mounted on two TRAMs (an IMS B404 TRAM, 20 MHz clock frequency, with 2 Mbyte external memory, and an IMS B411 TRAM, 25 MHz clock frequency, with 1 Mbyte external memory) plugged on a motherboard (INMOS IMS B017) [7] inserted in the host computer.

In this configuration, all the processes (22 in the *pipeline* program, 30 in the *tree* program) were running on the first transputer. The second TRAM was used only for measurements of the computing time by the methods described in Section 6.

For more conclusive performance measurements, the system was extended by the addition of eight more T800 transputers, mounted on eight TRAMs (INMOS B401, with 32 Kbytes external memory, 20 MHz clock frequency) plugged on a VME motherboard (INMOS B014). On this motherboard, the links could be configured by software, using C004 programmable switches driven by a dedicated T222 transputer [7].

The complete configuration is shown in Figure 6. A total of ten transputers were therefore available, connected in an easily reconfigurable network. They were used to implement both the *pipeline* and the *tree* network for six detector layers, as shown in Figure 7. The methods and results of the performance measurements are described in the next section.

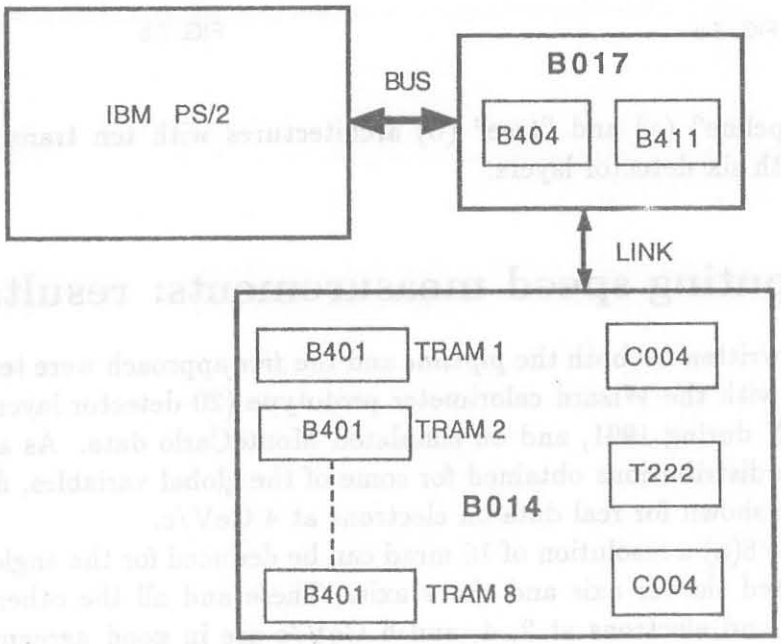

Figure 6: The test set-up: INMOS developement system.

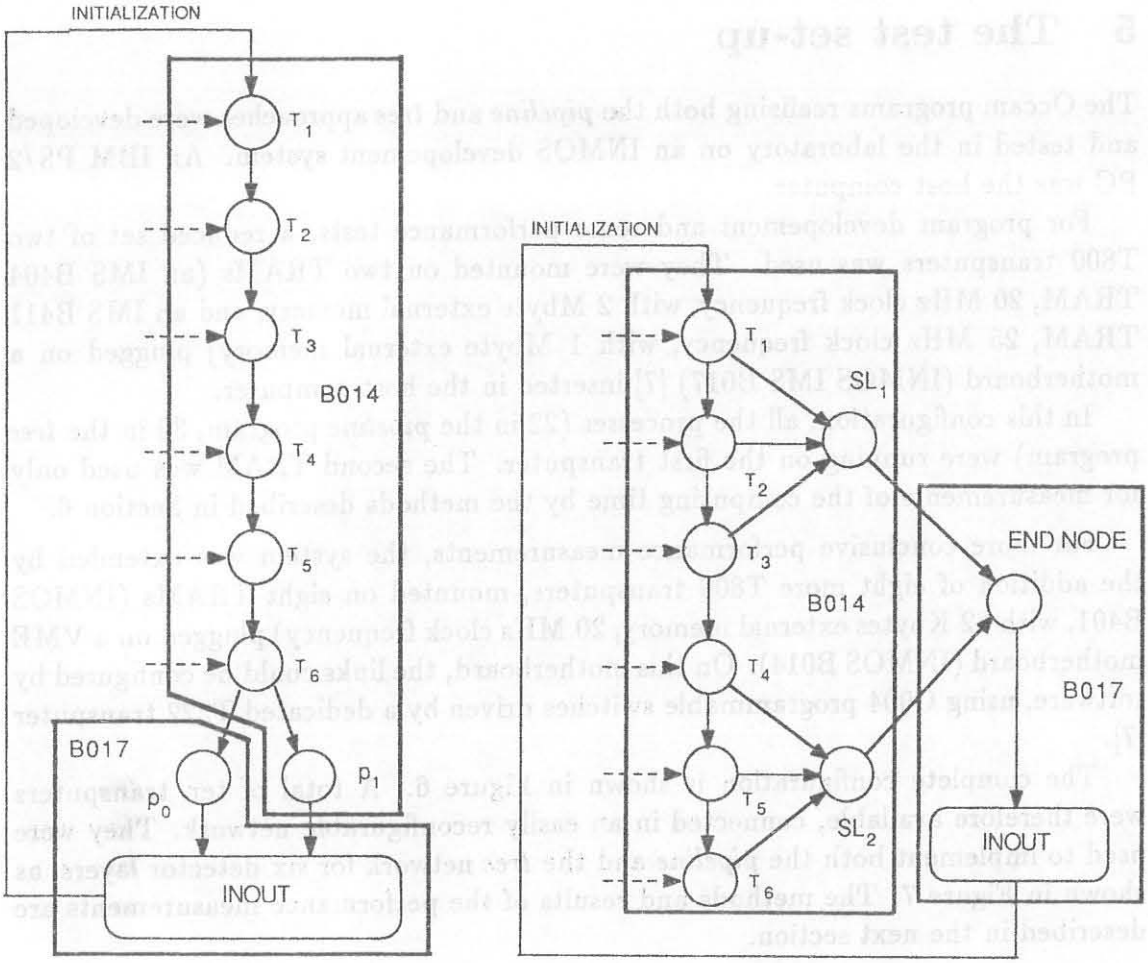

FIG. 7 a

FIG. 7 b

Figure 7: "pipeline" (a) and "tree" (b) architectures with ten transputers for a calorimeter with six detector layers.

6 Computing speed measurements: results

The programs written for both the *pipeline* and the *tree* approach were tested on real data obtained with the Wizard calorimeter prototype (20 detector layers) in a test beam at CERN during 1991, and on simulated MonteCarlo data. As an example, in Figure 8 the distributions obtained for some of the global variables, described in section 4.2, are shown for real data on electrons at 4 GeV/c.

From Figure 8(a) a resolution of 15 mrad can be deduced for the angle θ between the reconstructed shower axis and the z axis. These and all the other results on negative pions and electrons at 2, 4, and 6 GeV/c are in good agreement with a complete off-line analysis [4] performed with a standard technique.

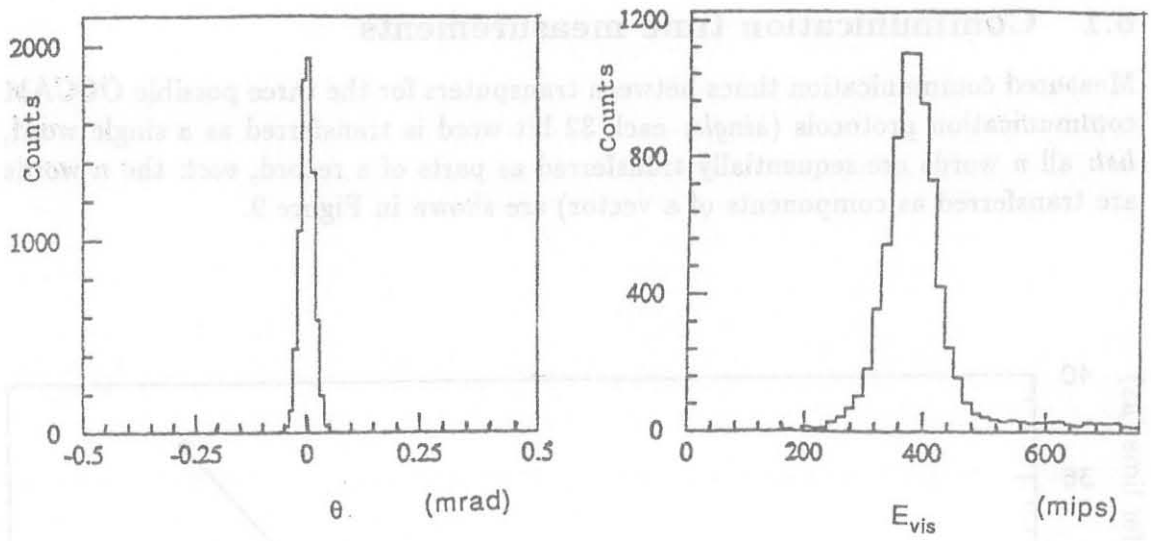

Figure 8: Distributions of two of the global variables obtained by the transputer program on real data from electrons at 4 GeV/c: (a) reconstructed shower angle in mrad in the xz projection; (b) total visible energy E in mips.

The analysis of the computing time performances for both configurations was developped in three phases:

- the execution times of all processes described in section 4.3, including communication processes, were first measured;
- the total computing time per event was then estimated by properly adding the measured values of individual processes execution times, for the cases of both 6 and 20 detector layers;
- finally, the total computing time per event was measured with the available system of ten transputers for the case of six layers and compared with the estimated value.

Computing time measurements of individual processes were performed with a configuration of just two transputers. One of them, called the *master* transputer, executes a program which generates simulated data, sends data to the other, the *slave* transputer, and receives from it the measured execution time. The *slave* receives the input data, reads the internal timer (resolution ranging between 1 and 64 μs) just before starting the execution of the process under test and just after its end, and sends the results to the *master*.

6.1 Communication time measurements

Measured communication times between transputers for the three possible OCCAM communication protocols (*single*: each 32 bit word is transferred as a single word, *list*: all n words are sequentially transferred as parts of a record, *vect*: the n words are transferred as components of a vector) are shown in Figure 9.

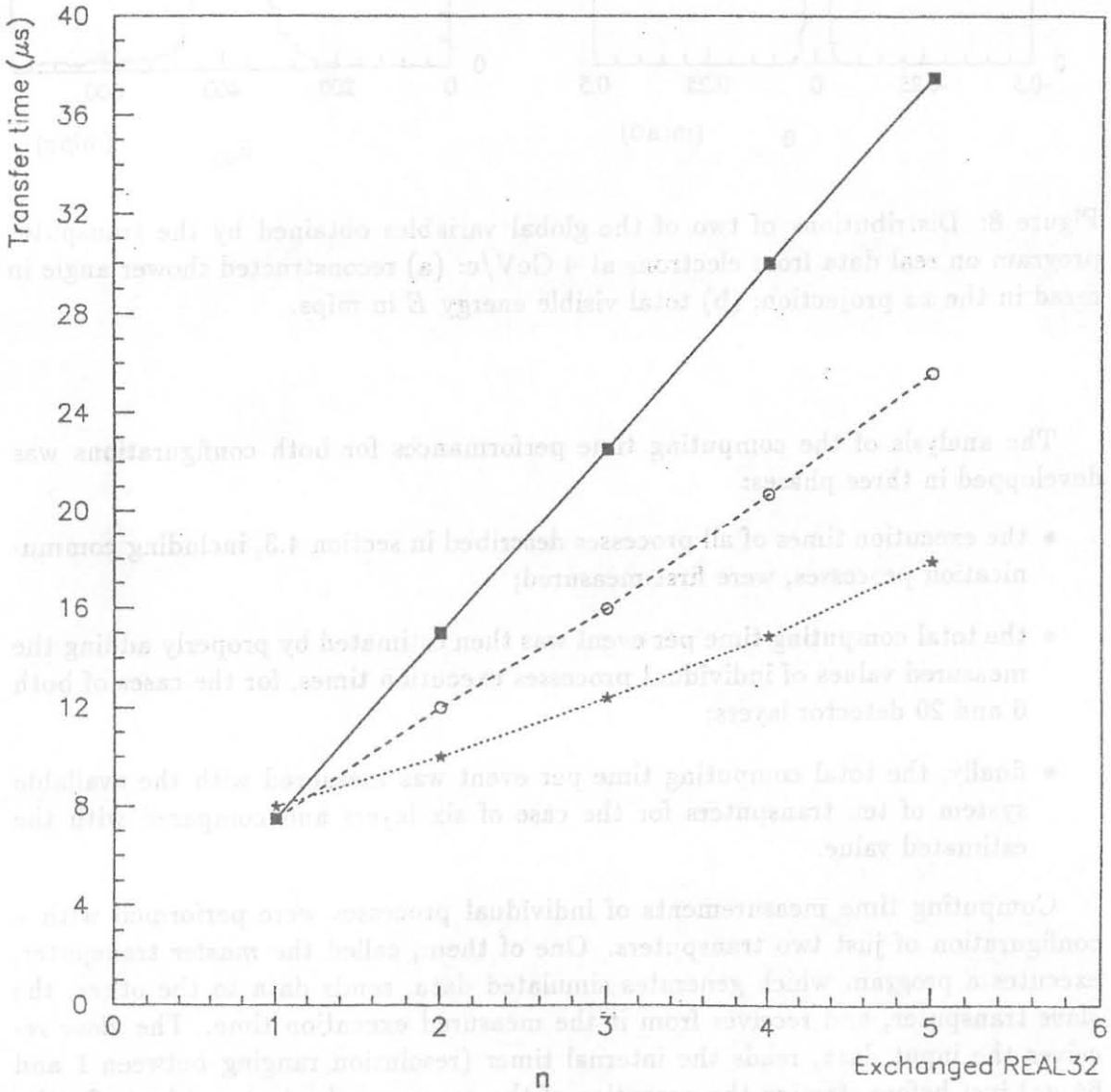

Figure 9: Measurements of the time needed for communication, as a function of the number n of transferred REAL32 words for different protocols described in the text: "single" (full line), "list" (dashed line), and "vect"(dotted line).

In Table 1 the results are summarized as functions of the number n of REAL32-type data transferred: the constant term is due to the initialization time for the communication protocol. All measurements were performed with link speed set at the maximum value.

protocol	*time* (μs)
single	$0.25 \times 32n$
list	$0.14 \times 32n + 3.0$
vect	$0.074 \times 32n + 5.5$

Table 1: Time needed to transfer n REAL32-type data, with three different protocols.

6.2 "Pipeline" program

The results on computing time measurements for each process involved in the *pipeline* program and the relative data exchange times are summarized in table 2.

Process	T_{ex} (μs)	T_{me} (μs)
element.analysis	892	18
event.analysis	520	240
inout	-	53

Table 2: Maximum execution time T_{ex} and message exchange time T_{me} for each process of the *pipeline* program; values are expressed in microseconds.

The maximum time for the *pipeline* configuration to get the final results for a calorimeter made of 20 detector layers is $T_p = 2740$ μs. This value is obtained by adding all single process execution times and all the required message exchange times, mainly contributed by 1020 μs for the message exchanges in the transputers running the *element.analysis* process. The value T_p overestimates the real computing time due to possible overlaps of execution for the *element.analysis* and *event.analysis* processes. A lower limit can be evaluated by neglecting the computing time of the *element.analysis* process, as in the case of very small size events: under this hypothesis $T_p^* = 1850$ μs is obtained.

6.3 "Tree" program

The results on computing time measurements for each process involved in the *tree* configuration and the relative exchange times are summarized in table 3 for the case of a calorimeter with 20 layers. The maximum elaboration time for the *tree*

Process	*Node*	T_{ex} (μs)	T_{me} (μs)	n
element.analysis	T_i	892	-	-
partial.sums	SL_i	295	29	10
partial.sums	TL_i	198	57	22
event.analysis	*end.node*	573	86	34
inout	-	-	31	-

Table 3: Maximum execution time T_{ex} and message exchange time T_{me} for each process involved in the *tree* configuration; time values are expressed in microseconds; n is the number of transferred messages per process.

configuration to get the event data vectors for a 20 element calorimeter is $T_t = 2060$ μs; this value is obtained combining all execution and message exchange times.

6.4 Event processing rates

In estimating the event processing rates of the *pipeline* and *tree* programs the following facts were taken into account:

- the execution time strongly depends on the amount of data in input: an electromagnetic shower requires a longer time than a minimum ionizing particle track; for both configurations rates are computed as averages over events corresponding to the worst possible conditions (all e.m. showers);
- in the *tree* configuration the time needed per event should be equal to that of the slowest process, i.e. *element.analysis* (890 μs), the other processes being completed in parallel in a shorter time. This time is independent of the number of detector layers, all analysed in parallel.

From the values quoted in tables 2 and 3 the maximum event processing rates can then be estimated to be lower than 540 events/s in the *pipeline* configuration and of the order of 1120 events/s for the *tree* configuration.

The total estimated computing time per event T_e was compared with the measured one T_m, experimentally obtained with the network of ten transputers for both architectures: results are shown in table 4.

network	$T_e(\mu s)$	$T_m(\mu s)$
pipeline	1400	1350
tree	890	890

Table 4: Total computing time per event in microseconds, for the case of six detector layers: T_e is the estimated value, T_m is the measured value.

The observed difference of 12% between estimated and measured event processing rates can be understood as due to the overlap in process execution in the different transputers, only approximately taken into account in the estimated values.

The programs used for the measurements described above included the computation of all the global variables described in section 4.2, but did not perform any type of event classification. Further tests showed that multi-dimensional classification algorithms based on feed-forward neural networks [8] with four global variables in input, could be executed in the *end-node* transputer in less than 60 μs, increasing the event processing time by less than 7%. This time can be reduced devoting more than one transputer to the neural network algorithm.

7 Conclusions

Transputer networks were used to fully reconstruct electromagnetic showers and charged tracks in a fine-grained electromagnetic calorimeter.

Of the two types of networks investigated, the *tree-like* configuration gave as expected the best performance, completing the analysis of electromagnetic showers in less than 0.9 ms per event, independently of the number of detector layers, analysed in parallel.

This result was obtained dedicating one transputer to the analysis of each detector layer, and can still be improved by roughly a factor two by devoting two transputers to the analysis of each layer, that is one per projection (xz and yz).

Indications were obtained that an event classification based on neural network algorithms only marginally increase the total computing time per event. This is a promising result in view of possible applications of transputer networks in high-level calorimetric triggers in particle physics experiments.

Acknowledgements

We would like to thank the members of the Wizard Collaboration for useful discussions and for making the calorimeter test data available to us.

References

[1] R.W.Dobinson, J.L.Pages and J.C.Vermeulen, Transputers in Particle Physics Experiments, Particle World, Vol.2, N.2, p. 39-45, 1991.

[2] A.Bernasconi et al., UA6 Collaboration, Phys. Lett. B206 (1988) 163.

[3] S.Bhadra et al., The use of transputers in the ZEUS on-line system, Comp. Phys. Comm. 57 (1989) 316.

[4] G.Barbiellini et al., A silicon imaging calorimeter prototype for antimatter search in space: experimental results, INFN/AE-92/17, 26 May 1992, to be published in Nucl. Instr. Meth.

[5] D.Pountain and D.May, A tutorial introduction to Occam programming, BSP books, London, 1987.

[6] The Transputer Applications Notebook, INMOS Limited, 1989.

[7] The Transputer Databook, INMOS Limited, 1989.

[8] P.Garlatti et al., A neural network for e/π classification in a calorimeter, INFN/AE-92/14, April 27, 1992, to be published in the Proceedings of the II International Workshop on Artificial Intelligence for High Energy Physics, L'Agelonde (France) 1992.