Real Time HEP Applications using T9000 Transputers, Links and Switches

Thesis submitted in accordance with the requirements of
the University of Liverpool
for the degree of Doctor of Philosophy

by

Roger Heeley

October 1996

Acknowledgements

Firstly I would like to thank my joint supervisors Bob Dobinson and Erwin Gabathuler for giving me the opportunity to carry out the work presented within this thesis and for their support and guidance over the last three years.

I would also like to thank the European Union for my funding through the GPMIMD ESPRIT project (P5404), CERN for hosting this project and the CPLEAR collaboration for their support.

I would like to thank the four people I have worked most closely with for the last three years: Dave Francis, William Lu, Michael Ward and Jean Moonen. All of whom have played an important part in allowing me to carry out the work for this thesis.

Thanks also to Stuart Fisher who has worked extensively on benchmarking T9000s and carried out the work on the TransAlphas.

Our hardware group was crucial for the completion of the work within CPLEAR, I would like to thank Brian Martin, Minghua Zhu and Xinjah Liu. In addition to hardware support Minghua also carried out the memory benchmarks presented within chapter 2.

A thank you also to the GPMIMD industrial collaborators: Parsys, Telmat and SGS Thomson.

Last but obviously not least I would like to thank my family and friends. In particular, I would mention my parents, Suzanne, Tony and Julie, without whom this thesis would never have been started. Also a collective thanks to the Heeleys, Ousleys, Hardmans and Smiths (including pets).

Real Time HEP Applications using T9000 Transputers, Links and Switches.

Roger Heeley

Requirements for triggering systems in future generations of High Energy Physics (HEP) experiments in terms of computing power and communications performance are orders of magnitude greater than those of existing experiments. Point to point links and switches interconnecting microprocessors allow the construction of the required large systems scalable in terms of both computational and communications performance. There are relatively few "open" technologies currently available which allow a practical investigation of microprocessors interconnected by point to point links and switches. The IEEE 1355 standard including DS links is one such technology. This thesis presents the application of this technology to two HEP experiments: CPLEAR and a future generation experiment the Large Hadron Collider (LHC) at CERN.

The CPLEAR application involved the use of the T9000 in an on-line event filtering farm. The aims were to demonstrate the feasibility of replacing the existing off-line analysis system, demonstrate reliable operation of the technology and to obtain information for experiments at the LHC. The application of the technology to the LHC allows extrapolations to predict the capability of currently available technology to meet the requirements of future generation experiments. The work also identifies a set of factors which will be crucial in the performance of future HEP systems. The technology presented within this thesis may not be the technology to be used at the LHC, however, the conclusions of this thesis should provide valuable information for the construction and design of the future triggering systems.

Table of Contents

Chapter 1 Introduction 1

1.1 Motivation 1

1.2 Outline of thesis.. 2

1.3 Context of thesis..3

1.3.1 Authors work.. 3

Chapter 2 The T9000 Transputer and DS links......................5

2.1 Transputers..5

2.2 Occam 6

2.3 The T9000 Transputer..8

2.3.1 T9000 Architecture 10

2.3.2 DS Links 10

2.3.3 Virtual Channels and the Virtual Channel Processor (VCP)......... 12

2.3.4 The Hardware Scheduler..14

2.3.5 Memory system..16

2.3.6 T9000 Known Hardware Bugs and Problems.............................. 19

2.4 The C104 Packet Switch..19

2.4.1 Hardware Bugs..21

2.5 The T9000 Toolset Development System.. 21

2.5.1 Problems with the Toolset..24

2.6 Hosting T9000 Systems.. 24

2.7 The GPMIMD Machine..25

2.8 The TransAlpha Module.. 27

2.9 Conclusions..28

Chapter 3 Evaluation of Technology 31

3.1 Communication Benchmarks..31

3.1.1 Single Link Results..31

3.1.2 Dependence on memory bandwidth.. 33

3.1.3 One to four physical links..34

3.1.4 Dependence on clock speed..35

3.1.5 Single Network GPMIMD Machine Results................................35

3.1.6 Multiple Network Results..38

3.2 Interrupt Response and Context Switch Results.. 40
3.2.1 Timeslice operation..40
3.2.2 Timer generated interrupts..40
3.2.3 Interprocess communications..40
3.3 Supervisor/Control processor benchmarks..41
3.3.1 Short Message Sends... 42
3.3.2 Combined Communication and Computation.................................44
3.4 Computation Benchmarks..47
3.5 Comparison to other platforms..47
3.5.1 VMEbus PowerPC systems running LynxOS.................................47
3.6 Conclusions..49

Chapter 4 The Application of the T9000 to CPLEAR............ 53

4.1 Motivation.. 53
4.1.1 Value to CPLEAR...53
4.1.2 Proof of existence and operation of the T9000 and C104..............54
4.1.3 Use of the T9000 for future generations of HEP experiments.......54
4.2 The CPLEAR Experiment..54
4.2.1 Physics Background...54
4.2.2 Event Detection...55
4.2.3 Data Acquisition... 56
4.3 T9000 Processor Farm System Overview... 57
4.4 Play-back mode...59
4.5 Event types...60
4.6 Porting of CPREAD..61
4.7 T805 to T9000 Interface...61
4.7.1 Overview... 61
4.7.2 An example T805 to T9000 interface.. 64
4.7.3 Extracting packet headers for the gateway T805........................... 67
4.7.4 Remaining problems.. 68
4.8 On-line Monitoring...68
4.9 History of Installations and associated problems..71
4.9.1 CPLEAR run June 1994...71
4.9.2 CPLEAR run September/October 1994..72
4.9.3 Practical problems and solutions..75
4.10 Results..81
4.11 Boosting computational power using TransAlpha modules........................83
4.12 Summary..84

Chapter 5 DS link technology applied to the LHC 87

5.1 Introduction..87

5.2 Atlas.. 88
- 5.2.1 Overview...88
- 5.2.2 The level 2 trigger...89
- 5.2.3 The level 3 trigger.. 91

5.3 GPMIMD machine applied to level 2 and level 3.....................................91
- 5.3.1 level 2 results.. 91
- 5.3.2 level 3 results.. 96
- 5.3.3 Extrapolation for level 2 to the SCT subdetector...........................99
- 5.3.4 Extrapolation for level 3 to the SCT subdetector...........................101
- 5.3.5 Conclusions..101

5.4 Comparison with other parallel platforms...102
- 5.4.1 Benchmark 1.1 (one-way)...103
- 5.4.2 Benchmark 1.2 (two-way)... 105
- 5.4.3 Benchmark 1.3 (all to all).. 106
- 5.4.4 Benchmark 2.1 (push farm with supervisor)...................................107
- 5.4.5 Summary of comparisons to other platforms..................................108

5.5 Conclusions..108

5.6 Future Work..111

Chapter 6 Concluding remarks 113

6.1 Original goals..113

6.2 Summary of results...113

6.3 Achievements..114

6.4 Future work...116

6.5 Final remarks...116

References 117

Chapter 1
Introduction

1.1 Motivation

Requirements for triggering systems in future generations of High Energy Physics (HEP) experiments in terms of computing power and communications performance are orders of magnitude greater than those of existing experiments [1]. The experiments will increase in complexity, event rates, event sizes and the amount of computationally intensive analysis required in real-time. The only way to meet these continually increasing requirements is the use of large arrays of processors interworking via high speed interconnects. The computational and communications performance of such systems must scale to meet the dynamic requirements of the experiment.

Existing experiments largely base their trigger systems on standard (shared) buses such as CAMAC, VME and FASTBUS. This approach is being replaced by a migration to microprocessors communicating via high speed point to point links interconnected by switches. Point to point links and switches interconnecting microprocessors allow the construction of the required large systems scalable in terms of both computational and communications performance. Bus based systems are limited in scalability, they suffer from bottlenecks in the bus bandwidth and the limited interconnectivity available between multiple buses.

Table 1.1 contains a summary of existing HEP experiment requirements compared to the requirements of future generation experiments. The increases in reduction factors performed by the trigger are three orders of magnitude. These increases in rejection factors demand increased computational power available in real-time.

There are relatively few “open” technologies currently available which allow a practical investigation of microprocessors interconnected by point to point links and switches, and no clear winner in industry. In particular, few technologies are available to allow an investigation into their use in triggering and data acquisition in HEP. The IEEE 1355 standard [2] including DS links, the T9000 Transputer [3] and associated C104 packet switch [4] are technologies that have already been used to build substantial systems. A Transputer is a complete microcomputer with four integrated on-chip links allowing multiple processors to be interconnected. Using these links Transputers are directly connected or connected via switches.

As part of the ESPRIT project, GPMIMD [5] a 64 node T9000 machine using C104 packet switches has been constructed. ESPRIT is the European Strategic Programme of Research and Development in Information Technology. This thesis presents the application of this technology to two HEP experiments: CPLEAR[1] and a future generation experiment the

1. CP violation at the Low Energy Anti-proton Ring [6].

Large Hadron Collider (LHC) at CERN[2]. For the LHC, I will investigate the application of the technology to the Atlas detector [7].

TABLE 1.1 Comparison between existing and future generation HEP experiments

	CPLEAR	A typical LEP[a] experiment	A typical LHC experiment - to be constructed in 2005
Raw Event Rate	10^6 Hz	5×10^4	10^9 Hz
Trigger Reduction	4×10^{-4}	~ 10^{-4}	10^{-7}
Number of Channels	3×10^4	$<10^6$	$>10^7$
Event Size	~ 2 Kbytes	~ 300 Kbytes	~1 Mbyte
Final Data Rate	~ 1 Mbyte/s	~ 1 Mbyte/s	10-100 Mbyte/s

a.Large Electron Positron collider at CERN.

The CPLEAR application involved the use of the T9000 in an on-line event filtering farm. The aims were to demonstrate the feasibility of replacing the existing off-line analysis system, demonstrate reliable operation of the technology and to obtain information for experiments at the LHC. The T9000s and C104s used for the CPLEAR application were prototypes, when the work was carried out, little or no experience existed in building large systems with these components.

The application of the technology to the LHC allows extrapolations to predict the capability of currently available technology to meet the requirements of future generation experiments. The work should also identify a set of factors which will be crucial in the performance of future HEP systems. The technology presented within this thesis may not be the technology to be used at the LHC, however, the conclusions of this thesis should provide valuable information for the construction and design of the future triggering systems.

1.2 Outline of thesis

This introduction chapter gives the general motivation for the thesis, an outline of the thesis and the background in which the work was carried out. Chapter 2 presents the technology of Transputers, links and switches along with supporting concepts and an introduction to parallel processing.

Chapter describes an evaluation of the technology, which identifies a set of critical factors that dictate the performance of multi-processor systems. Included in the evaluation are communications, interrupt response, context switching and computational performance. In addition the performance of the T9000 is compared to other platforms.

Chapter 4 contains the application of the T9000 to the CPLEAR experiment operating as an on-line event filtering farm. Results are presented, problems discussed and projections are made to demonstrate the feasibility of using the technology to build processor farms in CPLEAR. Chapter 5 outlines the application of DS link technology to the LHC, a continuation of the CPLEAR application. Results are presented and discussed, the work has resulted in a set of factors affecting the performance of DS links applied to triggering at the LHC.

2. The European Laboratory for Particle Physics Research.

The factors are of relevance to any future system based on switching networks and point to point links. Some of these factors have been investigated, and those which have not are presented as future work. The final chapter, chapter 6, is a summary of the conclusions presented throughout the thesis.

1.3 Context of thesis

The work presented within this thesis was carried out at CERN within the ESPRIT project GPMIMD, a collaboration between fourteen European industrial partners. An objective of the project was the construction of a parallel scalable computer using the T9000 and C104. The role of CERN within this project included the mounting of HEP applications onto this machine. These applications are only a small part of the project, however, the successful operation of the technology in the demanding environment of an HEP experiment and the availability of substantial demonstration systems was a crucial aspect of the project. This was recognised as such by our industrial partners and the external project reviewers appointed by the European Union. Work carried out within this project has lead to the basis of work for this thesis.

1.3.1 Authors work

The benchmarking and technology evaluation presented within chapter 3 is the authors own work. This includes the writing of benchmarks to assess communications, interrupt response and context switch performance. Any results from other sources used for comparison are clearly referenced at the point they are presented.

Chapter 4 is entirely the authors own work. This includes the design and implementation of the T9000 system and the integration with the existing CPLEAR data acquisition system. All extrapolations and conclusions are the authors own work. Clear references are given where existing systems or software produced by other persons are used.

In chapter 5 all extrapolations, interpretation of results and conclusions are the authors own work. Some results were produced in collaboration with another member of the group.

Chapter 6 is entirely the work of the author.

Chapter 2
The T9000 Transputer and DS links: hardware and software

In this chapter the relevant technologies used are presented, including supporting concepts and a general introduction to parallel processing using Transputers.

The requirement for processing performance in HEP data acquisition, and more generally real-time embedded systems, is continuously increasing as systems become more complex. The only way to satisfy these requirements in the long term is the use of multiple processors which provide efficient support for multiprocessing. Crucial issues will be the inter-processor communication performance, efficient support for concurrent communication and computation, fast interrupt response times and context switch times. The T9000 Transputer has been specifically designed to provide high performance in all of these areas.

Of equal importance is the software development environment and host system software. Multiprocessing systems require complex real-time monitoring and fault tolerance control systems. The T9000 provides dedicated control links for configuration, monitoring and control.

The Transputer model of concurrent programming is presented, followed by details of Transputer and switch implementations, i.e. the T9000 and C104. The software development environment and host system software for the T9000 is presented and a critique given. The TransAlpha module is presented which combines the T9000 with a DEC Alpha processor to boost its computational power.

2.1 Transputers

A Transputer is a complete microcomputer on a single VLSI chip. Each Transputer has a number of communication links, allowing them to be interconnected. These links allow concurrent programming in multi-processor Transputer networks. The Transputer instruction set contains single instructions to send and receive messages through these links, minimizing delays in inter-Transputer communication. Transputers can be directly connected, to form specialised networks, or can be interconnected via switches.

Transputers were explicitly designed for multi-processing, in addition to the specialised hardware functionality a Transputer model of concurrency has been developed. The programming model for Transputer systems consists of sequential processes which communicate using message passing, based on Hoare’s Communicating Sequential Processes (CSP) [8]. Each process can be viewed as a black box with internal state, which can communicate and synchronise with other processes via point to point communication channels, as shown in Figure 2.1.

Communication is synchronous, if one process is not ready, both processes will be suspended. The correct input/output instruction must be executed on both communicating processes for the communication to take place.

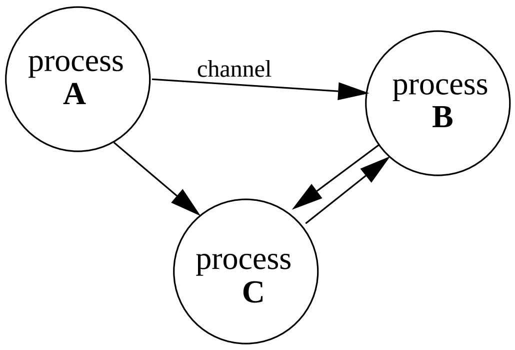


FIGURE 2.1 The Transputer Model Of Concurrency

Each Transputer has an on chip hardware process scheduler which allows it to share its time between a number of processes. Communication between processes on the same Transputer is performed using the local memory; communication between processes on different Transputers is performed using a link. Consequently, a program can be executed either by a single Transputer or by a collection of Transputers in a network. The same communication model is maintained irrespective of whether processes are located on a single Transputer or a network of Transputers. Figure 2.2 shows two possible mappings based on the processes presented in Figure 2.1.

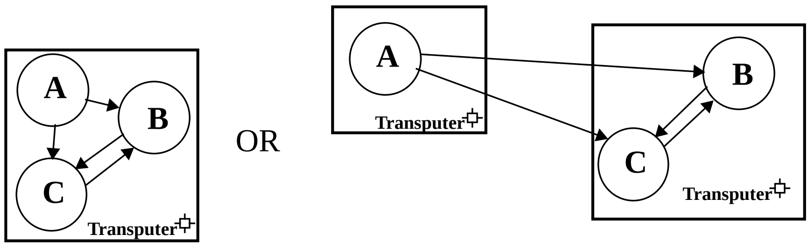


FIGURE 2.2 Mapping Of Processes to Processors

An application is not dependant upon a particular network configuration, thus a program can be run on various networks allowing optimisations in cost and performance.

2.2 Occam

Occam [9] is a parallel processing language based on Hoare’s CSP. The Transputer was developed with the specific intention of providing an efficient platform for the execution of the occam programming language. Other languages exist for the Transputer (for example C) but none has such a close relation to the Transputer instruction set, hence other languages are often less efficient and require larger amounts of code. Occam is a language designed for parallel processing, C is not. Unless otherwise noted, all coding of Transputer programs within this thesis has been performed using occam.

An occam program consists of 1 to N concurrent sequential processes, that can communicate via point to point links, mirroring the Transputer model of concurrency.

All occam programs are built from three primitive operations; assignment, input and output. The primitives produce processes by specifying their order; sequential or parallel. In addition to these primitives there is support for repetition and conditional statements. Data objects and their types may also be defined. Table 2.1 lists the three occam statements which describe the relation between multiple processes; SEQ (sequential), PAR (parallel) and ALT (alternate).

TABLE 2.1 Important occam statements

Statement	Syntax	Description
SEQ construction	SEQ *<process 1>* ... *<process N>*	A collection of processes to be executed in sequence, where a process is any occam construct.
PAR construction	PAR *<process 1>* ... *<process N>*	A collection of processes to be executed concurrently with each process starting execution at the same time.
ALT construction	ALT *<input 1>* *<process 1>* ... *<input N>* *<process N>*	Select one process from the list of alternative processes to be executed depending on which input guard has input available first.

The five lines of occam in Figure 2.3 demonstrate the ease with which occam can provide communication of variables between processes. An output on a channel is performed using ‘!’ and input is performed using ‘?’.

The channels talkin and talkout are connected to one another in a configuration language, which will also map the code onto a processor. Details of the configuration process are presented in Section 2.5, “The T9000 Toolset Development System,”. The code would have the following effect: two processes would run concurrently, in the first process data would be sent on the channel talkout from variable a, in the second process data is received on the channel talkin into the variable b.

The equivalent task written in C is considerably longer and more complex, see Figure 2.3, therefore immediately less readable. A set of libraries providing the C sub-routine equivalent functions and predefined data types of the occam constructs extend the C language for concurrent programming. In the example the functions ChanInInt and ChanOutInt are used to send integers on a channel. ProcAlloc is used to reserve memory space and allocate heap space for processes. The function ProcPar runs multiple processes concurrently and ProcAllocClean releases all memory used by a process.

There are clear problems when constructs for parallel programming are added on to a language after its initial design. The occam programming language was designed specifically for parallel processing.

Familiarity with C or existing C code, which can be directly compiled, are possible advantages of using C. The disadvantages are the increased code complexity and the possibility of losing performance.

occam required to output an integer on a channel

```
CHAN OF INT talkin,talkout:
INT a,b:
PAR
    talkout! a
    talkin? b
```

C required to output an integer on a channel

```
void out_proc (Process *p, Channel * c)      /* Code to send integer */
{
      int a;
      ChanOutInt (c,a);
      return;
}
void in_proc (Process *p, Channel *c)         /* Code to receive integer */
{
      int b;
      b = ChanInInt (c);
      return;
}
int main(void)
{
   Channel * talkin, * talkout;                  /* Declare two channels */
   Process * in, * out;                          /* Declare two processes */
   in=ProcAlloc (in_proc, 0, 1, talkin);
                                            /* Allocate memory for processes */
   out=ProcAlloc (out_proc, 0, 1, talkout);
                                         /* Use procedures in_proc & out_proc */
   ProcPar (in,out,NULL);        /* Execute processes ‘in’ and ‘out’ */
   ProcAllocClean(in);                           /* Release memory */
   ProcAllocClean(out);                          /* Release memory */
}
```

FIGURE 2.3 A comparison between occam and C

2.3 The T9000 Transputer

The T9000 is the latest generation of Transputers from SGS Thomson (see Figure 2.4). It has a 32-bit pipelined processor with a 64-bit FPU and 16 Kbytes of cache. There are four bi-directional serial data links and a Virtual Channel Processor (VCP) allowing efficient T9000-to-T9000 communications. There are two on chip 32 bit timer clocks with up to 1 microsecond resolution. These components are combined onto a single integrated circuit.

The T9000 has several improvements over previous generations of Transputers in both performance and functionality. Improved performance has been gained through an increase in design clock speed (50 MHz design), the implementation of an on-chip cache, and a pipelined superscalar architecture. This architecture allows multiple instructions to be executed every processor cycle.

Two separate control links allow the T9000 to be controlled (processor initialization and loading) and monitored for errors, even when there are faults in the normal communications network. These control links may be daisy chained and/or connected via C104 packet switches. A full description of the T9000 is given in the following sections.

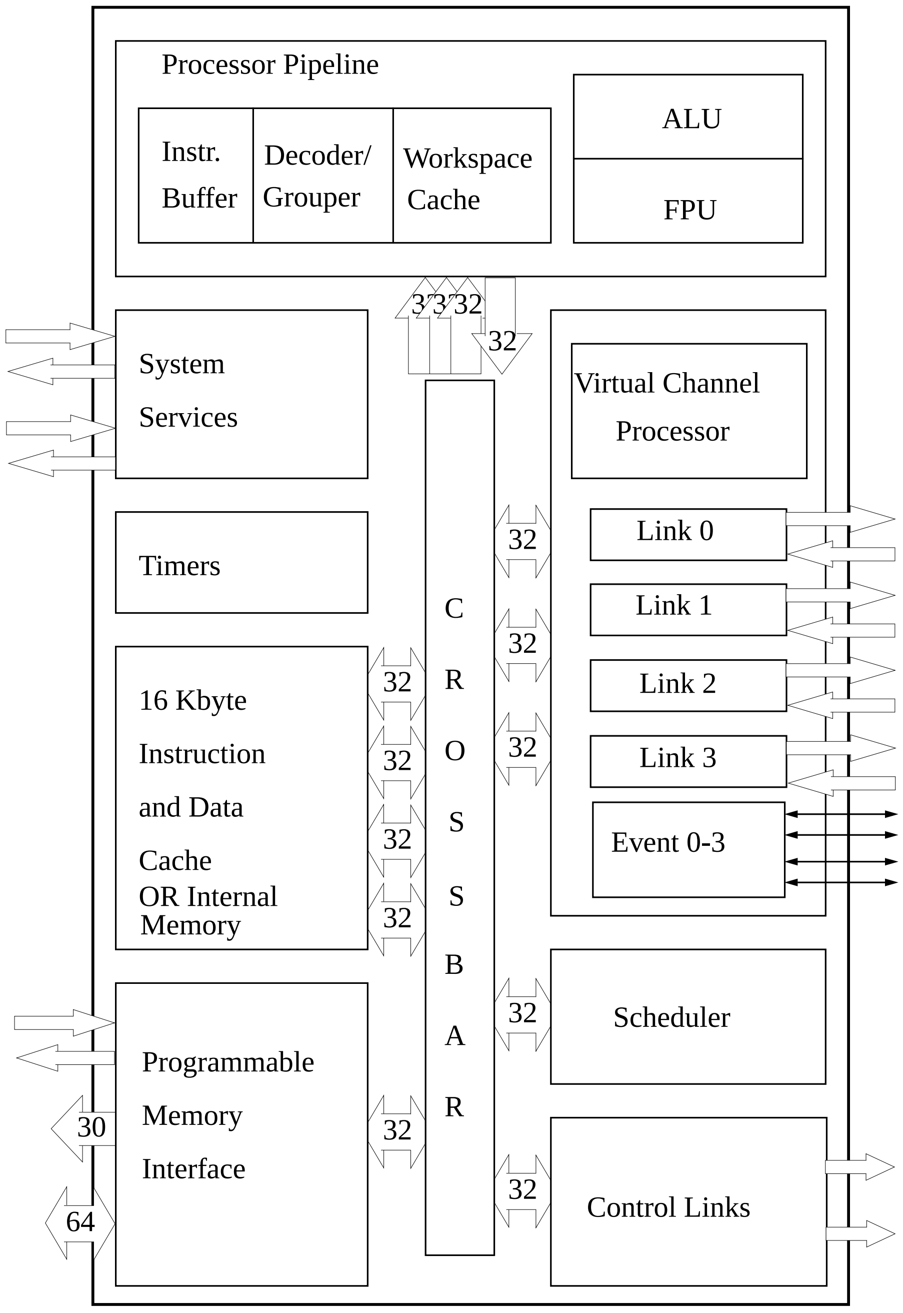


FIGURE 2.4 The T9000 Transputer

2.3.1 T9000 Architecture

The CPU contains three registers (Areg, Breg and Creg) used for expression evaluation which form a hardware stack. The floating point unit also contains three registers to form an evaluation stack. The Transputer uses two more registers to execute code: the instruction pointer to the next instruction to be executed and the workspace pointer (see Figure 2.5). The workspace is an area of memory where local variables are stored, which also has a dedicated cache. The workspace cache can hold up to the first 32 words of the process workspace, allowing fast access to the variables stored in this memory.

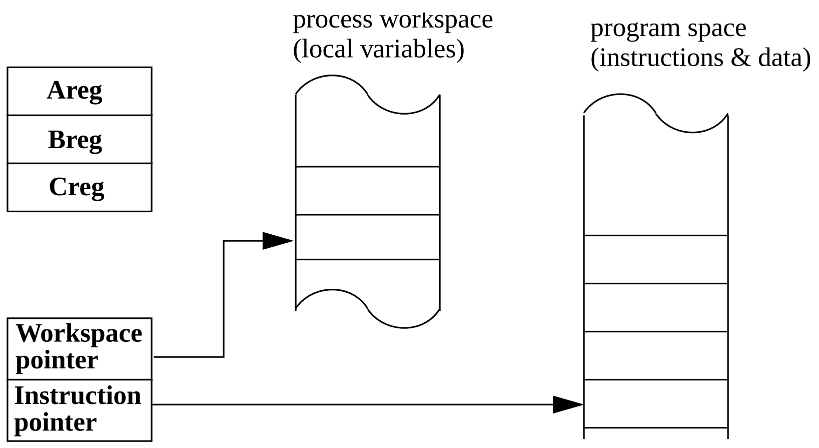


FIGURE 2.5 T9000 Architecture

2.3.2 DS Links

Improved communication is provided by the new Data/Strobe (DS) link technology which currently operates at 100 Mbits/s. I have carried out tests running C104 links successfully at 200 Mbits/s [3], and T9000 links at up to 160 Mbits/s.

The DS link technology is part of the IEEE 1355 standard. The standard covers the physical connectors, cables, and electrical/logical protocols for implementing point to point serial interconnects operating at speeds of 100 Mbits/s (DS) and at 1 Gbit/s (HS) over copper and optical fibres. The DS links run over printed circuit and cable interconnections.

The DS link uses four different levels of protocol:

- **Bit level**. The DS link uses 4 wires: a data/strobe pair in each direction. The protocol guarantees that exactly one of the two wires (for a single direction: data and strobe) will have an edge in every bit frame. The levels on the data wire are the transmitted bits, see Figure 2.6.

3. SGS-Thomson plan to produce products using DS links at over 200 Mbits/s

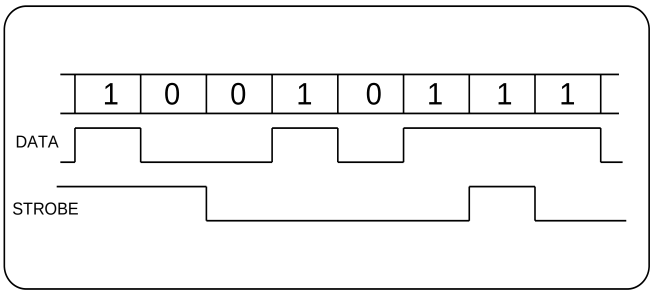


FIGURE 2.6 Bit Level Protocol on a Data Strobe link

- **Tokens**. The next level of protocol is the token, there are two types of token: data and control. Data tokens are simply data bytes, but there are multiple control tokens most of which are used for flow control and alignment of clocks against skew. Figure 2.7 shows the construction of the tokens.

Token = Parity + Function + Body
10 bit Data Token = P 0 b b b b b b b b (8 data bits + 2 others)

Control Tokens

Token	**Symbol**	**Bit Pattern**
Flow Control	FCT	P 1 0 0
End of Packet	EOP	P 1 0 1
End of Message	EOM	P 1 1 0
Escape	ESC	P 1 1 1
Disconnect Detection	NUL	ESC + P 1 0 0

P = parity bit
b = data bit

FIGURE 2.7 Token Level Protocol

- **Packet**. A packet is a sequence of tokens with a specific order and format: a header (containing routing information), zero or more data tokens and then an end of packet control token (see Figure 2.8). The standard does not specify a limit to the number of data tokens contained in the packet.
- **Message**. Messages are a sequence of packets. The final data packet sent in the message is terminated with an end of message token. The message protocol used by the T9000 is presented in the next section: “Virtual channels and the T9000 virtual channel processor”.

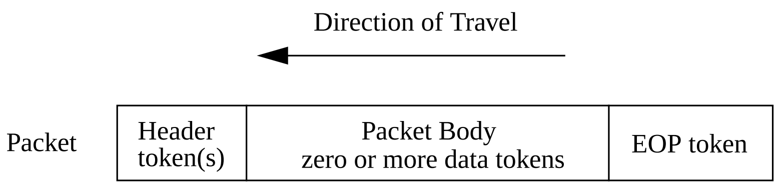


FIGURE 2.8 Packet Level Protocol

The output side of a DS link is set to run at a particular speed, the input side is clocked entirely by the incoming data, it does not need to know what speed to expect the incoming link to run at.

One of the four control tokens is an escape which allows for a number of composite control tokens made up of short sequences of 4 bit control tokens. DS link modules send a continuous stream of tokens, when there are no other tokens to send the link modules exchange composite NUL tokens. The link should continuously receive tokens, the absence of tokens on the link results in a link failure which is detected immediately. The failure is known as a link disconnect failure.

T9000 processors can be directly connected using their DS links or connected to a network of C104 packet switching chips, thus allowing the construction of large networks with scalable communication bandwidth between nodes. In the latter case, additional packet headers are required to route the packet through the switching network.

2.3.3 Virtual Channels and the T9000 Virtual Channel Processor (VCP)

Communication between T9000 processes is performed via virtual channels. A virtual channel is a single logical communication connection between two processes mapped onto a physical processor link (see Figure 2.9). A single physical link may carry up to 64,000 virtual channels, the current implementation limit. Processes on any two Transputers may be directly connected by a channel regardless of their relative positions in the network. The connection and switching between the two processes can either be performed using C104 packet switches or if they are not available by software virtual channel routing on other T9000s. Virtual channels allow any application to be mapped onto a network configuration regardless of the network topology.

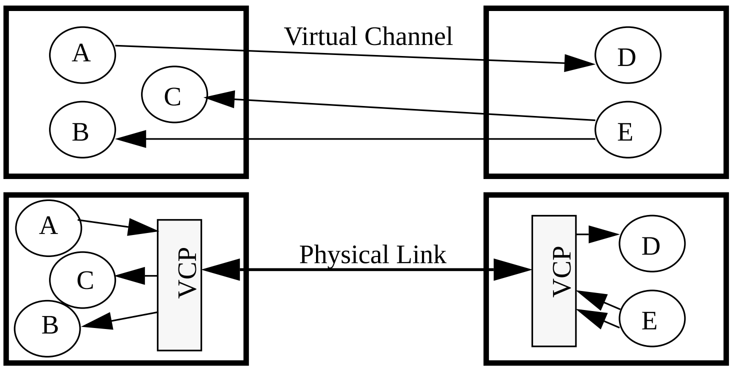


FIGURE 2.9 Virtual Channels

The VCP of the T9000 is a hardware communications processor which multiplexes the virtual channels onto a specified physical processor link. Packets from separate virtual channels are interleaved onto the physical link, allowing separate processes to communicate simultaneously. The virtual channel to which the packet is being sent is contained in the packet header. A virtual link is comprised of two virtual channels, one in each direction.

Messages are divided into a sequence of packets, each of which has the structure shown in Figure 2.10. All routing information is contained in the packet header. The T9000 implements a maximum packet body of 32 bytes. For example, a 48 byte message will be sent as a 32 byte packet followed by a 16 byte packet. Any device receiving a data packet replies to the sender with an acknowledge packet. No further packets are transmitted by the sender until the corresponding acknowledge has been received by the sender. The transmission of the packets on a single virtual channel is synchronous.

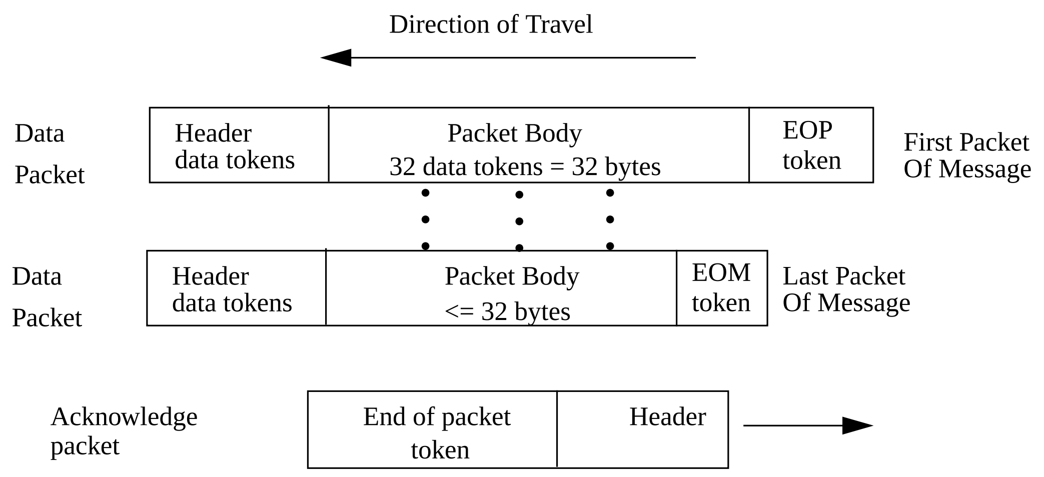


FIGURE 2.10 Packets on a DS Link

There can be only one message transmitting on a virtual link at any time. A process can only be communicating on a single virtual link at any time. If a single virtual link is in use, a packet is transmitted, then the virtual link is idle until the acknowledge for that packet is received. This utilises a small proportion of the available link bandwidth. If multiple virtual links are used (concurrently) then the first virtual link (waiting for an acknowledge) stays idle but another virtual link can use the physical link. The virtual links would require separate concurrent processes to drive them. If enough virtual links are used concurrently then the physical link can be saturated.

The VCP enables virtual links to share physical links. To achieve this, it maintains queues for each physical link. The receipt of an acknowledge packet for a data packet puts the next packet for that message (if any are remaining) on the back of the transmit queue for the appropriate link. The VCP alternately sends from the queue of data packets and acknowledge packets from high priority processes until both are empty. It will then send data and acknowledge packets from low priority processes. Packetisation, acknowledgements and link queuing are all handled by the VCP.

Each virtual link in a T9000 is represented by a structure called the virtual link control block (VLCB). The VLCBs are used to store the current status of communications on the virtual links. It also stores all header and routing information that is required for the packets to reach their destination virtual link.

2.3.4 The Hardware Scheduler

In most embedded systems there is a need for a real-time response (both to external interrupts and efficient context switching in multi-tasking systems). In all HEP data acquisition systems investigated within this thesis there are often strict real-time requirements.

The hardware scheduler implements the Transputer model of concurrency, handling of interrupts and process scheduling. The T9000 CPU uses very few registers to store the state of the current process. The result is that there is little information to be stored when there is an interrupt or process switch, hence the operations are fast. In addition the operations are performed by a dedicated hardware scheduler, improving performance.

Both internal and external communications performance are affected by the efficiency of rescheduling and descheduling processes. In general, communications are performed using the following three steps: deschedule process, perform communication then reschedule process.

2.3.4.1 The Transputer model of concurrency

The hardware scheduler on the T9000 allows the implementation of the Transputer model of concurrency, multiple processes executing concurrently on a single processor. A process can have one of two priorities: high or low. At any time, a transputer process may be:

ACTIVE	or	INACTIVE
-Being executed, or -Descheduled by higher priority process, or -On a list waiting to be executed		-Ready to input, or -Ready to output, or -Waiting until a specified time, or -Waiting on a semaphore

The inactive processes do not consume any processor time. The active processes waiting to be executed are held in a list of process workspaces. This is implemented using two registers, one of which points to the first process on the list, the other to the last. In Figure 2.11, A is executing, and B, C and D are active, awaiting execution. A separate queue is maintained for low and high priority processes, with low priority processes only being given CPU time when the high priority queue is empty.

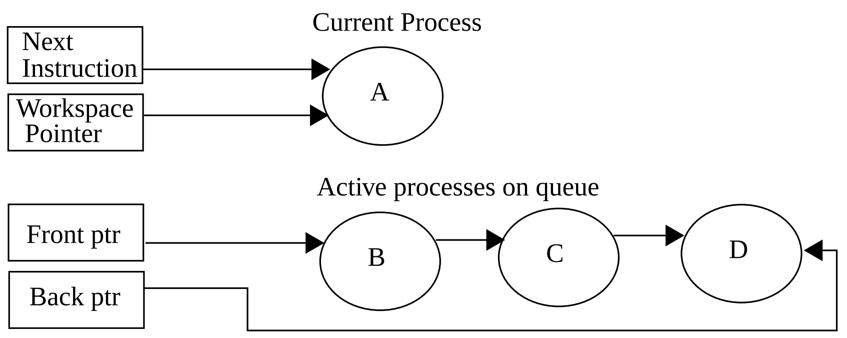


FIGURE 2.11 T9000 Process Queue

2.3.4.2 Low priority processes

A low priority process executes until it is unable to proceed, and is descheduled for one of the following reasons:

- It initiates I/O, i.e. waiting to input or waiting to output
- It is waiting for a value of the timer to be reached
- Any high priority process becomes active. Low priority processes will be pre-empted by any high priority process that is ready to execute (active).
- It has occupied the CPU for it's 'timeslice' which is approximately 1 millisecond. The process is descheduled and added to the end of the low priority process queue. When no high priority processes are active the CPU is shared between all active low priority processes, each is allowed a timeslice before being descheduled. This process is known as timeslicing.

The next four points detail the requirements that must be satisfied before a low priority process can be rescheduled. In addition, the process will always have to wait until there are no active high priority processes before it can execute:

- If descheduled by I/O, the I/O must be completed before the process can be rescheduled
- If descheduled waiting for a timer, the required value of the timer must be reached
- If descheduled by a high priority process, that high priority process must become inactive
- If descheduled after executing for a timeslice period, all other active low priority processes must be allowed to execute before the process can be rescheduled

2.3.4.3 High priority processes

A high priority process executes until it is unable to proceed, and is descheduled for one of the following reasons:

- It initiates I/O, i.e. waiting to input or waiting to output
- It is waiting for a value of the timer to be reached

When the I/O is complete or a timer value is reached the process becomes active. At that time it will pre-empt any low priority process executing, or if a high priority process is executing it will be added to the end of the high priority process queue.

2.3.4.4 Full and partial context switch

When any current process is descheduled and another replaces it a context switch has occurred. I distinguish two types of context switch: a partial context switch and a full context switch.

A partial context switch can only occur when the current process is executing certain instructions, for example a jump or loop end instruction. The current process has very little 'state' or context that needs to be saved when it is executing a jump or loop end. The values of A, B and C registers are unused for these instructions and therefore do not need to be saved when the process is descheduled. A timeslice only requires a partial context switch.

A full context switch can occur when the current process is executing any instruction. A full context switch requires the entire 'state' or context of the current process to be stored before it is descheduled. The full context switch can occur for any instruction so the values of A, B and C registers and floating point unit registers will need to be stored. A full context switch occurs when a high priority process pre-empts an executing low priority process. The high priority process pre-empts the executing low priority process immediately, regardless of the instruction the low priority process was executing. Shadow registers in the CPU are used to store the state or context of the descheduled low priority process, this process will be the first low priority process to execute when no more high priority processes are active. The rescheduling of the low priority process will also require a full context switch to allow all the state of the process stored in shadow registers to be re-loaded.

A partial context switch should be quicker than a full context switch. The difference being the time required to store the state of a low priority process into the shadow registers.

2.3.4.5 Interrupts

The hardware scheduler also supports the handling of interrupts and timers. Any event that a process may need to wait for (communication, interrupt or timeout) can be treated as a normal communication. The event causing the interrupt would be mapped to a communication channel (seen as a virtual channel). An interrupt handler is a process waiting on an input from the interrupt ‘channel’, in the same way it would wait on a communication channel.

The T9000 has four event pins or channels which can be used for control and synchronisation from external events and devices. The event pins can be configured as input or output. As input they can be used to trigger fast processor response to external signals. As output, the process outputting will be descheduled until the external device has performed the necessary handshaking on the event pin. If an interrupt occurs the processor immediately stops executing the current low priority process and begins executing the high priority interrupt handler (requiring a full context switch).

There are two timers: with 1 microsecond resolution and 64 microsecond resolution. The value of the processor timer can be read, or a timer input executed which will deschedule a process until a certain time is reached (interrupt from timer).

2.3.5 Memory system

The T9000 programmable memory interface (PMI) connects the crossbar of the T9000 to external memory. External memory (addressing up to 4 Gbyte) supports page mode and is partitioned into four banks which allows mixed memory systems. The timing and width of these four banks can be altered, with support for 8,16,32 and 64 bit data buses. If the 64 bit interface is in use then the memory must be cached.

The T9000 has a 16 Kbyte write back cache. Four independent banks each serve one quarter of the whole memory space. When the T9000 comes out of reset the cache acts as 16 Kbytes of internal memory. This allows the T9000 to operate with no external memory, for small amounts of data and code. The only compiler capable of producing code that may run within 16 Kbytes is the oc (occam) compiler. The minimum size of a linked C binary is greater than 16 Kbytes. At configuration time, the cache can have three modes of operation:

- 16 Kbytes internal memory (only on non 64 bit memory interfaces)
- 16 Kbytes cache
- 8 Kbytes internal memory, 8 Kbytes external memory

All measurements in this thesis use T9000s with 70 nanosecond DRAM (Dynamic Random Access Memory). Data is addressed using a memory page address (Row address) and an offset within that page (Column address). If subsequent memory addresses occur within the same page then it is only necessary to specify the column address. This is known as page mode and reduces the processor cycles required to access the memory.

2.3.5.1 Theoretical performance of the PMI

The theoretical performance of the PMI is shown in Table 2.2 for a 20 MHz T9000, giving a processor clock cycle time of 50 nanoseconds. The numbers estimate the rate that external memory can be accessed through the PMI by calculating the number of processor cycles required to transfer a single cache line. This is not affected by the revision type. A cache line is 16 bytes or 128 bits, which is also the size of a memory page. The second column shows the processor cycles required to read 128 bits into the PMI either into or from the external memory.

TABLE 2.2 Theoretical memory performance

Memory type and access mode	Number of processor clock cycles required for single cache line (16bytes)	Theoretical memory bandwidth
32bit, non page mode	16	20.0 Mbytes/s
32bit, page mode	10	32.0 Mbytes/s
64bit, non page mode	8	40.0 Mbytes/s
64bit, page mode	6	53.3 Mbytes/s

2.3.5.2 Memory to memory transfer rates

Table 2.3 shows actual measured performance along with theoretical values based on results in Table 2.2. The results are for memory to memory transfer rates, i.e. the rate at which data can be copied from source to destination. The measurements have been made by copying an array using the T9000 ‘move’ instruction, no page mode. The following points detail the calculation of the theoretical rates:

- Internal to internal, 32 bit interface. To write 32 bits from internal memory to internal memory will require 1 or 2 processor cycles, depending on whether the write and read can be performed in parallel over the 32 bit wide T9000 crossbar. This gives 40 to 80 Mbytes/s. There are four ports between internal memory and the central crossbar of the T9000, see Figure 2.4, each port reads from/writes to one quarter of internal memory. The relative position of the two 32 bit words dictates whether more than one port can be used per clock cycle, i.e. 1 or 2 processor cycles required to transfer a single 32 bit word.
- Internal to external, 32 bit interface. The external bandwidth from the PMI has been presented as 20 Mbytes/s in Table 2.2, requiring 16 processor cycles to transfer a single cacheline from the PMI into external memory. The extra cycles required are those to read the data from internal memory into the PMI. This should be at most one processor cycle for every 32 bits, or 4 per cacheline, which would reduce the bandwidth to 16 Mbytes/s. If the read from internal memory is overlapped with the writes to external memory the performance would stay at 20 Mbytes/s. This is theoretically possible, there are separate ports onto the T9000 crossbar for the PMI and internal memory (see Figure 2.4).
- External to internal, 32 bit interface. The same calculation as for internal to external.

- External to external, 32 bit interface. The bandwidth between two areas of external memory should be half of the rate data can be read into through the PMI, i.e. 10 Mbytes/s.
- External to internal, 64 bit interface. The external bandwidth into the PMI has been presented as 40 Mbytes/s. Again the cycles required to get the data into internal memory from the PMI must be added. At most this would be two processor cycles for every 64 bits, which reduce the performance to 32 Mbytes/s.

TABLE 2.3 Memory to memory transfer rates, Revision D02 T9000s at 20 MHz, non page mode

Array location	Memory interface	Measured memory to memory transfer rate	Number of processor clock cycles required for single cache line	Theoretical rate (Mbytes/s)
Source: Internal RAM Destination: Internal RAM	32 bit	66.4 Mbytes/s	4 to 8	40.0 to 80.0
Source: Internal RAM Destination: External RAM	32 bit	10.8 Mbytes/s	16 to 20	16.0 to 20.0
Source: External RAM Destination: Internal RAM	32 bit	10.8 Mbytes/s	16 to 20	16.0 to 20.0
Source: External RAM Destination: External RAM	32 bit	9.22 Mbytes/s	32	10.0
Source: External RAM Destination: Internal RAM	64 bit	16.4 Mbytes/s	8 to 10	32.0 to 40.0

The three benchmarks which use both internal and external memory result in half the maximum bandwidth expected. This has been investigated by using a scope to monitor the processor cycles required to move data between the PMI and internal memory. For a 32 bit interface I had expected 0 or 1 processor cycles to move the data from the PMI into internal memory, corresponding to the range in theoretical values in Table 2.3. Using a scope 3 cycles were observed to move data between the PMI and internal memory. For the 32 bit interface this reduces performance from 16 Mbytes/s (assuming 1 processor cycle for PMI to internal memory) to 10.8 Mbytes/s (using 3 processor cycles for PMI to internal memory).

The extra cycles required to read or write data between the PMI and internal memory also occur when a link reads/writes from the PMI. These limitations are demonstrated by communication benchmarks presented in the next chapter, the evaluation of the technology.

The problem has been partly solved on the revision E03 T9000. The external to internal memory bandwidth for a 32 bit interface is now 15.4 Mbytes/s. Which is still below the theoretical range of 16 to 20 Mbytes/s, but is a large improvement over 10.8 Mbytes/s for the revision D02. The memory performance will also increase linearly for 25 MHz T9000s, initial tests with 25 MHz revision E03 T9000s show 19 Mbyte/s external to internal transfers with a 32 bit interface.

The T9000 has 4 bi-directional links running at 100 Mbits/s, corresponding to approximately 75 Mbyte/s. With the revision D02 T9000 there is a severe bottleneck in the memory interface when the links must be driven from external memory. This situation is improving with the new revision E03 of the T9000.

The external to external and internal to internal results are as expected. The slight reduction in performance for external to external is caused by the extra DRAM refresh cycles which are required, loop overheads and the time for the processor pipeline to start.

2.3.5.3 Memory transfers using the cache

Memory transfers which fit into the cache will be cached by the first memory transfer, subsequent transfers will effectively be performed as internal memory transfers. Larger arrays will not fit into the cache, at which performance will be limited by the external memory interface. The result is that the performance of a memory transfers can vary from the 9.22 Mbytes/s to the 66.4 Mbytes/s in Table 2.3 depending on whether the message will fit into the cache. The size of the array to be transferred is crucial.

2.3.6 T9000 Known Hardware Bugs and Problems

The experiences and results presented within this thesis are largely based upon the Gamma D02 revision of the T9000. There are four major problems with this revision affecting work within this thesis:

- The maximum stable clock speed is 20 MHz, some chips limited to 10 MHz, this compares to the design speed of 50 MHz.
- Double scheduling bug, causing a transmitting process to be rescheduled twice, can crash entire application
- Cache corruption bug, transmitted data is corrupted, crashes entire application
- The external memory interface is slow, causing up to a 50% degradation in performance compared to theoretical maximums

Further details of these problems (and other less serious bugs) and how I limited their effect are given in the relevant sections as they arise. General limitations related to the T9000 and software support are also presented as they arise.

The Gamma E03 revision has become available at the time of writing this thesis. The processor speed is still limited to 20 MHz, however the two main hardware bugs (cache corruption and double scheduling) have been removed. The external memory interface has also been significantly improved. For communications to internal memory or cache then the performance has remained identical. My initial tests suggest that the Gamma E03 can be used to build stable and reliable T9000 systems for arbitrarily complex traffic patterns and network configurations. The manufacturer also claims that the Gamma E03 will operate reliably at 25 MHz. I have yet to test this claim.

2.4 The C104 Packet Switch

The C104, see Figure 2.12, developed by SGS Thomson, is an asynchronous 32-way dynamic packet switch with DS links operating at 100 Mbits/s. It can interconnect up to 32 devices (for instance T9000s), and may be cascaded to form large switching networks. I have carried out tests running C104 links at 200 Mbits/s, investigations are under way into the reliability of running links at this speed.

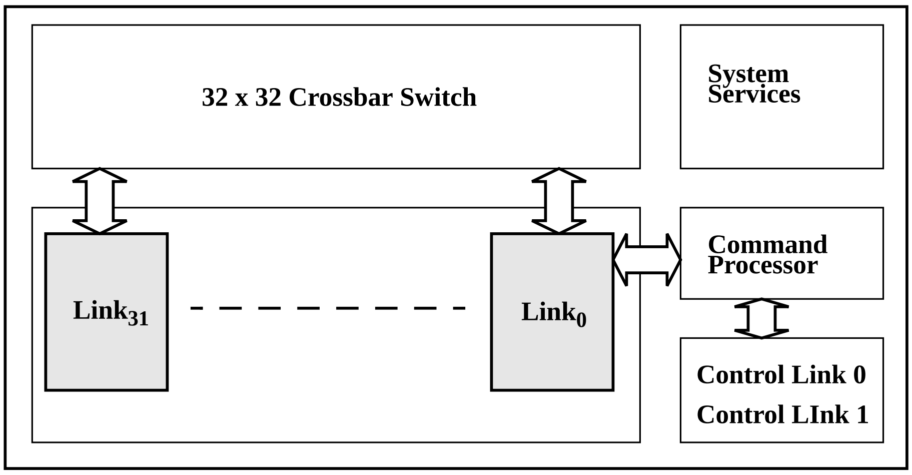


FIGURE 2.12 The C104 Packet Switch

The technique used to route packets through a C104, to select the required output link, is that of interval labelling. In this technique each link of a C104 is assigned a range of device labels (a device interval) which indicates the physical devices that are accessible via that link. Each physical device has a unique label associated to it. When a packet enters a C104, the device label contained in the packet header is compared to the device intervals. The output link whose device interval contains the required device label is selected to route the packet out of the C104.

The C104 uses wormhole routing (see Figure 2.13). A routing decision is made as soon as a packet header enters the C104. This routing decision leads to the creation of a temporary circuit through the C104 which vanishes as the packet terminator passes through. As a consequence of wormhole routing a single packet may pass through multiple C104s at any one time and the header may be received at the destination before the whole packet has been transmitted, thus minimizing the time required to transmit the packet through the network.

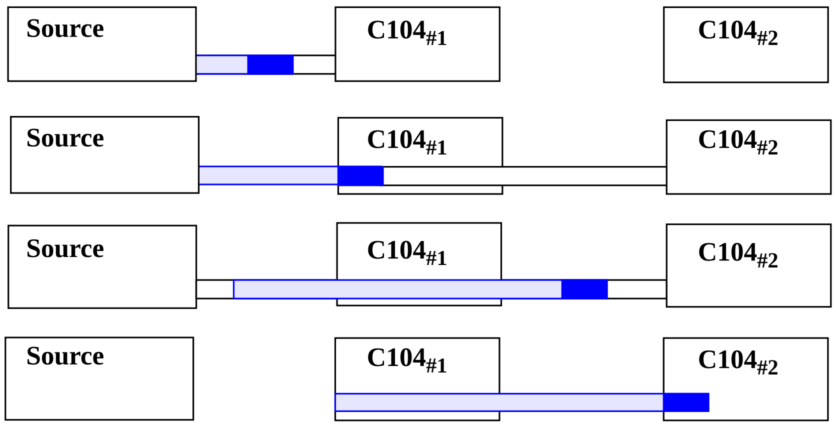


FIGURE 2.13 Wormhole Routing

In switching networks, there will often be many possible routes that a packet may take to reach a specified destination. Should one of these links be in use or in error then it is desirable that an alternative link be chosen. To fulfil this requirement the C104 supports grouped adaptive routing. Output links can be grouped so that packets routed to the first link of a group can be routed to the other links of that group should the first link not be available.

In addition to grouped adaptive routing the C104 supports universal routing. In this technique packets are first sent by a C104 to a random device (another C104). At this device, the packets are then routed to their final destination. The initial randomisation spreads the load across the network, reducing hot spots. Hot spots are localised bottlenecks in a network.

2.4.1 Hardware Bugs

There have been two major revisions of the C104 available during the work carried out for this thesis: revisions alpha and beta. There are no current plans for further revisions.

The alpha revision suffered from errors on the selection of interval separators, regarding whether the upper limit of the interval range was inclusive or not.The chip behaved differently from the hardware manual. This problem was solved for the beta revision.

On both alpha and beta revisions the links run at half the speed they are programmed to run at. This is solved by programming twice the speed required. The core speed of the C104 is also half the speed programmed, 15 MHz core speed is adequate for 100 Mbits/s link speed, so 30 MHz is programmed.

A link that is in error and part of a link group will still be selected to be used for output. The packets sent to this link will be discarded.

2.5 The T9000 Toolset Development System

Programs and applications for T9000 systems are developed on a host system using a T9000 toolset. For the work in this thesis that host was a Sun workstation. The toolset allows the user to compile code to be used for processes, define the interconnection of these processes and map these processes onto Transputer hardware.

Creation of executable code for a Transputer or Transputer network requires several stages involving the use of specific tools at each stage. The stages are:

- Compile and link source code for individual processes, creating a linked unit for each process.
- Produce a description of the hardware. A network initialisation file is produced from this description which can initialise the hardware (T9000s and C104s).
- Define the interconnection of processes by virtual channels, and map the processes and their interconnection onto the hardware.

Figure 2.14 relates these three steps to the Transputer model of concurrency presented in Section 2.1, “Transputers,”.

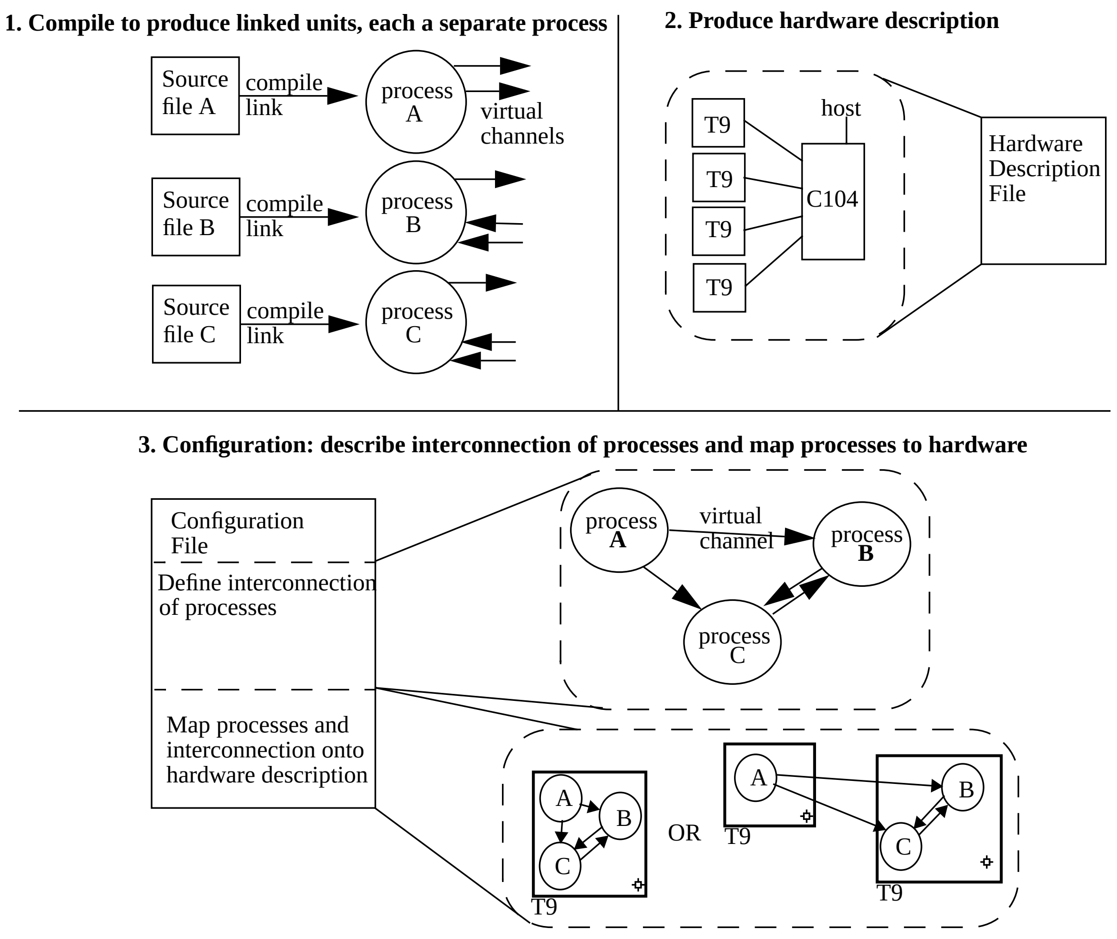

1. Compile to produce linked units, each a separate process
Source file A
compile link
process A
virtual channels
Source file B
compile link
process B
Source file C
compile link
process C
2. Produce hardware description
T9
T9
T9
T9
host
C104
Hardware Description File
3. Configuration: describe interconnection of processes and map processes to hardware
Configuration File
Define interconnection of processes
Map processes and interconnection onto hardware description
process A
virtual channel
process B
process C
A
B
C
T9
OR
A
T9
B
C
T9

FIGURE 2.14 Overview of toolset

I now describe the stages of development in more detail. Figure 2.15 shows the main phases in development and Figure 2.16 details the tools that are used to achieve them.

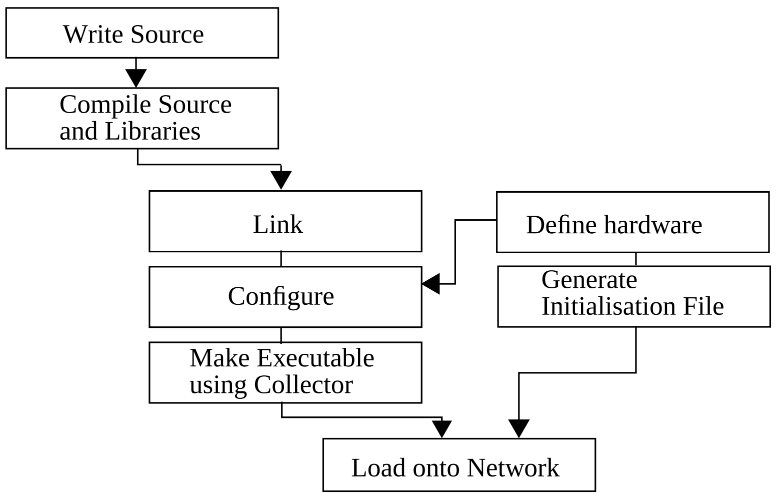

Write Source
Compile Source and Libraries
Link
Define hardware
Configure
Generate Initialisation File
Make Executable using Collector
Load onto Network

FIGURE 2.15 Main Toolset Development Stages

Source can be written in C or occam. Each source file is compiled using the appropriate language compiler (*oc* or *icc*) to produce one or more compiled object files. Commonly used object code can be combined into libraries using the librarian tool *ilibr.* The compiled object files and libraries are linked together using the linker *ilink.* This generates a single linked unit.

The configuration description is processed by the configuration tool *inconf* to produce a configuration data file.

The configuration description refers to a description of the hardware on which the code will run. The hardware description is written in NDL (Network Description Language).

T9000/C104 networks have a system of control links which are separate from the data network. The control network is used to configure the T9000s and C104s and in the case of the T9000 load the initial bootstrap code. The information required to perform this initialisation is held in the network initialisation file, which is generated by the initialisation file generator *inif* from the hardware description.

Before a program can be run it must be made executable. This involves adding bootstrap and loading information and is performed using the collector tool *icollect*. The configuration binary file generated by the configurer is read by *icollect* which generates a single executable file for a T9000 network, a btl file. A transputer network is initialised using the network initialisation file and the code is loaded onto the T9000s from the btl file

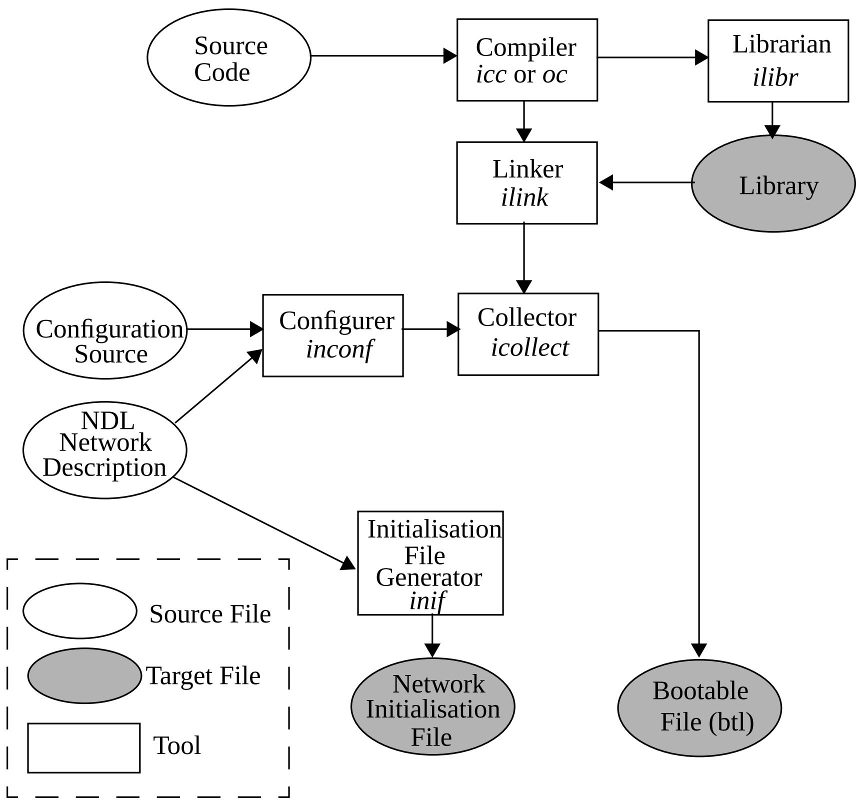


FIGURE 2.16 Key Programs used in Developing an Application

2.5.1 Problems with the Toolset

The general problems of the Toolset environment are presented within this section. More detailed specific problems that apply to a certain areas of work within the thesis will be discussed later as they arise.

The list of tools that the user is required to use is long and the dependencies are complex. Figure 2.16 only shows the most important tools that are required to develop a Transputer application, there are more tools that would further complicate the explanation given. For example, there are a separate tools for aiding the writing of network description files.

There are many important areas of software that are not covered by the toolset. A hardware 'spy' is an important area not covered by the toolset. If the user wishes to confirm that a certain network exists or list the devices present in a network then a 'spy' is used. Unfortunately, this program is not supported as part of the toolset and has been unusable in any work carried out for this thesis. To discover whether all devices were present in a network the only method was to write an application that used all the devices and try to configure and load it.

A correctly operating 'spy'[4] would provide the following essential functionality:

- Recover if an error occurs in the network (currently the spy does not)
- Validate if a network matches a given network description file (NDL)
- Produce an NDL file for a physical network
- Tests the data links of all devices

The C104 has many attributes which are contained in the network description file (NDL), including information on intervals and grouped adaptive routing. The toolset was supposed to provide an automated procedure for producing NDL files, and the attributes it contains. The bugs in the tools that carried out this process were so severe that I had to write all NDL files used within the work for this thesis.

2.6 Hosting T9000 Systems

A T805 Transputer [11] initializes and loads applications onto the T9000 network. The T805 is the previous generation of Transputer to the T9000, which uses four serial bi-directional OS (over sampled) links running at up to 20 Mbits/s. The T805 is connected to a SUN workstation via an Ethernet-to-OS link converter (B300). A single link of the T805 is used to configure the T9000 system using the control link via a C100 [3] protocol converter. The C100 is used to interface the DS link to an OS link, since the DS link is based on the exchange of packets and the OS link on the exchange of bytes. Another link of this T805 is used to load applications via a T9000 data link (see Figure 2.17). Access to file systems and host system services by the T9000s is provided by the SUN workstation via the T805.

After a network has been initialised the application must be loaded onto the network. The software used to perform this has not been presented, there are multiple versions and methods, which often vary from manufacturer to manufacturer. The main problem with the particular type used within the scope of this thesis was the complete absence of error recovery

4. A spy with the required functionality is now being produced at CERN as part of the ESPRIT project ARCHES [10].

and fault tolerance. Any error with the host link would result in the shutdown of the entire system. The bandwidth between the network and the host was also very low (20 to 30 Kbytes/s), which became a problem when large amounts of code and data were required. All available host systems lack the following essential functionality[5]:

- Report errors in a way such that the user can readily identify the physical device that has suffered an error. Currently the relation of logical to physical device numbers is very difficult.
- Attempt to recover the error, instead of immediately exiting

Bandwidths to host should be at least an order of magnitude higher

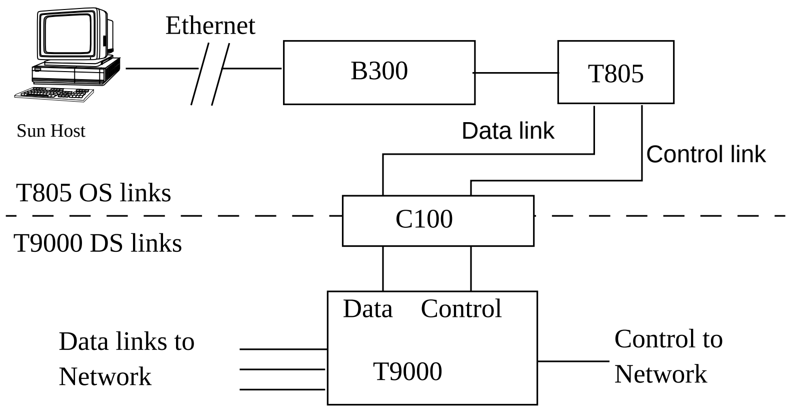


FIGURE 2.17 Hosting T9000 Systems

2.7 The GPMIMD Machine

The GPMIMD machine has been designed as a switching fabric of 1000 DS links interconnecting up to 256 T9000 processors. In its present configuration 56 C104s provide full interconnectivity between 64 T9000s. The T9000s are mounted on small individual plug-in boards based on the SGS Thomson HTRAM (High performance TRAnsputer Modules) standard. A single size 4 HTRAM holding a T9000 and 16 Mbytes of memory is approximately 10 cm by 10 cm, and plugs into matching connectors on the motherboards.

The present configuration contains a total of 8 motherboards (see Figure 2.18) each carrying 8 T9000s and 5 C104s. In addition, 4 switch cards, each carrying 4 C104s, provide the inter-motherboard connectivity (see Figure 2.19). Four independent folded Clos networks [12] have been implemented to efficiently use the four DS links connected to each of the T9000s.

One motherboard C104 is used for connection to external devices and as a control link fan-out. The other 4 C104s provide the 4 networks that connect to all T9000s, there are also links from each of these 4 C104s to the C104 with external connections.

5. Host systems are now being developed at CERN with the required functionality as part of the ESPRIT project ARCHES.

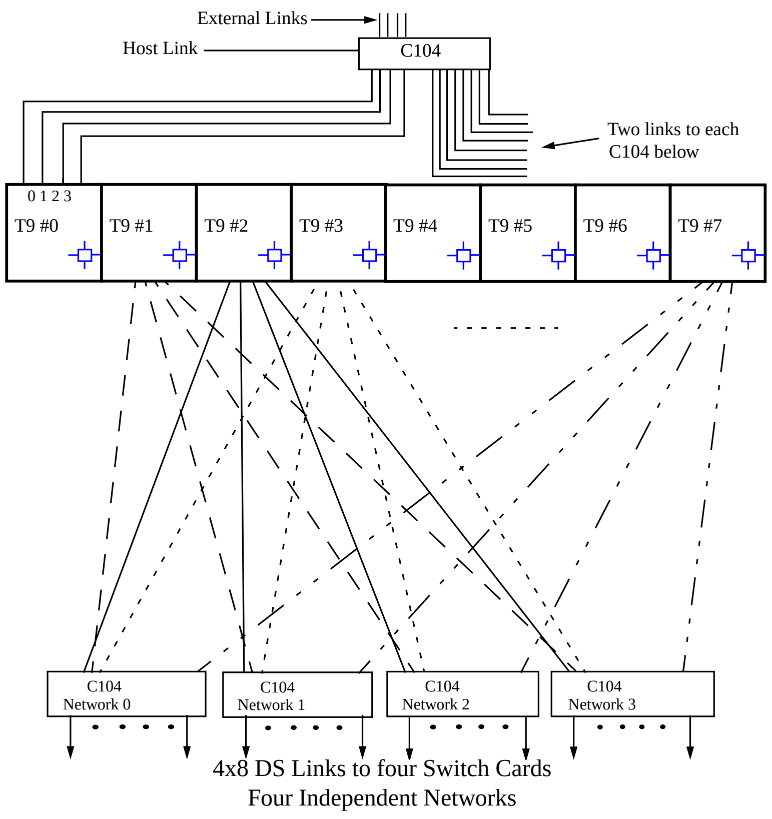


FIGURE 2.18 A GPMIMD Machine Motherboard

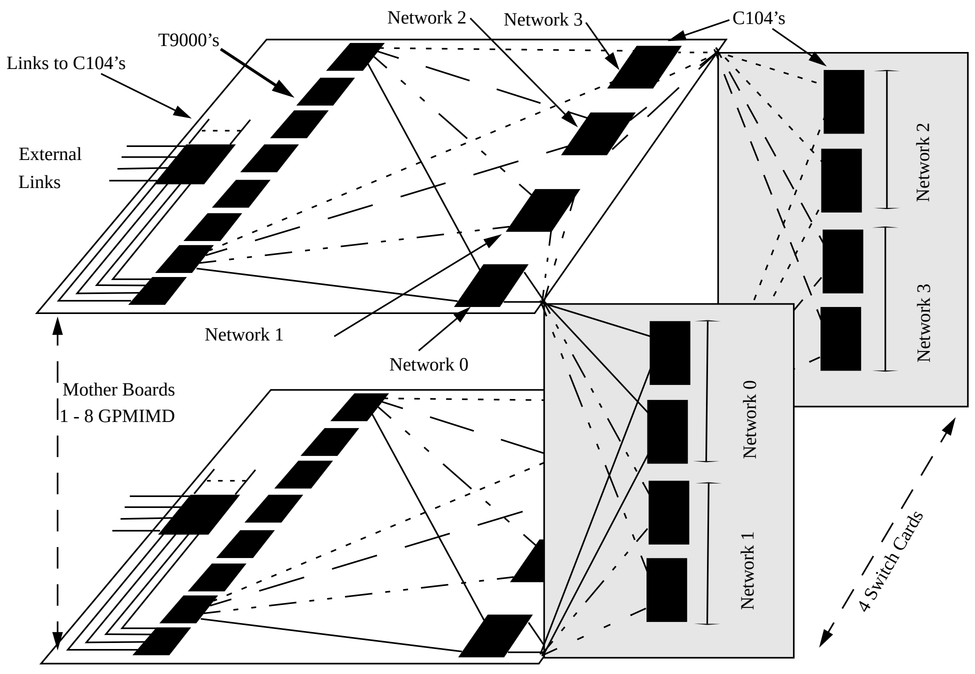


FIGURE 2.19 The GPMIMD Machine: 8 horizontal motherboards and 4 vertical switch cards

The topology of the Clos network for a 48 node, 6 motherboard machine with 2 switch cards is shown in Figure 2.20. This topology has been presented because all communication benchmarks and other communications measurements presented within this thesis were carried out on this topology. Each C104 on the left and right represent a single C104 on a motherboard and the switch card C104s are those in the centre. The 4 links between each motherboard and switch card C104 may be grouped into ‘bundles’ of 1 to 4 links. These links are performing grouped adaptive routing, as presented in Section 2.4, “The C104 Packet Switch,”. A single network may consist of 3 to 12 links between the motherboards and the switch cards from one half of the machine to the other.

A message between T9000s on the same motherboard passes through one C104, a message between T9000s on different motherboards passes through three C104s. The layer of C104s connected to the T9000s is called the terminal stage of the Clos; the other stage is called the centre stage.

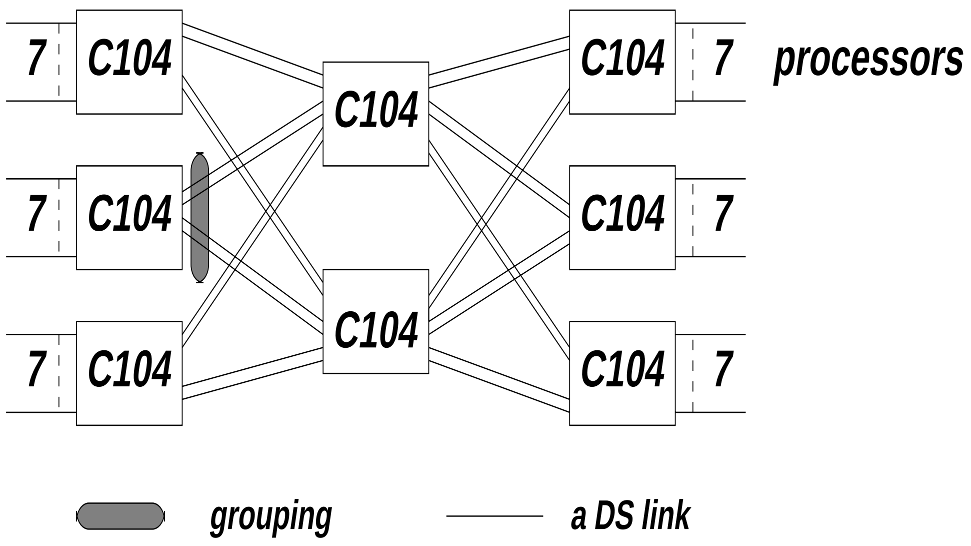


FIGURE 2.20 **A Single (of 4) GPMIMD Machine Clos Network**

2.8 The TransAlpha Module

The low clock speed of the T9000 (among other factors) has resulted in poor computational performance compared to its communication abilities. The resulting imbalance is being addressed by the TransAlpha module. The performance of the T9000 for communications, context switching and interrupt response is not matched by the low computational performance of a 20 MHz T9000 processor.

A high performance computing node has been designed using a T9000 as a communications controller for a DEC Alpha 21066A microprocessor [13]. This hybrid node is called the TransAlpha module, and is designed to combine the complementary strengths of the two processors. The module is being developed by Parsys in collaboration with my group at CERN.

Figure 2.21 shows the layout of the TransAlpha module, which connects the T9000 and Alpha via a 32-bit PCI standard bus. The module conforms to the SGS Thomson HTRAM standard and can therefore fit directly into existing T9000 systems. The TransAlpha module is a size 8 HTRAM, double the size of the current T9000 HTRAMs used in the GPMIMD machine. All components fit into the single HTRAM with dimensions of approximately 20 cm by 10 cm.

The hardware allows the direct replacement of two GPMIMD machine T9000 HTRAMS with a single TransAlpha module. The T9000 HTRAMs can be removed, and the TransAlphas plugged in to replace them. The result is that a single motherboard which now contains 8 T9000s can have them replaced with 4 TransAlpha modules. The full capacity of the present machine if all T9000s were replaced would then be 32 TransAlphas.

The module comprises of four parts:

- A 233 MHz 21066A DEC Alpha microprocessor with 512 Kbyte external cache and 32 or 64 Mbyte of 64-bit DRAM connected to its local bus.
- A 20 MHz T9000 with 512Kbyte of 32-bit SRAM and 128Kbyte of flash ROM.
- Bus bridge between T9000 local bus and PCI bus
- FPGA logic attached to the T9000 local bus used to adapt the T9000 bus to the PCI 9060 and to initialise and control the Alpha processor and the PCI 9060.

The occam programming environment is not available on these boards. The only option is the C programming environment, using extensions to allow the Alpha access to the point to point communication channels. The CSP model on which occam relies is still implemented, with some restrictions.

The communications performance of the TransAlpha relies heavily on the performance of the T9000 external memory interface. Results produced using the revision D02 T9000 have suffered from the limitations of the external memory interface. Work is currently ongoing to produce TransAlphas with revision E03 T9000s that will improve performance.

2.9 Conclusions

The technology used within the thesis have been presented. The T9000 and C104 have been shown to support functionality which is crucial to performance in multi-processor systems: concurrent communication and computation, fast interrupt response and context switching times. This performance is quantified in the following chapter.

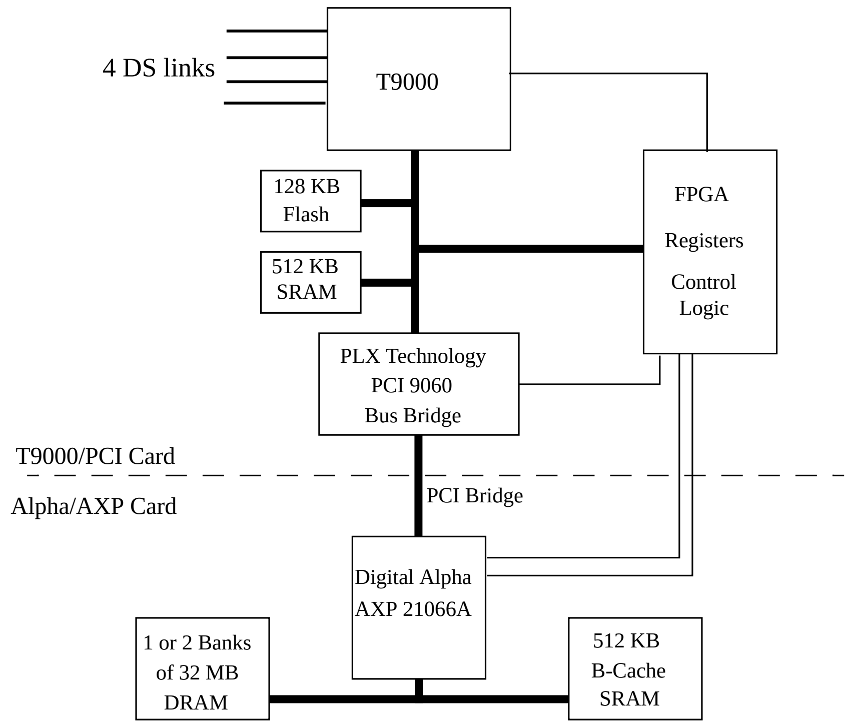


FIGURE 2.21 The TransAlpha Board Layout

Chapter 3
Evaluation of technology

In this chapter I present an evaluation of the technologies used within this thesis. Communication and computation benchmarks are presented along with context switching and interrupt response performance.

A comparison is then made to other platforms, including the PowerPC. I will show that in the areas crucial to multiprocessing the T9000 compares favourable to other less integrated technologies.

I have identified a set of critical factors which affect the performance of multi-processor systems, I evaluate these parameters for the T9000 and C104 within this chapter. These parameters are summarised at the end of the chapter in Table 3.6, along with their general relevance to multi-processing. In chapter 5, I use the performance of the T9000 and C104 compared to the requirements of the LHC to quantify the importance of each parameter.

General communication performance and networking results use a 48 node revision D02 T9000 machine using alpha C104s. Context switching performance and interrupt response related benchmarks only requiring small numbers of T9000s use revision E03 T9000s and beta C104s.

3.1 Communication Benchmarks

3.1.1 Single Link Results

The time between the sending of a short packet (one word) and the reception of an acknowledge packet between two directly connected 20 MHz revision D02 T9000s was measured to be 7.5 µsecs. The additional delay incurred by connecting two T9000s via a C104 was measured to be 1 µsec (Figure 3.1 shows the difference in elapsed time for a message between two directly connected T9000s and T9000s connected via a C104). The curve intercepts the y-axis at 2 µsec which represents the extra delay in transmitting two packets: an acknowledge and data packet. For a single message the data packet itself will suffer a delay at the C104 and also its corresponding acknowledge packet, hence the intercept is at 2 µsec, not 1 µsec. Therefore the C104 requires 1 µsec to switch a packet.

In Figure 3.2 the dependency of the bandwidth between two directly connected 20 MHz revision D02 T9000s on the number of virtual links used is presented. The figure shows the bandwidth as a function of message size for one to five virtual links mapped onto a single physical link. The bandwidth represents the usable amount of data exchanged between two T9000s running at 20 MHz. The discontinuity in bandwidth at each 32 byte boundary is due to the packetisation performed by the VCP. A 32 byte message requires a single packet to be sent and acknowledged, whereas a 33 byte message requires two packets to be sent and acknowledged.

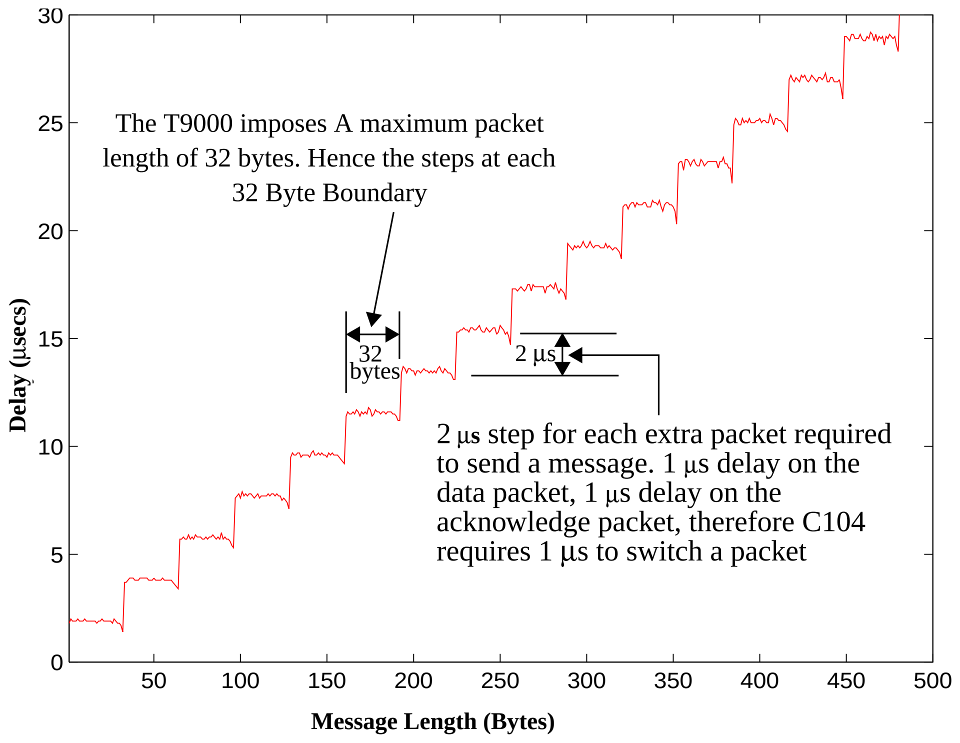


FIGURE 3.1 **Latency due to C104. 20 MHz D02 T9000s, 100 Mbits/s links, 16 K internal memory**

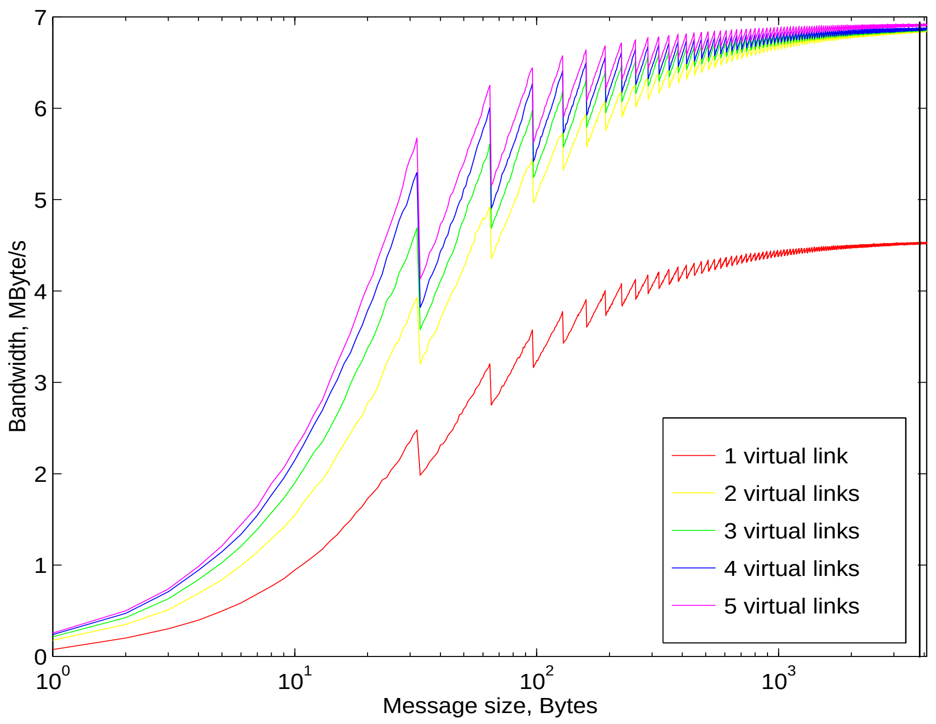


FIGURE 3.2 **Single link bandwidth, uni-directional. 20 MHz D02 T9000s, 100 Mbits/s links, 64 bit interface, 8K cache 8K internal memory, directly connected links**

In Figure 3.2 the increase in bandwidth for more virtual links can be accounted for by the increased packet inter-leaving performed by the VCP and more efficient use of its pipelined architecture. When multiple virtual links are used, packets for different virtual links may be transmitted independently of the reception of acknowledge packets on other virtual links. If a single virtual link is in use, a packet is sent, then the virtual link (and physical link) is idle until the acknowledge is received, which reduces the bandwidth. If another virtual link is in use, the first virtual link (waiting for an acknowledge) stays idle but the second can use the link. If enough virtual links are used then the bandwidth limit will be that of the VCP, or the limit of the memory to supply the VCP with data. A constant message start up time accounts for the reduced bandwidths for smaller packets.

The results give a maximum for the achievable performance of a 20 MHz T9000 processor. Conditions have been chosen such that all data is read from the cache memory and therefore the external memory interface of the T9000 does not have to be used.

The theoretical maximum of a DS link running at 100 Mbits/s using single byte headers is 9.55 Mbytes/s [14]. The measured limit is only 7.0 Mbytes/s, this is not a problem of memory bandwidth, all data resides in the cache. The problem is the limit of the VCP at 20 MHz, it is the only possible bottleneck other than the memory interface. The problem can be removed by increasing the clock speed of the T9000, and hence the speed at which the VCP operates. The VCP could fully saturate the links if the T9000 was running at 30 MHz. The T9000 was designed to run at 50 MHz, with the VCP able to saturate the links at 30 MHz.

3.1.2 Dependence on memory bandwidth

If Figure 3.3 is compared to Figure 3.4 the effect of reading data from external memory (as compared to internal memory) can be seen. The two figures show the achieved bandwidth out of the T9000 onto 1 to 4 physical links. In Figure 3.3 external memory is used, in Figure 3.4 data is always read from internal memory.

In Figure 3.3 when longer messages and more than two physical links are used data is accessed from external memory, only 8K of cache is available. At this point there is a clear reduction in performance. The limitation is due to problems in the external memory interface, the PMI, the external memory interface can only write data to all four links at 16.4 Mbytes/s. This figure agrees with the value in Table 2.3 on page 18 for external to internal RAM using a 64 bit interface. This agreement confirms the limitations presented in Section 2.3.5.2, “Memory to memory transfer rates,”, i.e. 3 processor cycles required to transfer data between the PMI and the link or internal memory. This results in the limit of 16.4 Mbytes/s and not the expected performance of 32 to 40 Mbytes/s (also presented in Table 2.3).

In Figure 3.4 data is written to the links from internal memory, the limit is 28 Mbytes/s. The bottleneck is the VCP which can only drive the four links at 28 Mbytes/s. The limit cannot be internal memory transfers, which have been benchmarked at 66.4 Mbytes/s, see Table 2.3.

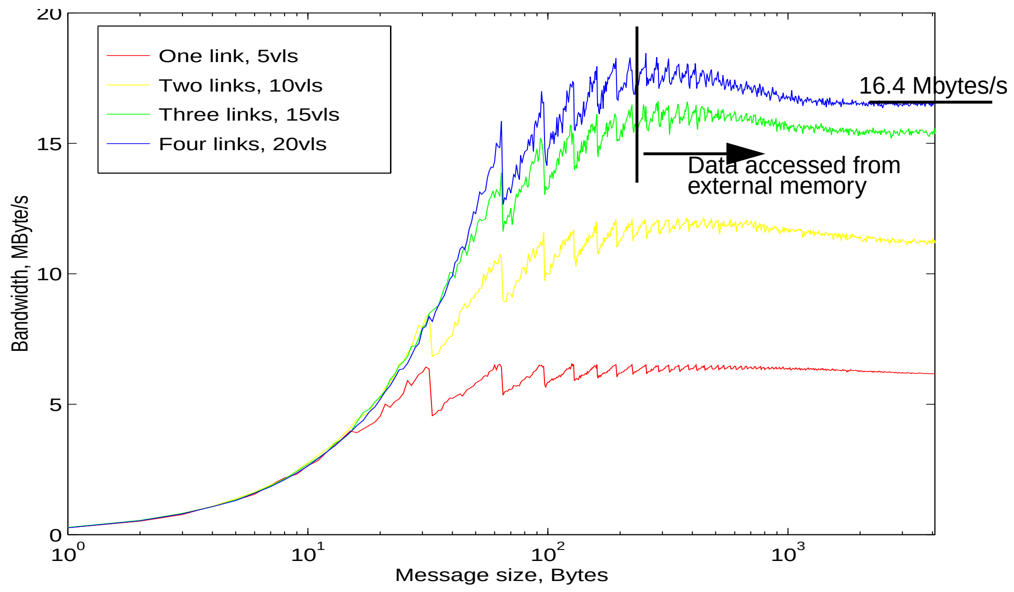


FIGURE 3.3 **Multiple links bandwidth, 20 MHz D02 T9000s, 64 bit interface, 8K cache 8K internal memory, 100 Mbits/s links. Connections via C104s. Performance limit is the PMI.**

3.1.3 One to four physical links

Figure 3.4 demonstrates that the achieved bandwidth scales linearly with the number of physical links used for uni-directional traffic. The rate at which the VCP can drive the links scales linearly, however, the performance (28 Mbytes/s) is still well below the theoretical limit of four DS links: 38.2 Mbytes/s (4 * 9.55). These results use 16 K internal memory, there is no access to the external memory.

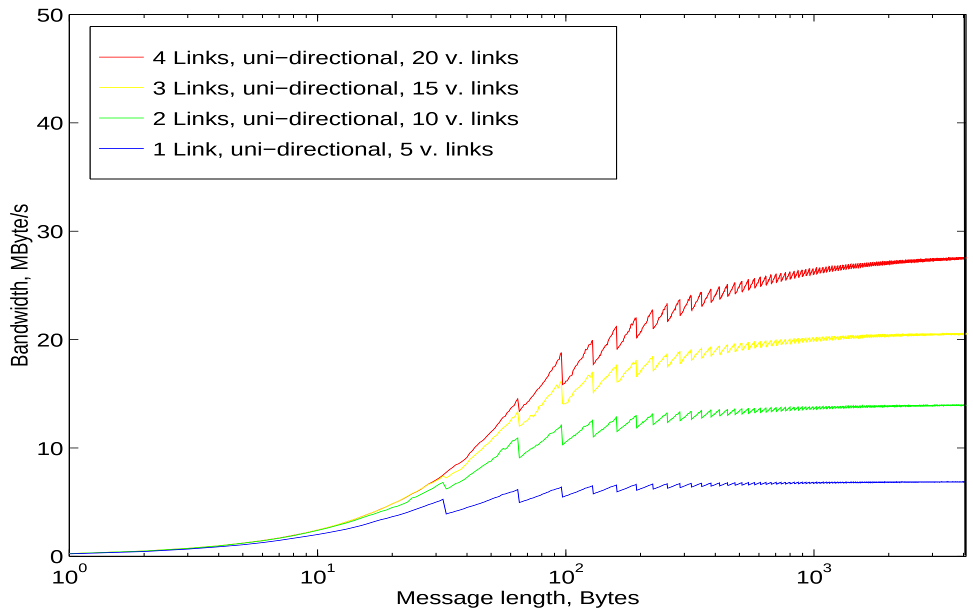


FIGURE 3.4 **Multiple links, 20 MHz D02 T9000s, all data in internal memory, 100 Mbits/s links. Connections via C104s. Performance limit is the VCP.**

3.1.4 Dependence on clock speed

The measured bandwidths as a function of message size when using 20 and 25 MHz processors are shown in Figure 3.4 (20 MHz) and Figure 3.5 (25 MHz). These two plots are taken from Low Level Benchmarking of the T9000 Transputer [15]. In these measurements the links are routed through different C104s, i.e. the curve for four links uses four independent C104s. This configuration is dictated by the architecture of the GPMIMD motherboards. This was the only architecture where 25 MHz T9000s were available[6]. The theoretical limit for the single link bandwidth is now 9.26 Mbytes/s [14] on each physical link. This is lower than the previous 9.55 Mbytes/s due to the extra packet headers required to route through the C104s. Two byte headers are now required instead of one. The bandwidths measured at 20 and 25 MHz fall short of the 9.26 Mbytes/s, but there is a clear improvement from the 20 MHz to 25 MHz processors. This demonstrates the improvement of increasing the speed of the VCP, however, it is still unable to exploit fully the capacity of the links. If the T9000 were running at 30 MHz the VCP should be able to reach the theoretical limits for the link bandwidth, currently reliable 25 Mhz T9000s are not available in quantity.

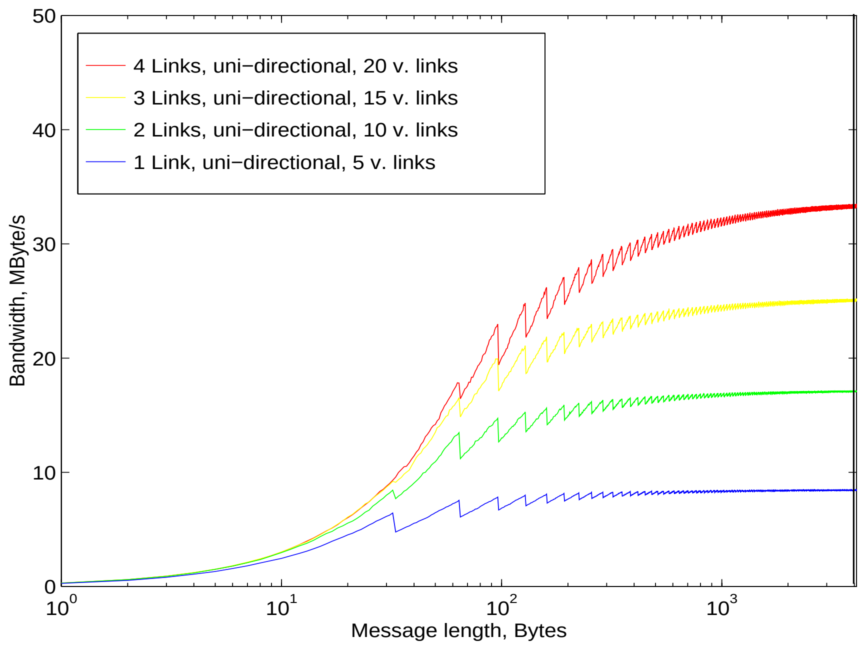


FIGURE 3.5 Multiple links, 25 MHz D02 T9000s, all data from internal memory, 100 Mbits/s links. Connections via C104s.

3.1.5 Single Network GPMIMD Machine Results

The results presented are based on a 48 revision D02 T9000 machine with 6 motherboards and two switch cards, i.e. the Clos presented in Figure 2.20 on page 27. This was not a complete configuration as there were only two instead of four switch cards available. The result is a maximum of 12 links between the terminal and centre stages of the Clos between one

6. The T9000s (REV D) could run communication benchmarks at 25 Mhz but were unstable running many other test codes. The floating point unit is likely to fail at 25 MHz.

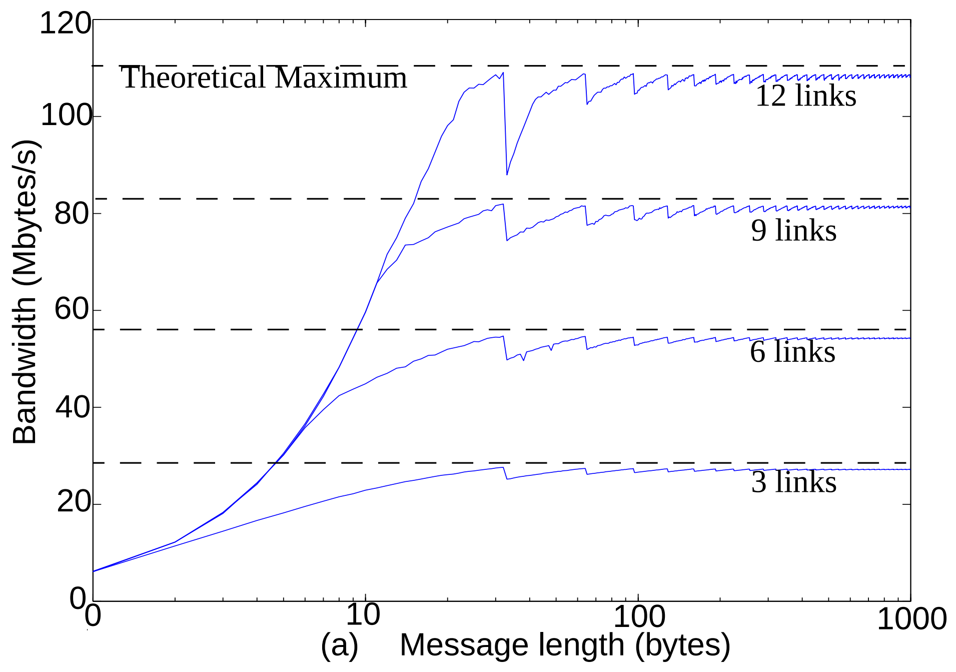
half of the machine and the other for a single network. The 6 slot 0 T9000s have been omitted, they are not directly part of the Clos (see Figure 2.18 on page 26) and would make the interpretation of results very difficult.

The T9000s were divided into 21 sources and 21 destinations and the average uni-directional cross-sectional bandwidth of a single Clos network (out of the possible four) was measured for two different traffic patterns. These two traffic patterns give information on the general communications performance. The T9000s were only used as data producers and consumers. A source communicates a message to a destination using 5 virtual links.

In the first of these traffic patterns, sources and destinations are formed into fixed pairs; no two sources send data to the same destination. In this situation, the maximum uni-directional bandwidth allowed by the technology, network and routing algorithm is measured. The achieved uni-directional bandwidth as a function of message size and the extent of grouped adaptive routing (3,6,9 or 12 links between the terminal and centre stages of the Clos network) is shown in Figure 3.6a. The theoretical maximum for 12 DS Links is 111 Mbytes/s (12 * 9.26 Mbytes/s). The measured value is in good agreement with this theoretical value to within 3%. The switch card links are the bottleneck (2 switch cards available instead of 4), there is a funnelling effect that allows the 20 MHz T9000s to saturate the links.

In addition, the achieved uni-directional bandwidth scales linearly with the amount of grouped adaptive routing in use (3,6,9 or 12 links). This shows there are negligible overheads due to its implementation. These results show that with no contention on the destination links and the maximum amount of grouped adaptive routing, each network has a uni-directional bandwidth of 108 Mbytes/s. This is an important measurement for future comparison, it provides the maximum possible throughput of a single network uni-directional.

In the second traffic pattern, messages are sent from each of the 21 sources to the 21 destinations randomly. The measured uni-directional bandwidth as a function of message size and grouped adaptive routing is shown in Figure 3.6b. A significant degradation in the uni-directional bandwidth compared to the paired traffic pattern can be seen. For 6 and 12 grouped links only 61% and 38% of the uni-directional theoretical bandwidth is achieved. Although the bandwidth achieved with 12 links is greater than with 6 links, there is a lower utilisation of the available bandwidth. This indicates that under this traffic pattern, contention on the final destination links limits the performance. It also suggests that there would have been little improvement in bandwidth if all four of the switch cards had been available, the switch cards were not the bottleneck.



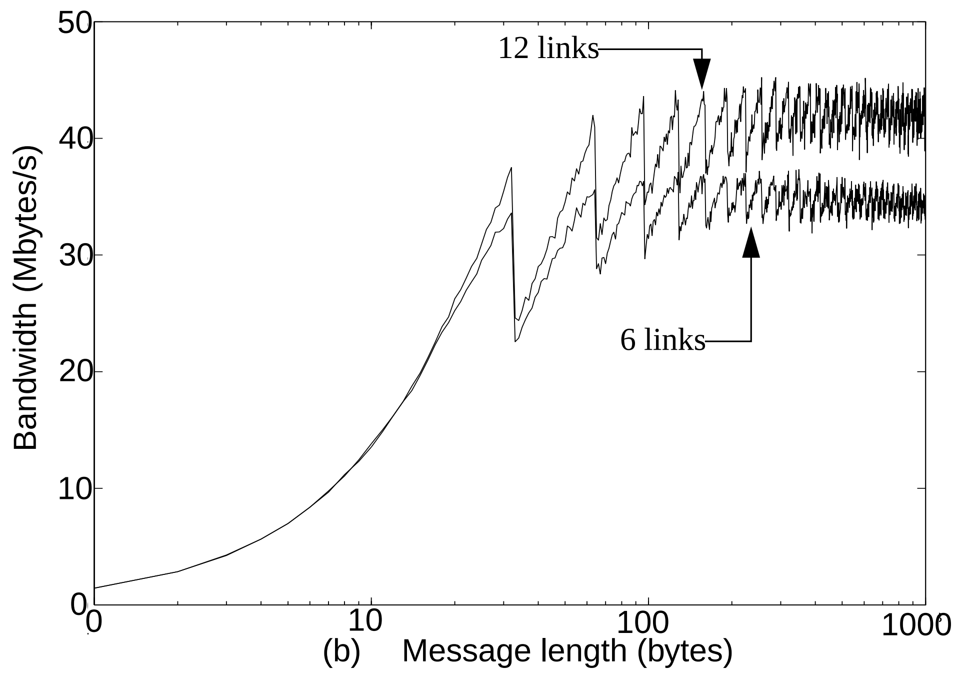


FIGURE 3.6 **Uni-Directional Bandwidth for a single GPMIMD network (a): fixed (b): Random Sources and Destination Pairing. 20 MHz D02 T9000s, 64 bit interface, 8K cache 8K internal memory, 100 Mbits/s links**

3.1.6 Multiple Network Results

To obtain the maximum cross-sectional bandwidth of the whole machine measurements were made using all four Clos networks in parallel. Under these conditions the 20 MHz revision D T9000s were not stable, however, sufficient running time was achieved to allow measurements of the fixed pair traffic patterns. The measurements were made for both uni-directional and bi-directional traffic.

3.1.6.1 Figure 3.7: Four networks, uni-directional fixed pairs

There are 21 sources and 21 destinations which corresponds to 84 source and 84 destination links. Figure 3.7 shows the results for uni-directional bandwidth for fixed pairing of 84 sources and destinations. There is a clear bottleneck at 379 Mbytes/s. This bottleneck is due to the switch cards. For each network I have shown that a maximum of 12 links are available to the switch cards, this is multiplied by 4 when 4 networks are in use. Unfortunately, at the time these benchmarks were carried out there were 6 links removed in error. The result was that 42 links were available to the switch cards. The theoretical maximum of a DS link under these conditions is 9.26 Mbytes/s, so 42 links gives a maximum of 389 Mbytes/s. The measured saturation value (379 MBytes/s) is very close to the theoretical switch card maximum (389 Mbytes/s). Figure 3.4 shows that a single 20 MHz T9000 can provide 28 Mbytes/s uni-directional, therefore, if there were no other bottlenecks 21 T9000s would provide 588 Mbytes/s, as opposed to the observed limit of 379 Mbytes/s.

3.1.6.2 Figure 3.8: Four networks, bi-directional fixed pairs

Figure 3.8 shows the equivalent result but using bi-directional traffic. Again there are 42 links to the switch cards. Under these conditions a DS link has a bi-directional theoretical maximum of 16.54 Mbytes/s [14], 42 links would allow 695 Mbytes/s. Clearly the limit of the switch cards is not reached. Driving the links bi-directionally has caused the VCP of the T9000 to become the bottleneck. In addition, when the message size becomes large enough to force the T9000 to read from external memory, the external memory interface becomes a second bottleneck. This second bottleneck is seen as a sharp fall off at approximately 600 byte messages for 2 virtual links per physical link and approximately 250 byte messages for 3 virtual links per physical link. The message size required to see this performance reduction reduces when more virtual links are in use. Distinct data is sent on each virtual link, hence more memory is required for larger numbers of virtual links. The switch cards are not the bottleneck, hence using four instead of two switch cards would not have increased the bi-directional performance.

The achieved bi-directional bandwidths correspond to 32 Mbytes/s for each T9000. This figure is in good agreement with ‘Low Level Benchmarking of the T9000 Transputer’ which measures a limit of 33 Mbytes/s for 4096 byte messages.

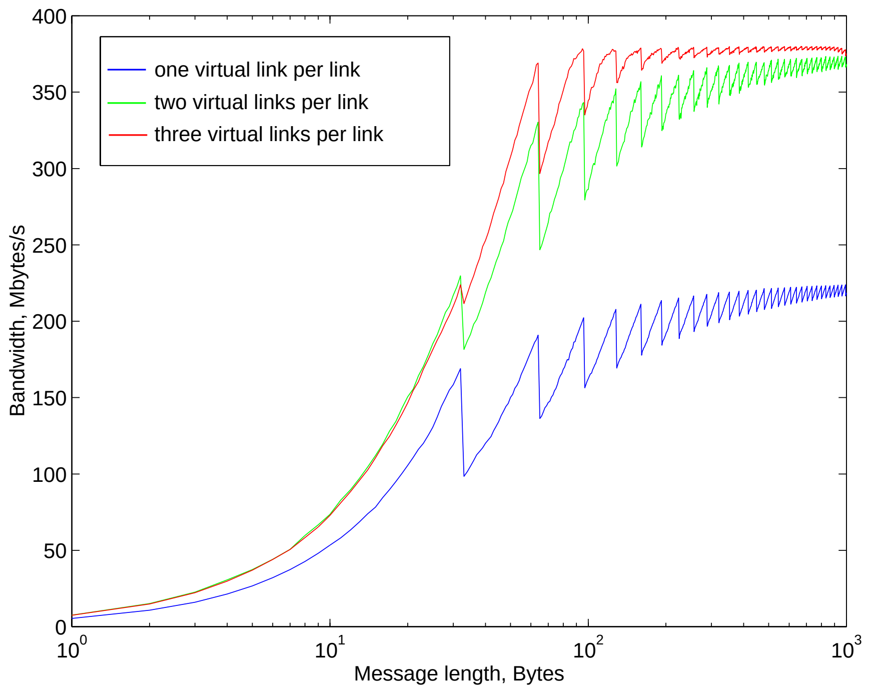


FIGURE 3.7 **Four Networks, Uni-directional Fixed Pairs. 20 MHz D02 T9000s, 64 bit interface, 8K cache 8K internal memory, 100 Mbits/s links**

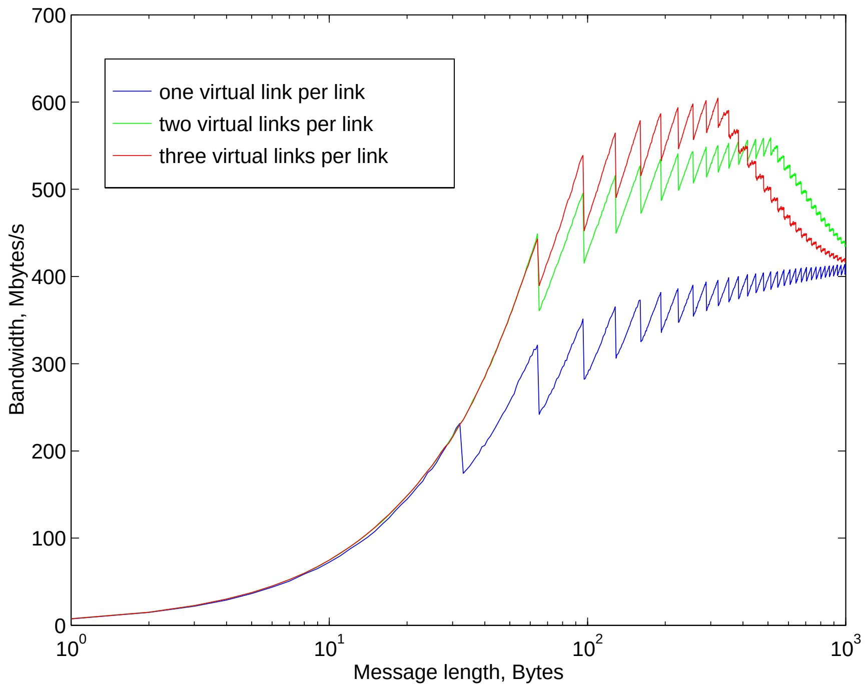


FIGURE 3.8 **Four Networks, Bi-directional Fixed Pairs. 20 MHz D02 T9000s, 64 bit interface, 8K cache 8K internal memory, 100 Mbits/s**

3.2 Interrupt Response and Context Switch Results

The results in Table 3.1 use a revision E03 T9000 with a 64 bit memory interface, 8 Kbytes cache and 8 Kbytes internal memory. The measurements are made by performing the operation to be timed over many iterations, then dividing the total elapsed time by the number of iterations. The total time is calculated using the Transputer's 1 microsecond clock, the accuracy is improved further by using a large number of iterations.

3.2.1 Timeslice operation

The timeslice operation is measured by timing the RESCHEDULE occam function. This function reproduces a timeslice operation, by sending the current process to the back of the active process queue. As expected the timeslice operation (requiring a partial context switch) is faster than the timer generated interrupt response time (requiring a full context switch). Details of the different types of context switch are presented in Section 2.3.4.4, "Full and partial context switch," on page 15. The difference of 0.5 microseconds is the time required to store the process state.

3.2.2 Timer generated interrupts

The timer generated interrupt originates from the internal Transputer clock. A high priority process is waiting for a certain processor time. When that time is reached, it becomes active and immediately pre-empts the current low priority process, requiring a full context switch, after which the high priority process will start executing. The interrupt response time is the time between the high priority process becoming active and actually starting to execute, i.e. the time required for a full context switch. The time required to return from this high priority process to the low priority is also measured. The time for the interrupt response and the return from the interrupt are very similar since the operations are essentially the same.

3.2.3 Interprocess communications

The interprocess communication time is the time for one process to write a single 32 bit word to another process on the same T9000. Communication between processes on the same T9000s are handled by the CPU, the VCP is not used. To communicate a single word from one process to another requires two context switches, one for the sender to be scheduled, and another for the receiver to be scheduled. Therefore the time for an inter-process communication should be composed of the time for two context switches and any additional time required to transfer the data. For each benchmark Table 3.1 details the component actions required for a single iteration of the benchmark.

The type of context switch required will depend on the priority of the two processes. An I/O instruction does not require any state to be stored, therefore only partial context switches should be required. However, when one high priority process and one low priority process are used the low priority process will always be pre-empted by the high priority process, i.e. a full context switch will always be required. This is because the low priority process is the only low priority process present, therefore it will not be descheduled when it initiates I/O, it will wait until the high priority process becomes active and pre-empts it.

When the priorities of the two processes are the same neither process will be able to pre-empt the other, i.e. no full context switch will occur. The communication instructions only require partial context switches. The result is that inter-process communication is faster between processes of the same priority than between mixed high and low priorities.

TABLE 3.1 Interrupt and Context Switch Results.

Benchmark	Time (microseconds)	Breakdown of component actions
Timeslice operation. high to high or low to low priority	1.4	single partial context switch
Timer generated interrupt (interrupt response time)	1.9	single full context switch
Return from interrupt operation	1.9	single full context switch
Inter-process communication high to high priority	3.2	two partial context switches and data transfer
Inter-process communication high to low or low to high priority	4.0	two full context switches and data transfer
Inter-process communication low to low priority	3.2	two partial context switches and data transfer

3.3 Supervisor/Control processor benchmarks

In multiprocessor systems there will be requirements for centralised supervisor/control processors. In particular, I will present requirements for centralised supervisor/control processors in all HEP data acquisition systems presented within this thesis. The general requirements of this processor will be the control and monitoring of a large number of distributed processors at high rates. The traffic will generally consist of short control or status messages. Two benchmarks have been created to assess crucial performance aspects for these supervisor/ control processors:

- Short message sends, i.e. the ability to send short messages at high rates.
- The CPU load of link communications, i.e. the ability to perform computation (make supervisor/control decisions) while communication is taking place.

The risk of low supervisor/control rates is that the entire network (communications network and processors) performance will be limited by the supervisor/controller. In distributed systems, bottlenecks will occur at centralised intelligences or communication points that must communicate with all or most of the components in a system. In these cases the supervisor/ controller may have to be replaced by an alternative technology: a dedicated controller in hardware and independent links, adding cost and complexity.

The benchmarks depend heavily on the performance of the VCP, interrupt response time and context switching. This allows the benchmarks to be used as a comparison to other platforms. Both run on two 20 MHz REV E T9000s on the same GPMIMD machine motherboard, i.e. a single C104 switch between them. Small messages and occam code are used which stay within the cache, and all links run at 100 Mbits/s.

3.3.1 Short Message Sends

The benchmark is a measure of the number of short messages that a single T9000 can produce per second. Between one and ten high priority processes run concurrently on a single source and a single destination, each process using a different virtual link. The source sends in parallel on all virtual links, continually switching between the processes, sending as many short messages as possible. On the destination there are one to ten processes all waiting for input. When a packet arrives at the destination the corresponding process for that virtual link will be rescheduled. The following parameters will be varied: message length, number of virtual links and number of physical links. The source and destination processes were run at both low and high priorities, it made no difference to the results. This benchmark is of interest for two reasons:

- It allows comparison between different platforms for efficient context switching during communications. When short messages are used the latencies of context switching are more dominant than when larger messages are in use. Ten processes continuously context switching on a single processor every time they send a single packet (message).
- It measures the capability of the processor to perform as a centralised supervisor or control processor, where many short messages must be sent to large numbers of possible destinations.

Figure 3.9 shows the number of messages sent per second for variable message lengths using a single physical link.

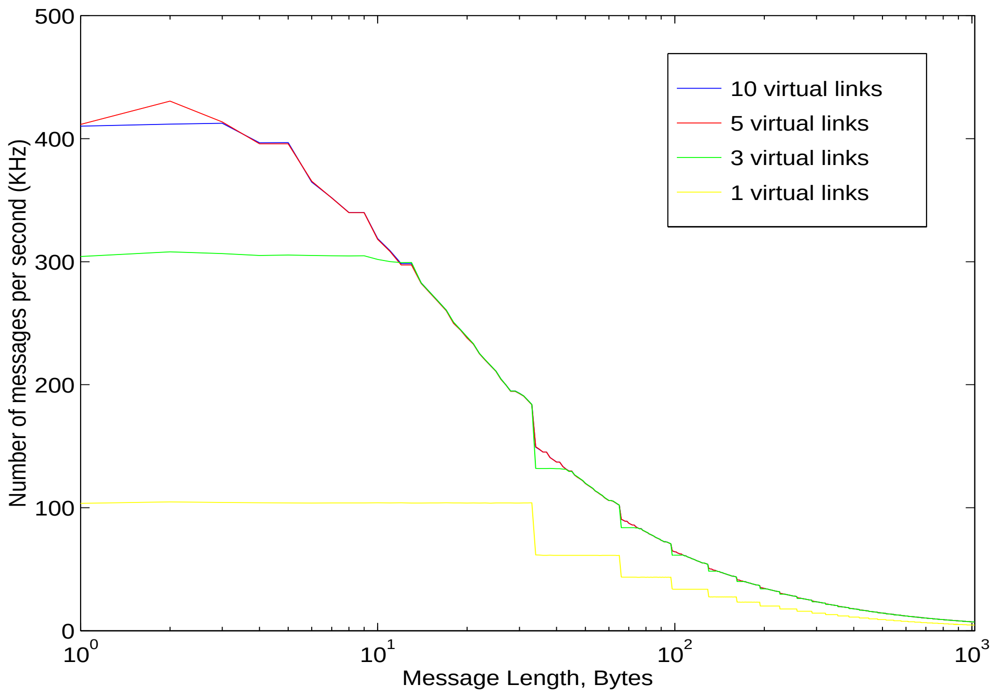


FIGURE 3.9 Short message sends, vary number of virtual links, between two 20 MHz E03 T9000s connected by C104s, 64 bit interface 8 K cache 8 K internal memory. 100 Mbits/s links. Single physical link in use.

There are four lines which represent the number of virtual links in use. The maximum performance is reached for 5 virtual links. Again the increase in performance is due to the increased interleaving of packets by the VCP. In addition, the packet boundaries (every 32 bytes) cause sudden drops in performance, the reduction is more severe when less virtual links are in use. The causes are identical to the effects presented in section Section 3.1.1, "Single Link Results," on page 31. For 4 byte messages the maximum rate is 415 KHz which corresponds to 1.66 Mbytes/s. For 64 byte messages the maximum rate is 106 KHz which corresponds to 6.8 Mbytes/s. These numbers are in agreement with results from Section 3.1.1, "Single Link Results," which show a maximum bandwidth of a single link to be 7.0 Mbytes/s, achieved for larger messages.

Figure 3.10 shows the performance gained when using up to four physical links. The maximum rate at which messages can be sent only increases to 430 KHz. The bottleneck is not in the links, there is a limit at 430 KHz for 4 to 32 byte messages, adding links or shortening the messages will not increase performance. This limit is not proportional to the message length. There are a constant 10 virtual links in use on each physical link.

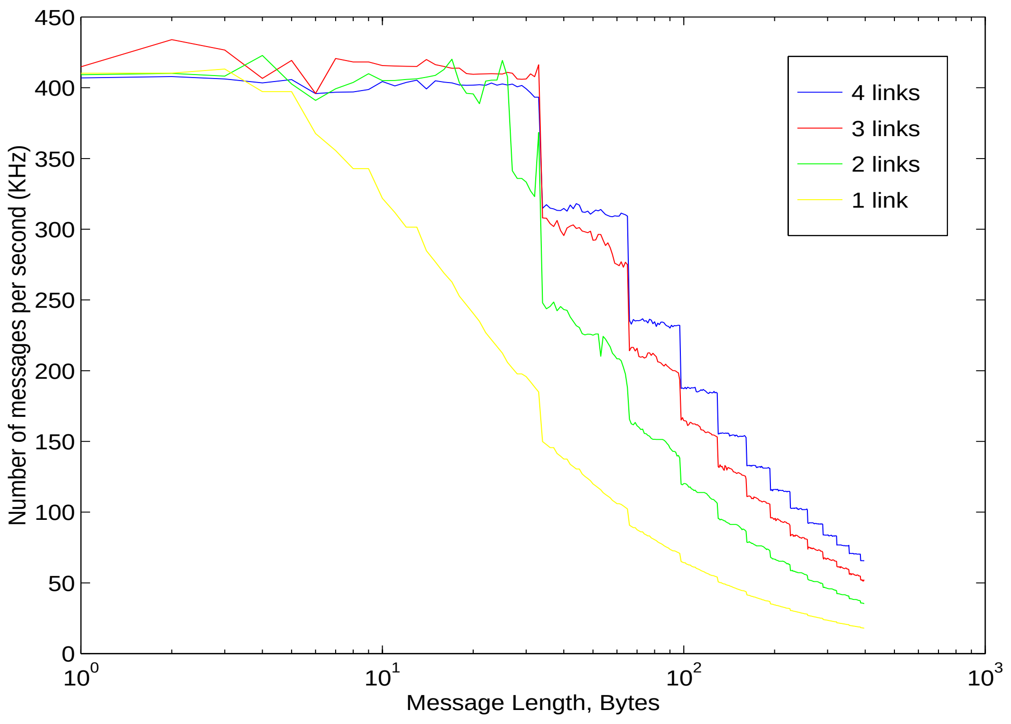


FIGURE 3.10 Short message sends, vary number of physical links, between two 20 MHz E03 T9000s connected by C104s, 64 bit interface 8 K cache 8 K internal memory. 100 Mbits/s links. 10 virtual links in use per physical link.

At 430 KHz the total CPU time required to produce each message is 2.33 microseconds. All processes are the same priority (high) so no process is ever pre-empted, i.e. a full context switch is not required. The I/O instructions only require partial context switches, because the I/O instructions do not require the entire state of the process to be stored. For each message the following steps will be carried out on the source:

- Current process performs output instruction

- I/O causes process to be descheduled and added to back of process queue, next process rescheduled in (a partial context switch)
- I/O performed by VCP

All these steps are carried out in a total CPU time of 2.33 microseconds. 1.4 microseconds is required for a partial context switch, and the remainder to perform the I/O. The results show that the measured performance is very close to the limits of the underlying hardware.

3.3.2 Combined Communication and Computation

An important aspect of parallel computing in general is the load which communications place on the CPU. There are two important times to consider for a communication: the actual time of the communication and the time the CPU is utilised.

A benchmark has been produced to measure the ability of a processor to perform communication and computation concurrently. I will calculate the percentage of the CPU's performance that is consumed for communication. This is not only a measure of the communications performance, but depends heavily on the interrupt and context switch performance.

A single low priority process performs computation and a high priority process repeatedly outputs short messages to another T9000 on a single virtual link. Figure 3.11 shows the four entities that are of interest in this benchmark (all processes on the source): the two processes, the scheduler and the I/O controller. This benchmark is of interest for two reasons:

- It measures the capability of the processor to perform communications and computation concurrently. It is important that a control/supervisor processor can send control messages at high rates and at the same time make decisions to perform the monitoring and control of the network.
- It allows a comparison between different platforms evaluating this behaviour.

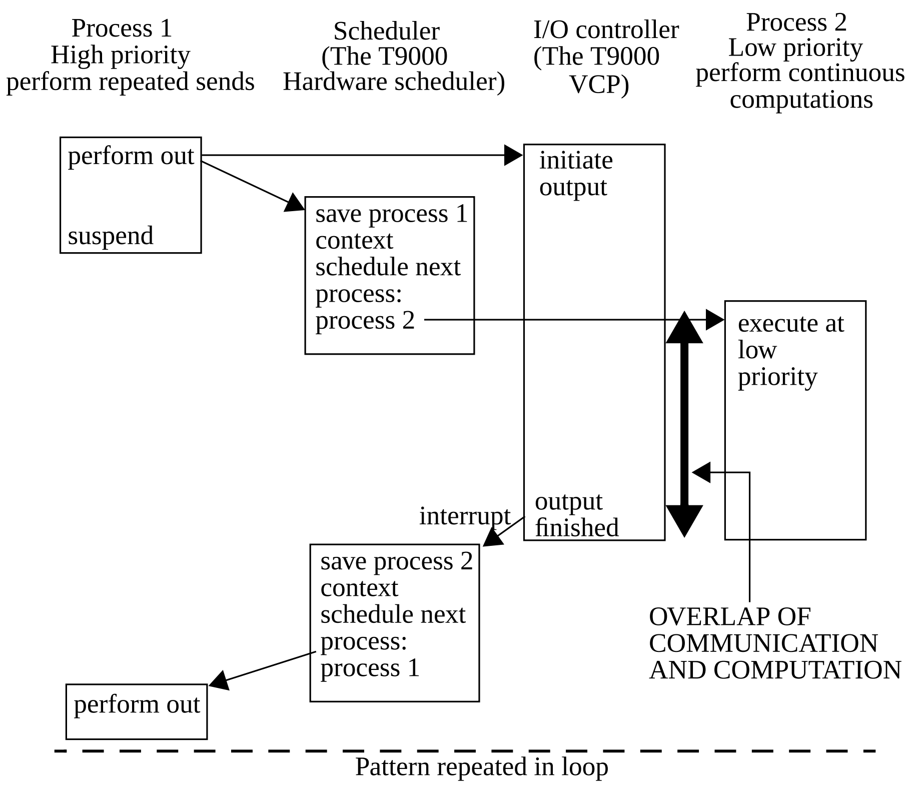


FIGURE 3.11 Overlap of communication and computation

The number of communications performed per second is the same whether the low priority computation process is present or not. The high priority process queue is always serviced before the low priority queue, the presence of a low priority process cannot affect the communications performance of a high priority process. This has been confirmed by measurement and is 104 KHz (for 4 to 32 byte messages). This result is an identical measurement to the one virtual link curve in Figure 3.9. In one second of running the benchmark the communication time is the full second, the communication is continuous (messages sent back to back).

The number of floating point operations performed by the low priority process was measured for two situations: with and without the presence of the high priority process communication. Dividing the with communication result by the without communication result gives the fraction of communication time (= total time) that can be used for computation.

The rate of floating point operations that the low priority process can perform is 1.43 MHz [7]. If the high priority process sends 4 byte messages the rate of floating point operations performed by the low priority process is 0.74 MHz, hence 52% of the communication time can also be used to perform computation. Using 4 byte messages a single message takes 9.6 microseconds (corresponds to 104KHz), 5 microseconds of this can be used for computation. Table 3.2 represents similar information for different message sizes.

7. The floating point operations are executed inside an occam SEQ loop. More efficient constructs are possible but for the benchmarks the relative performance of the computation with and without communication are all that is of interest.

TABLE 3.2 I/O load on CPU results, between 20 MHz E03 T9000s connected by a C104, 64 bit interface 8 K cache 8 K internal memory, 100 Mbits/s links

Message length (bytes)	Floating point operations performed by low priority process (MHz)	percentage of CPU used for I/O or I/O load on CPU	Rate of high priority communications (KHz)	Time for single message (microsecs)	Time left for computation in a single message (microsecs)
4	0.74	48%	104	9.6[a]	5.0
32	0.68	52%	104	9.6	4.6
64	0.80	44%	61	16.4	9.18
128	0.87	39%	34	29.4	17.9
256	0.92	36%	17.5	57.1	36.5
512	0.94	34%	9.0	111.0	73.3
1024	0.96	33%	4.6	217.0	145.0
10240	1.02	29%	0.47	2120	1510

a. The time for a single word message between two directly connected T9000s has been presented as 7.5 microseconds in Section 3.1.1, "Single Link Results,". The value for a short message between two T9000s connected via a C104 (two GPMIMD machine T9000s on the same motherboard) is 9.6 microseconds. This is due to the switching latency of the C104, the message is delayed for 1 microsecond and the acknowledge is delayed for 1 microsecond when they are switched through the C104. There is also a small increase due to the extra header required when packets are switched through a C104.

In all cases the I/O load on the CPU (I/O loading) is less than 100%. The percentage of communication time made available for computation generally increases as the message size increases, except for the 4 to 32 byte measurements. The T9000 is able to overlap communication and computation because of the onchip hardware support of the VCP. For larger messages, 71% of the communication time can also be used for computation, corresponding to 29% I/O loading. This demonstrates two important aspects of the Transputers performance:

- High rates of I/O should not saturate or heavily load the CPU
- I/O and computation should be performed concurrently

From the first entry in Table 3.2 for 4 byte messages it can be seen that 5 microseconds are allowed for computation, therefore 4.6 microseconds are required for the communication. The steps that require 4.6 microseconds can be identified. For each communication 2 context switches will be required. One context switch to reschedule the sending process and another to reschedule the computing process. These context switches will always be full context switches, the high priority process will always pre-empt the low priority process. The 4.6 microseconds is therefore made up of 2 full context switches (3.8 microseconds) and the extra time required to transfer data to the link.

3.4 Computation Benchmarks

The computational performance of the T9000 and Alpha processor have been measured for a number of standard benchmarks. The results are summarised in Table 3.3. The highest speed T9000 currently available has been used. It can be seen that the computational performance of the TransAlpha module is about a factor 7 higher than a 25 MHz T9000. Clearly the T9000 does not compare favourably, and a boost to the computational power (in the form of the TransAlpha module) was required.

TABLE 3.3 Computational Benchmarks

Benchmark	T9000, 25 MHz, revision E03, 64 bit interface, 16K cache	TransAlpha Module, 233 MHz
Linpackd 100x100 (Equation solving performance)	2.2 (1.0)	16.7 (7.5)
Livermore Loops (Fortran loop performance)	2.0 (1.0)	19.1 (9.6)
Rinf 1 (Vector arithmetic)	1.5 (1.0)	14.5 (9.7)
Poly 1 (Cache performance)	5.1 (1.0)	37.0 (7.3)
Poly 2 (Memory performance)	5.1 (1.0)	26.6 (5.2)
CERN benchmarks (12 Fortran HEP codes)	2.2 (1.0)	20.0 (9.1)

3.5 Comparison to other platforms

3.5.1 VMEbus PowerPC systems running LynxOS

Results for the PowerPC boards are taken from ‘An evaluation of VMEbus PowerPC based processor boards running the LynxOS operating system’ [16]. The boards evaluated vary from PowerPC 603 ‘CES 8067EA’ (64 MHz, no second level cache) to PowerPC 604 ‘CES 8067NA’ (96 MHz, with 512 Kbyte second level cache). These two boards are used for comparison to the T9000.

3.5.1.1 CPU performance

The standard Dhrystone and Whetstone [17] benchmarks are used to compare the microprocessor and memory systems performance of the two systems. The results are shown in Table 3.4.

TABLE 3.4 CPU performance

	PowerPC 603 (64 MHz)	PowerPC 604 (96 MHz, 512 Kbyte 2nd level cache)	T9000 (20 MHz revision D02, 32bit memory interface, 8K cache)
Dhrystone	109,900	212,800	29,411
Whetstone	9.8	16.4	7.14

The T9000 clearly suffers on CPU performance. Fundamental problems are the low clock speed, slow memory interface, extra NOP [8] operations and inefficient implementation of floating point operations. It is clear that if an application is compute intensive the TransAlpha module is required. All three systems used optimisation on compilers.

3.5.1.2 Interrupt response and process scheduling performance

A basic measure of scheduling performance is the time for a context switch. The time is 1.9 microseconds for the T9000. Two values are measured for the PowerPC boards: a process switching between its self (noswitch) and a context switch between two different processes (switch).

In addition the time required for a process to send a signal to another process can be measured. In the T9000 case, the measurements were for 4 byte messages and required between 3.2 and 4 microseconds (see Table 3.1 on page 41) depending on the priorities of the sender and receiver. The equivalent measurements have been made for the PowerPC boards and are referred to as sigs_sent_switch. The measurement for the PowerPC board is a bi-directional signal, i.e. a ping-pong, so the time is divided by two.

The only interrupt response time presented for the PowerPC boards is the VMEbus interrupt response. The interrupt is handled by part of the VMEbus driver, and does not reach the user application. The interrupt response time for a timer generated interrupt on the T9000 (into a user application) was measured at 2.0 microseconds.

Table 3.5 summarises the results.

TABLE 3.5 Process scheduling benchmarks

	PowerPC 603 (microsecs)	**PowerPC 604 (microsecs)**	**T9000 (20MHz rev E03) (microsecs)**
noswitch	13.4	5.46	1.4
switch (true context switch)	57.5	17.2	1.9
sigs_sent_switch (signal between 2 processes)	373.0	79.4	3.2 to 4.0
interrupt response	10.8 VMEbus interrupt to VMEbus driver	8.2 VMEbus interrupt to VMEbus driver	1.9 internal clock interrupt to actual user application

There is a clear improvement between the PowerPC 603 and the PowerPC 604. The T9000 benefits from a dedicated hardware scheduler and has lower context switch time by an order of magnitude. When the signal between processes has to be handled (sigs_sent_switch) then the gains are even greater. The comparison for interrupt handling is difficult, further measurements have to be made, revealing the time for a user application to respond to the interrupt for the PowerPC boards.

8. The revision D02 versions of the T9000 required extra null operations (NOPS) to avoid hardware bugs.

3.5.1.3 Supervisor/control processor benchmarks

No results corresponding to the short message sends and communication/computation overlap benchmarks are available for the PowerPC boards. The results would require multiple boards inter-communicating.

3.6 Conclusions

In this chapter I have presented and critiqued concepts and technologies used within the work of the thesis. Benchmarks have been presented and results understood and explained. I have identified a set of critical factors which affect the performance of multi-processor systems. In Table 3.6 and Table 3.7 I summarise the factors which have been evaluated, along with their relevance to performance in multi-processor systems. The factors are split into two groups: factors vital to network performance and factors vital for nodes driving the network (see Table 3.6 and Table 3.7). The text following summarises the performance of the T9000 and C104 for each factor. I introduce the following definitions for message overheads and latencies:

- The message passing overhead: The elapsed time to transfer a zero length message from an application program in a source to an application program in a destination. It can be divided into two components, the network latency and the node message latency.
- The network latency: the time to propagate a single byte through the switching network
- The node message latency: the I/O initiation time in a source or destination.

A single message passing overhead value will be composed of three parts: a node message latency in the source, a network latency and finally a node message latency in the destination.

TABLE 3.6 Critical factors affecting network performance

Factor	General relevance or importance to performance
C104 adaptive routing	Adaptive routing allows the efficient use of all links available between the terminal and centre stages of a Clos network. A possible alternative of grouped adaptive routing is to have nodes pre-determine the route which a packet would take before it entered the network, adding load and complexity to the nodes. Another alternative would be a fixed route through the network for all packets which would increase contention.
Scalability	Performance must scale to large networks
Network latency	The time to switch packets through the network is a component of the message passing overhead. Therefore it should be minimised, the C104 switches packets in one microsecond.
Link bandwidth	To provide high throughput the network will require high speed component links. However, this is a necessary but not sufficient condition for high throughput, the performance of the node interfacing to the network will also be crucial.

TABLE 3.7 Critical factors affecting nodes driving the network

Factor	General relevance or importance to performance
Node message latency	Low node message latencies are important to allow low message passing overheads.
Link interface bandwidths	High link interface bandwidths between the raw link and the application program are crucial to exploit high bandwidth links.
The VCP and virtual channels (facilitating low node message latency and high link interface bandwidths)	Low I/O loading due to the VCP is crucial for high message rates, efficient support for concurrent computation/communication and low message latencies. The VCP facilitates low node message latency and high link interface bandwidths. The VCP is essentially an efficient tightly coupled on-chip communications controller for the T9000.
Fast context switch times	Fast context switch times are important for multi-processing and exploiting low I/O loading of the CPU to provide efficient support for concurrency of communications and computation. A process must be scheduled quickly and efficiently to exploit the CPU time made available by the low I/O load. Interrupt times are crucial in any system with real-time constraints.
Multiple physical links	Additional performance, fault tolerance and communications priorities can be provided by multiple networks.
Computational performance	There are a number of compute intensive applications where multiple processors with high computational performance are required. A possible solution to the shortfall in T9000 computational power are hybrid nodes, e.g. the TransAlpha, an high performance RISC processor using a dedicated communications controller.
Language support for parallel processing	The programming interface and environment are crucial to allow the user to exploit the performance of the hardware. Occam provides an example of the advantages of a language designed for parallel processing, see comparison in Section 2.2, “Occam,” on page 6.

The communications results are very promising. Single link results show low overheads and low latencies. The C104 has been shown to switch packets in a single microsecond.

I have demonstrated the ability of the T9000 using the VCP to concurrently perform communications and computation due to low I/O loading on the CPU. Low I/O loading allows concurrent computation and communication, with the communication performed concurrently over many virtual links on multiple physical links. The control/supervisor benchmarks show high message rates from a single T9000 node (greater than 400,000 per second).

The C104 performs the grouping of links with no measurable overhead, and the achieved bandwidth has scaled linearly with the number of links grouped. A large network has been built and benchmarked: cross-sectional bandwidths of over 600 Mbytes/s have been measured for the GPMIMD machine.

The uni-directional communication results for the T9000 scale linearly for one to four physical links. The bug fixes in the revision E03 T9000 allow all four of its links to be used in parallel.

The T9000 revision D02 suffered from serious external memory problems, the 64 bit interface T9000 could only write data at 16.4 Mbytes/s to the links from external memory, whereas data could be written to the four links at 28 Mbytes/s from internal memory.

Context switching performance and interrupt response have been measured and compared to other platforms. Context switch times are better by orders of magnitude.

The late delivery of the T9000 has meant that in the meantime many vendors have passed the 50 MHz full specification of the T9000. The T9000s that finally became available are only 20 MHz, in the near future perhaps 30 MHz, which has created a situation where the T9000 has a fundamental shortfall in computational power. This has been demonstrated in comparisons to the PowerPC and DEC Alpha. The solution to this particular shortfall has been the development of the TransAlpha board.

The Gamma E03 T9000 is now commercially available. At the time of writing the GPMIMD machine has been upgraded to 64 Gamma E03 T9000s and 58 beta C104s. The main improvement over the D02 is the increased stability due to the removal of hardware bugs. The external memory interface has also been improved. The external memory interface is crucial for the efficient operation of the TransAlpha board.

The T9000 is still interesting due to its specialised design for use in multiprocessing, in particular the communications performance through on chip hardware and software development environment.

Chapter 4
The application of the T9000 to the CPLEAR experiment

In this chapter I present a system of T9000s and C104 switches operating as an on-line event filtering farm in the CPLEAR experiment [18]. The work has been carried out within the framework of the ESPRIT project GPMIMD. An objective of the GPMIMD project was to build a parallel scalable computer using the T9000 and C104, the GPMIMD machine. The role of CERN within this project included the mounting of an HEP application onto a 64 node machine: a processor farm in the CPLEAR experiment.

An introduction to the CPLEAR experiment is given in Section 4.2, “The CPLEAR Experiment,”. Section 4.3, “T9000 Processor Farm System Overview,” presents an overview of the system used, details on the components are given in the following sections.

Results presented are based on the three week experiment run in September/October 1994. The problems encountered are discussed and projections are made to investigate the feasibility of the T9000 and C104 for use in building processor farms in CPLEAR. Work involving the use of TransAlpha modules to boost the computational power of the T9000 is also presented.

4.1 Motivation

4.1.1 Value to CPLEAR

In 1994 the CPLEAR off-line event reconstruction involved the recording of approximately 15,000 tape cartridges every year. This corresponded to an event rate of 350 Hz, producing 1 Mbyte of data per second. When this data was processed off-line 80% of it was rejected and then discarded, it was not of interest for physics analysis, therefore it need not have been recorded. Four fifths of the tapes recorded were not required. The turnaround time for off-line processing, i.e. the time between an event being recorded and the final reconstructed and selected events are produced, was measured in months. Histograms were only available for a small subset of the data on-line.

An on-line processor farm which could reconstruct and filter the full event rate would have removed all of these problems. There would have been an 80% reduction in the amount of data that needs to be stored. The turnaround time for reconstructed events would have been measured in seconds and histograms available on the full event rate, allowing better monitoring of the experiment. The aim of the application of the T9000 to CPLEAR was to demonstrate the feasibility of using a processor farm to reconstruct and filter the full event rate on-line.

4.1.2 Proof of existence and successful operation of the T9000 and C104

The T9000s used in the CPLEAR experiment were revision D02 prototypes.The C104s used were revision alpha. Little or no experience existed in building large systems with these or any other T9000 revisions. The networks assembled at CERN were larger than any others in existence at that time. This resulted in the discovery of many problems before they were seen by the manufacturer of the chips or the companies building the Transputer systems.

The hardware and software had multiple flaws and bugs, the details have been presented in chapter 2. Any one of the revision D02 T9000 hardware bugs could have completely disabled the T9000 systems, or have made them completely unreliable.

Despite these problems I believed that the T9000 was unique in the way it approached the requirements of multiprocessing. It was very important to show that large T9000 and C104 systems could be constructed and operated reliably in an experimental environment. At the time of implementation, there was no other application using the T9000 and C104 on a similar scale.

A reliable T9000 and C104 platform operating in an experimental environment was also important for the GPMIMD project and our industrial partners within that project. Within the project a technology demonstrator was required to prove the feasibility of using these technologies to build stable systems outside of the manufacturers test environment.

4.1.3 Use of the T9000 for future generations of HEP experiments

The final aim was to investigate the use of the T9000 and C104 in data acquisition studies at the LHC. The data acquisition systems for the LHC will require large scale switching networks, and few technologies were (and still are) available to investigate the performance and behaviour of switching networks. The T9000 and C104 were technologies that offered such switching networks, and the application of the T9000 to CPLEAR produced valuable information for the LHC.

4.2 The CPLEAR Experiment

4.2.1 Physics Background

The aim of the CPLEAR experiment is to study CP (Charge-Parity), T (Time) and CPT -violation phenomena on the neutral kaon system. The work is intended to give insight into the abundance of matter present in the universe today. The violation can be seen in asymmetries between decays of K^0 and $\overline{K}^0$ to their respective final states f and $\bar{f}$ ($\bar{f}$ is the CP conjugate state of f) as a function of decay time. Where f may be any of the following decay products:

$\pi^+\pi^-$	$\pi^0\pi^0$
$\pi^+\pi^-\pi^0$	$\pi^\pm e^\mp \upsilon$
$(\pi^0\pi^0\pi^0)$	$(\gamma\gamma)$

In the experiment anti-protons are accelerated in the LEAR ring at CERN and collide with a fixed target. This target is located in the centre of the detector, see Figure 4.1, it is filled with hydrogen at 16 bars. The proton anti-proton pair annihilates via the following reactions, producing K^0 and $\overline{K}^0$:

$$p\bar{p} \Rightarrow K^{+}\pi^{-}\overline{K}^{0} \qquad \text{(EQ 1)}$$

$$p\bar{p} \Rightarrow K^{-}\pi^{+}K^{0} \qquad \text{(EQ 2)}$$

Subsequently, the K^0 and $\overline{K}^0$ decay into one of the six possible decay products shown above.

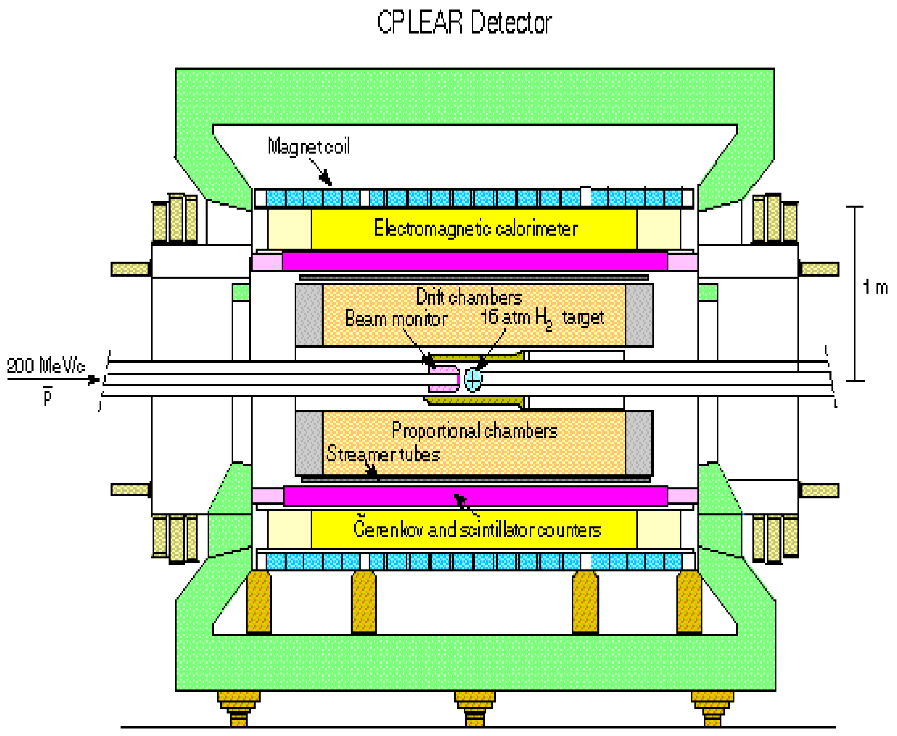


FIGURE 4.1 The CPLEAR Detector Schematic

4.2.2 Event Detection

The K^0 and $\overline{K}^0$ are neutral, hence they do not give a visible track. Using EQ1 and EQ2 (proton anti-proton reactions) the K^0 and $\overline{K}^0$ can be differentiated by identifying the charged kaon and the sign of the charge. Charged particle identification is performed by a Cerenkov counter sandwiched between two scintillators.Tracking is performed in a cylindrical decay volume with two layers of proportional chambers, six layers of drift chambers and two layers of streamer tubes. A calorimeter allows the identification of neutral pions through their decay photons. The whole apparatus is located inside a solenoidal magnet which allows momentum and charge determination.

A multi-level trigger system is used to select the decay channels of interest. The aim is to record the charged Kπ pair produced in the annihilation. This requires a two or four track event and at least one of these tracks should be a kaon candidate. A typical CPLEAR event is shown in Figure 4.2.

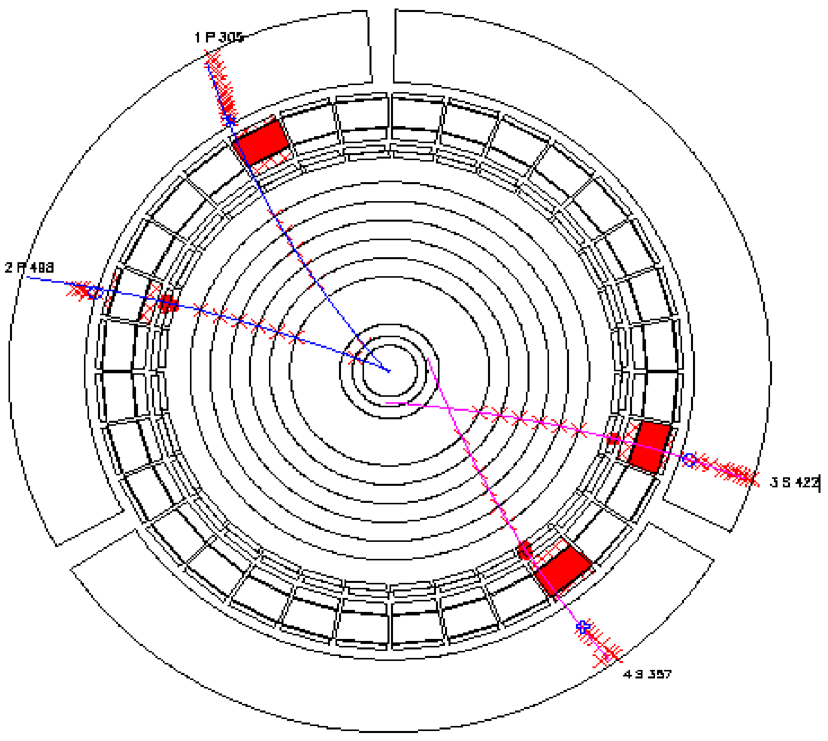


FIGURE 4.2 A Typical CPLEAR Event

4.2.3 Data Acquisition

The CPLEAR data acquisition system is shown in Figure 4.3. It is a distributed VME system collecting data together from the 7 sub-detectors. Each of the sub-detectors corresponds to a Root Read Out (RRO) which transfer data concurrently to a corresponding area of memory located in the Event Builder (EB) via point to point links. The Event Builder collects these event components into standard Zebra blocks of approximately 10 events where one event is approximately 2 Kilobytes. The VIC bus (VME interconnect) is then used to transfer the data to a VME crate (Tape) where blocks are recorded to tape cartridges, for off-line filtering and reconstruction. CPREAD is the standard off-line reconstruction and event selection program. In addition, the monitoring crate receives a sample of Zebra blocks and sends them to a Vax cluster via Ethernet, where detector monitoring and data checks are performed. The Transputer network was added to the VIC bus by a Transputer to VME interface board.

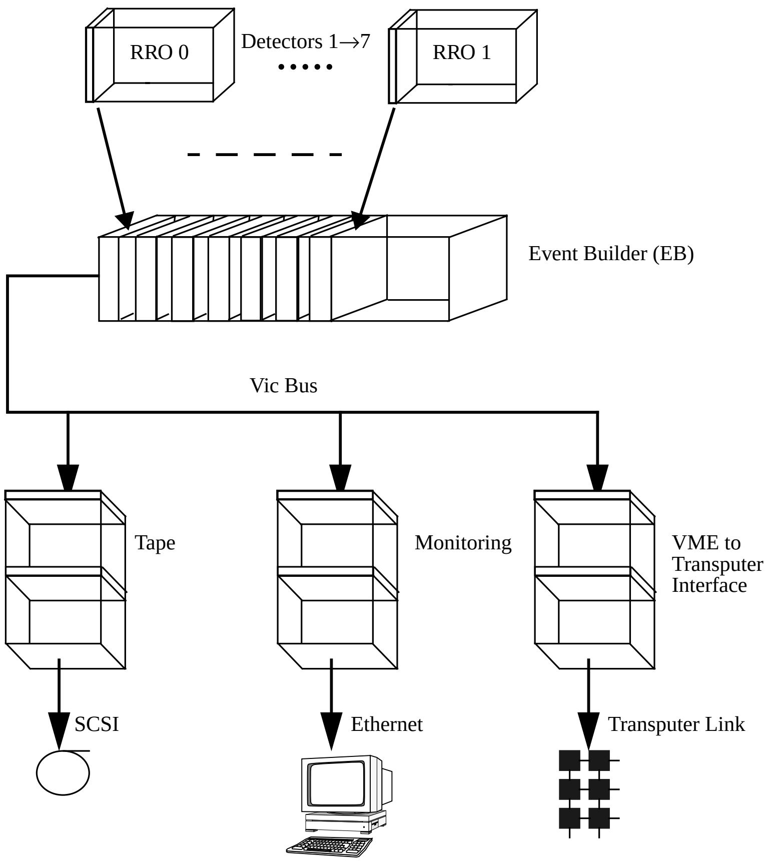


FIGURE 4.3 The CPLEAR Data Acquisition System

4.3 T9000 Processor Farm System Overview

The T9000 processor farm was interfaced to the experiment via an existing T805 data acquisition system [19], see Figure 4.4. The T805 is a previous generation of the Transputer using 20 Mbits/s over sampled (OS) links.

The T805 system was required as standard I/O devices were not available at the time for the T9000. For example, a SCSI interface was not available[9] which was required to write the reconstructed data to Exabyte tape, and a T9000 to VME interface was in development[10].

9. At the time of writing a SCSI interface is available.

10.At the time of writing a T9000 to VME interface is available [20].

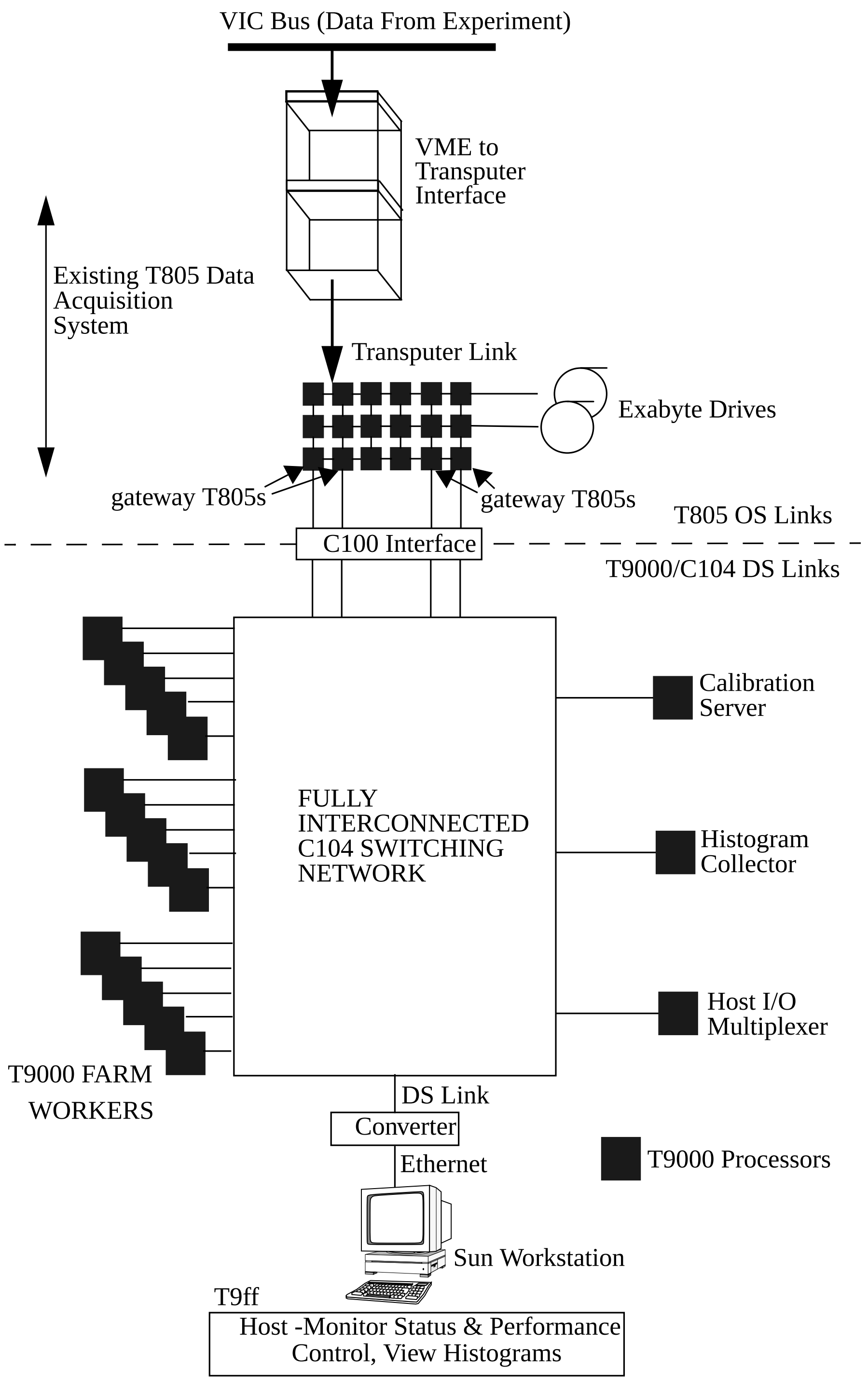


FIGURE 4.4 System Overview

The T805 network provided the following services:

- access to the raw experiment data
- access to Exabyte tapes.

The T9000 ran the standard off-line version of CPREAD on-line. CPREAD produces a set of standard histograms displaying statistics on reconstructed events. In addition to raw data and access to Exabyte tapes CPREAD required the following services:

- Access to the latest version of the calibration files. A calibration server process ran on a T9000, and read a single copy of the calibration files from the host. It then serviced requests from the farm workers for calibration files.
- Access to an histogram collector, which summed together all histograms from all farm workers. A histogram collector process ran on a T9000, this process then sent the histograms to the host.
- Access to the host to transmit status and performance information. The host software only allowed a single process to be connected to the host. All farm workers required host access, so a host I/O multiplexer was implemented on a T9000. This process multiplexed all I/O requests onto the single connection to the host.

A monitoring and control program was written, T9ff (T9 Farm Frontend), which displayed the status and performance of all workers. The user also used T9ff to start and stop the operation of the T9000 network. T9ff interfaced to PAW and provided access to all the standard CPREAD histograms.

In contrast to the T9000 the T805 had no VCP and there was no dynamic packet switch for OS links. On the T805, virtual links were implemented in through-routing software. If a channel was not connected between processes on directly connected T805s, then the message would have to travel through some number of intermediate T805s between the source and destination T805. This reduced the performance of the communication and the intermediate nodes, the intermediate nodes having to run software to service messages destined for other T805s. The software through-routed virtual links on the T805 were not designed to connect to the virtual links of the T9000.

The T805 network was connected to the T9000 network via a C100 DS to OS link converter. The interface is presented in detail in Section 4.7, “T805 to T9000 Interface,”. For any T805 to communicate with a T9000, DS link communication protocols had to be implemented in software on the T805 to allow it to interface to the virtual links of the T9000 system. The T805s directly connected to the C100 links ran this software, the T9000s could only communicate to these so called gateway T805s, see Figure 4.4. The gateway T805s serviced requests from the T9000s: to supply raw data or write back reconstructed data to the Exabyte drives. A full description of the T805 network can be found in ‘A Transputer based Scalable Data Acquisition System’ [19].

4.4 Play-back mode

The T805 system provided raw data to the T9000s from one of two sources: either previously recorded raw data stored on Exabyte or data acquired from the experiment in real time. This option was controlled via the T805 user interface. The use and configuration of the T9000 system was independent of the source of raw data. The T805 network was used to write these raw data tapes during normal data taking from the experiment.

This option was crucial for the testing of the system and CPREAD code before the experiment runs. It was the only way to test CPREAD on large amounts (Gbytes) of events, host disk systems were limited in size (Mbytes) and provided data at low rates. The bandwidths over the host links were the bottleneck. Some problems with the port of the CPREAD code only occurred for one in 100,000 events (or more). Testing with limited numbers of events would leave many problems uncovered. Play-back mode also meant that the T9000 network could be used as an off-line analysis tool, to re-analyse data, a very important and necessary feature.

4.5 Event types

Each reconstructed event had a certain physics type. Six different types were identified and each was assigned to a different Exabyte drive. The different event types corresponding to six different Exabyte drives were: 2π,3π, πeν, others, sample and two track events. The CPREAD code was changed so that it sent different event types out on different virtual links. The event type could then identified by the gateway T805 using the virtual link on which the event was transmitted. The gateway T805 then sent the event data to a T805 which wrote the data to the correct Exabyte. The events were sorted into different physics groups and the load was spread over the different Exabyte drives.

The overall use and configuration of the Exabyte tape drives is shown in Figure 4.5.

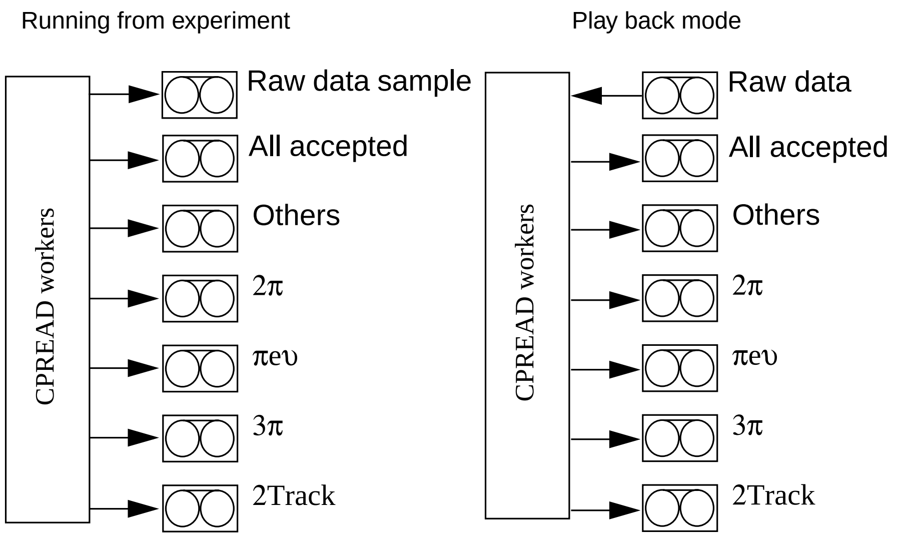


FIGURE 4.5 Exabyte configuration

Figure 4.6 summarises all the connections that were made to the CPREAD workers. Each connection was represented by a separate virtual link. An interface was written in C to translate the Fortran I/O statements used by CPREAD into the Transputer toolset channel I/O functions. The interface also performed the transmission of histogram information and constructed events on the correct virtual link. The interface was written in C and not occam because all the CPREAD code was in C (produced by f2c).

A simple protocol was used between the gateway T805s and T9000s to identify messages. A single positive integer was transmitted from a T9000 to a gateway T805 to signal that a Zebra block of raw data was required (using the function ChanOutInt). The destination T805 would then send a Zebra block of raw data. A single negative integer was sent from the T9000 before it sent a reconstructed Zebra block. The block would not be sent until the T805 had received the integer, confirming it was ready to service the request. The gateway T805s would continually wait until they received an integer, at which point they would act accordingly depending on the sign of the integer. The protocol was not strictly required as the virtual link that a communication was made on was sufficient to identify its type. However, the protocol added security and the additional overhead was insignificant.

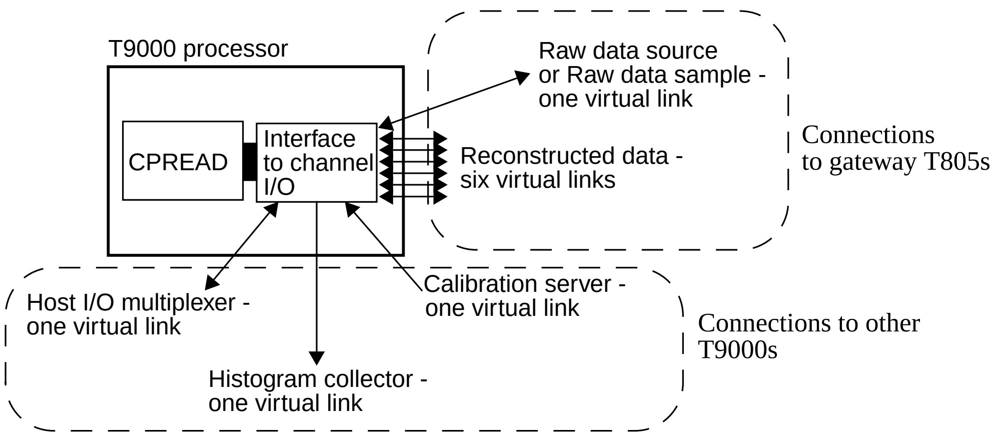

T9000 processor
CPREAD
Interface to channel I/O
Raw data source or Raw data sample - one virtual link
Reconstructed data - six virtual links
Connections to gateway T805s
Host I/O multiplexer - one virtual link
Calibration server - one virtual link
Histogram collector - one virtual link
Connections to other T9000s

FIGURE 4.6 Virtual links to CPREAD workers

4.6 Porting of CPREAD

The CPLEAR event reconstruction code (CPREAD) has been ported onto the T9000. The whole package contains 230,000 lines of Fortran’77 code. The AT&T Fortran-to-C (f2c) compiler was used to translate all Fortran into C due to the present lack of a Fortran compiler for the T9000. No optimization of CPREAD was performed during this step and only minor changes were imposed due to the use of f2c.

In addition, the CERN programming libraries, used by CPREAD, have also been ported to the T9000. These include 170,000 lines of Fortran and additional C code. These libraries include data structure, histogram and mathematics packages. This work demonstrates that other large physics Fortran applications may be ported onto the T9000. A native Fortran compiler would improve the performance of the T9000 for applications written in Fortran.

4.7 T805 to T9000 Interface

4.7.1 Overview

All T9000 farm workers required access to the T805 network. The T805 provided raw data from the experiment or Exabyte tape and wrote back all reconstructed data to Exabyte tapes. For any T805 to communicate with a T9000 virtual link it had to emulate the T9000 DS link communication protocols.

The protocol used by T9000 DS links to implement virtual links is based on the exchange of acknowledged packets of up to 32 bytes, whilst the OS link protocol of the T805 is based on the exchange of acknowledged bytes. The two protocols are not compatible.

There was minimal support available for the integration of T805 and T9000 systems, in the form of the low-level OS to DS link protocol conversion offered by the C100. The T805 system was connected to the T9000 via the C100. A C100 connects 4 DS links to 4 OS links. The C100 converts a sequence of bytes on the OS link into a DS link packet, or vice versa. In Figure 4.7 a user process on a gateway T805 outputs a message, which is an array of bytes. This array is passed to an interface process which implements all communication protocols used by the T9000 above the token level in software on the T805. This is splitting the message into packets or packetisation (T9000s uses packets of up to 32 bytes), headers (required for routing through C104s and identifying destination virtual links) and packet acknowledgements. For a full description of the protocol layers see Section 2.3.2, “DS Links,” and Section 2.3.3, “Virtual Channels and the T9000 Virtual Channel Processor (VCP),”.

Once the interface process has formed the first packet of the message (added the required headers and data bytes) it sends the packet out onto the OS link as a sequence of bytes. The first byte contains two pieces of information, whether the packet is the last packet of a message (EOM/EOP) and the total length of the packet. The C100 then knows how many bytes will be sent on the OS link to make up the packet and whether it should end the packet with an EOM/EOP token. The interface process will send the data bytes for the packet which includes any headers that are required for the DS packet. The C100 has no knowledge of headers, it simply forms a packet from all the bytes sent from the interface process. When the C100 has received the correct number of bytes it transmits a single DS packet containing the header and data bytes collected from the OS side and appends a EOM or EOP token.

The gateway T805s implemented this interface software (which I had to provide) and were the only T805s that communicated with the T9000s.

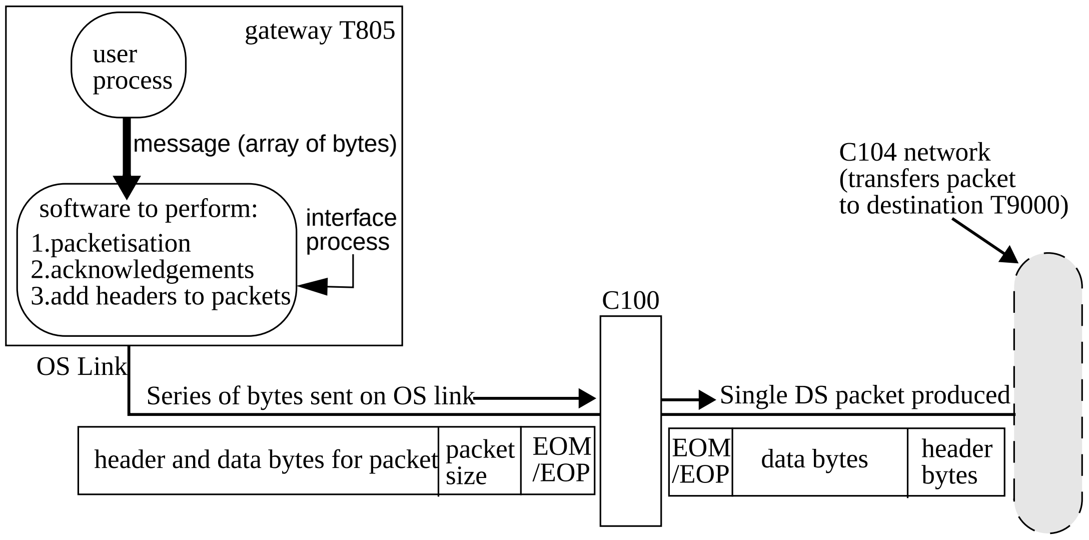


FIGURE 4.7 A T805 to T9000 interface, example shows T805 communicating to T9000

The T9000s declared the T805 link as an edge, the T9000 uses an edge to connect to an unsupported unknown device, i.e. one which is not part of the DS link control chain. The edge is specified in the hardware description file, software processes can then place virtual links onto this edge if they wish to communicate with it. The T9000 and the toolset has no information about the device that will be connected to this edge, they still expect the full T9000 DS link protocols to be adhered to for communications across this edge.

Each of the T9000 farm workers had seven virtual links to the T805 system, all of which had to be connected to gateway T805s. Figure 4.8 shows the mapping of virtual links onto the gateway T805s for two T9000s. One virtual link corresponded to the raw data source and six to the individual Exabyte drives for writing back reconstructed events. All 4 gateway T805s could supply a T9000 worker with raw data, the T9000 workers were spread evenly over the 4 gateway T805s. One quarter of T9000s requested data from gateway T805 1, one quarter from T805 gateway 2, etc. All events of a certain type had to be sent to the same gateway T805. For example, all 2Track events from all workers had to be sent to gateway T805 3.

Each gateway T805 was connected to two other T805s, a feeder and a tape driver. The feeder provides raw data from the B016, the Transputer to VME interface which connects to CPLEAR. The tape driver writes the reconstructed events to the Exabyte drives. The software running on the gateway T805s was able to identify the virtual link on which a message was received, allowing it to identify the message and send it to the appropriate Exabyte.

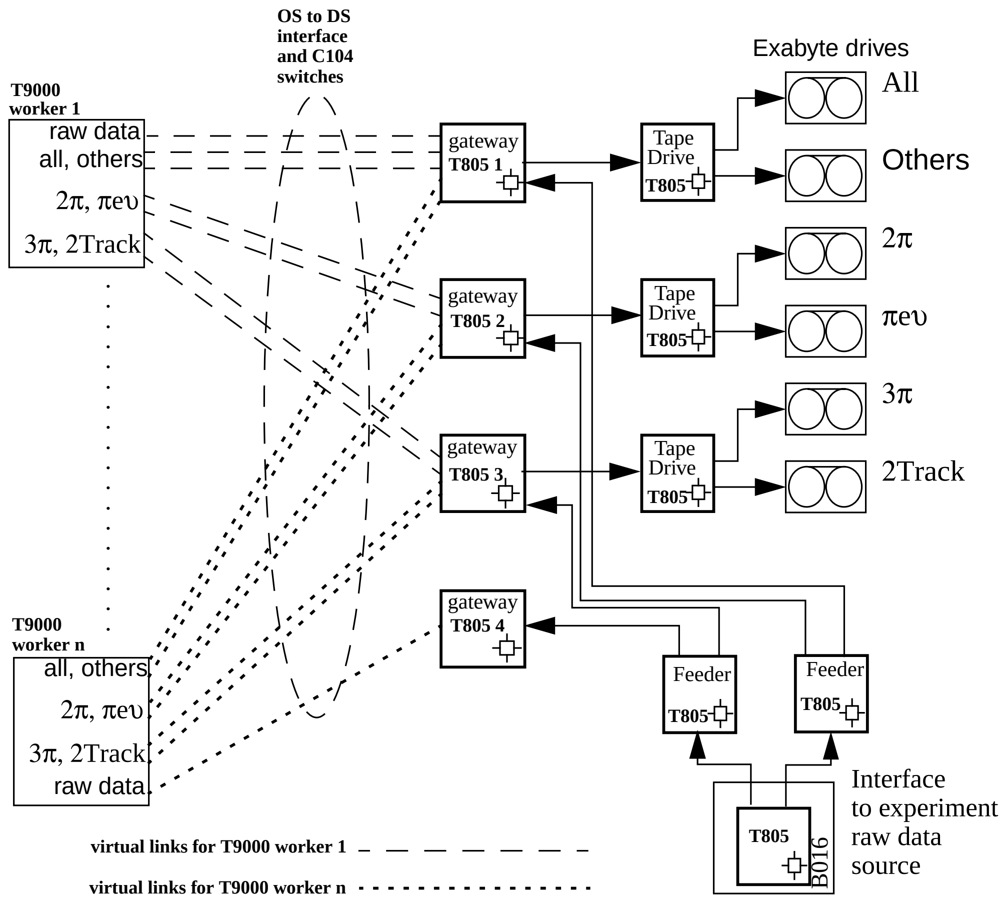


FIGURE 4.8 Mapping of virtual links from the T9000s onto the gateway T805s

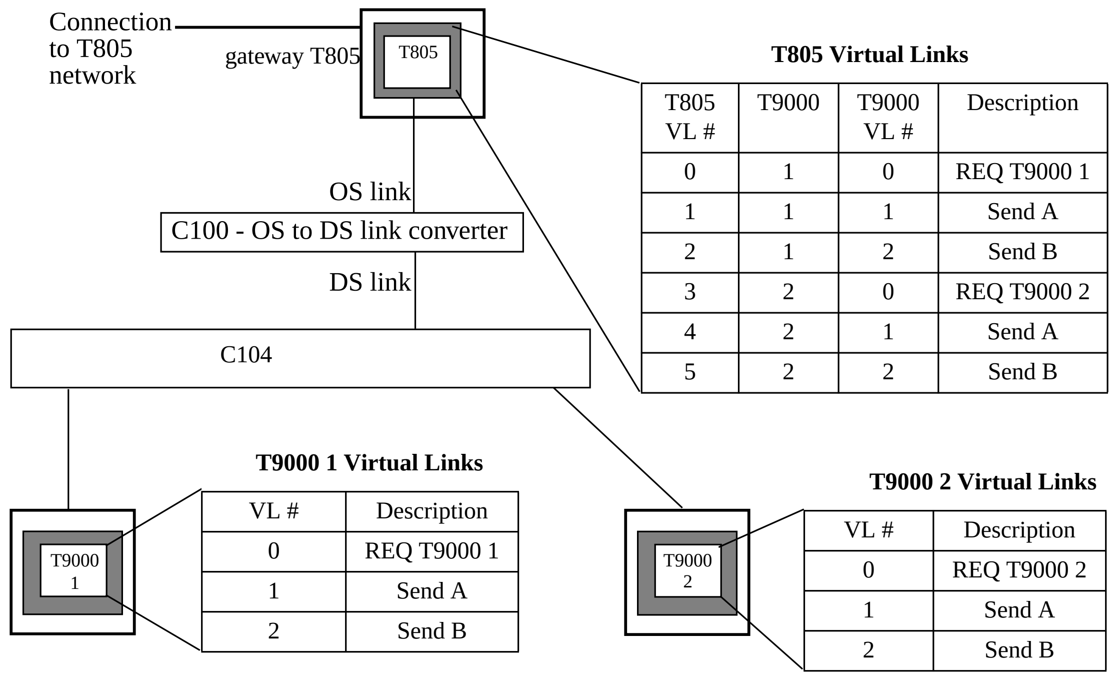
4.7.2 An example T805 to T9000 interface

This section presents the information required by the T805 and T9000 to communicate with each other. Figure 4.9 shows a simplified example of connecting T9000 virtual links into a gateway T805. For each T9000, three virtual links map across to the gateway T805. This is less than in the actual implementation, but demonstrates the principles involved. The tables for each T9000, in Figure 4.9, show two pieces of information: the virtual link number (VL #), and the description of the use of that link. In this example virtual link 0 of each T9000 will be used to request raw data from the gateway T805, and two other virtual links (1 and 2) will be used to send back reconstructed event data to the T805. The virtual links 'Send A' and 'Send B' would correspond to the two different destination Exabytes. The table for the gateway T805 shows all the virtual links mapped onto the T805 link.

Each table entry on the gateway T805 corresponds to one table entry on a T9000: the two end points of a virtual link. Each table entry connects to a single virtual link, even though it may have different virtual link numbers on the T805 and T9000. T9000 2 virtual link number 2 and gateway T805 virtual link number 5 are the two end points of a single virtual link. A virtual link is bi-directional, T9000 2 virtual link number 2 can send to T805 virtual link number 5 and T805 virtual link number 5 can send to T9000 2 virtual link number 2. A virtual link is made up from a single virtual channel in each direction.

For the gateway T805 to send or receive on any virtual link to the T9000 it must comply with the T9000 DS link protocol. It must produce 2 headers, one to route the packet through the C104 switch and one to identify the destination virtual link. It must split (packetise) data into 32 byte packets, and it must acknowledge all packets that arrive on a virtual link. These acknowledge packets must also have the appropriate 2 return headers.

When a T805 link receives a packet the header allows the identification of the virtual link on which the packet was received. This number is shown as T805 VL# in the T805 virtual links table. This virtual link uniquely identifies which T9000 has sent the packet and which one of the three T9000 virtual links the packet was sent from. The receiving gateway T805 can then acknowledge and provide raw data to the correct virtual link on the appropriate T9000.



T805 Virtual Links

T805 VL #	T9000	T9000 VL #	Description
0	1	0	REQ T9000 1
1	1	1	Send A
2	1	2	Send B
3	2	0	REQ T9000 2
4	2	1	Send A
5	2	2	Send B

T9000 1 Virtual Links

VL #	Description
0	REQ T9000 1
1	Send A
2	Send B

T9000 2 Virtual Links

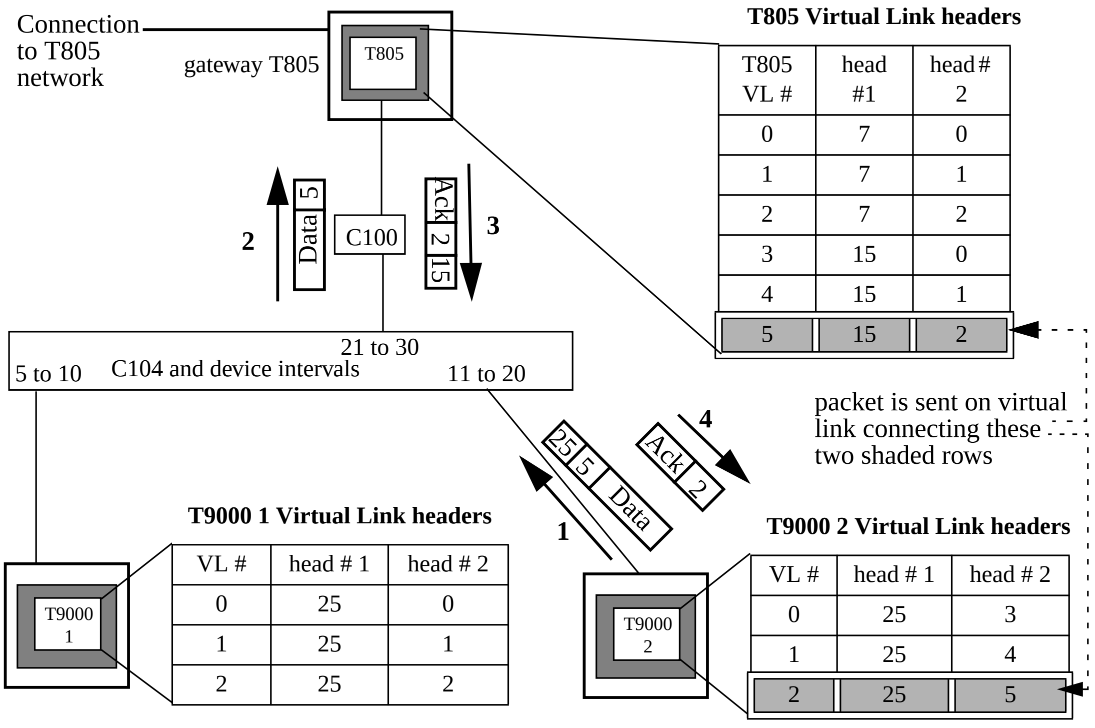
VL #	Description
0	REQ T9000 2
1	Send A
2	Send B

FIGURE 4.9 Mapping of virtual links between gateway T805 and T9000s

The C104 uses a device interval on each link to choose an output link for a packet, see Section 2.4, “The C104 Packet Switch,”. The intervals for the output links of the C104 are shown in Figure 4.10. A packet with a header between 5 and 10 will exit via the link to T9000 1. A packet with a header between 11 and 20 will exit via the link to T9000 2. A packet with a header between 21 and 30 will exit via the link to the C100 and then gateway T805.

Figure 4.10 is a more detailed representation of Figure 4.9, it includes the header values that must be used for each virtual link. The four packets shown in Figure 4.10 represent a single data packet (and its corresponding acknowledge packet) sent from T9000 1 virtual link 2 to the gateway T805 virtual link 5. The packet is typically part of a message sent on the send B virtual link. The following points detail each of the four packet states shown in Figure 4.10.

- Packet state 1. T9000 1 sends a single packet on virtual link 2. The other end of this link is gateway T805 virtual link 5. The packet is sent with two headers: 25 and 5. The 25 routes the packet through the C104.
- Packet state 2. The C104 has routed the packet towards the gateway T805. The first header has been deleted, it is no longer required. The final header (5) is still present to allow the T805 to identify that the packet is destined for virtual link number 5.
- Packet state 3. The gateway T805 has identified and received the packet on virtual link 5. It then acknowledges the packet. The acknowledge requires two headers: 15 and 2, which are taken from the shaded row of the table. The 15 is used to route the acknowledge to the correct T9000.
- Packet state 4. The C104 has routed the packet to T9000 2. The first header has been deleted, it is no longer required. The T9000 reads in this acknowledge and recognises it is for virtual link 2. Subsequent packets can now be sent on virtual link 2.


Connection to T805 network
gateway T805
T805
2
Data 5
C100
Ack 2 15
3
21 to 30
5 to 10
C104 and device intervals
11 to 20
25 5 Data
1
Ack 2
4
packet is sent on virtual link connecting these two shaded rows
T9000 1
T9000 2

T805 Virtual Link headers

T805 VL #	head #1	head # 2
0	7	0
1	7	1
2	7	2
3	15	0
4	15	1
5	15	2

T9000 1 Virtual Link headers

VL #	head # 1	head # 2
0	25	0
1	25	1
2	25	2

T9000 2 Virtual Link headers

VL #	head # 1	head # 2
0	25	3
1	25	4
2	25	5

FIGURE 4.10 Virtual link header values

4.7.3 Extracting packet headers for the gateway T805

I have presented the process by which a T9000 can communicate to a T805 and I have presented the information required by the T805 and T9000. Both the T805 and the T9000 require a table, containing virtual link numbers and all headers which these virtual links will require.

The packet headers and virtual link numbers used by the T9000 are produced by the configuration tool, *inconf*, and are loaded into the T9000 memory at boot time. Inside the T9000 packet headers are stored in a table, which contains the headers for packets sent on each virtual link. The configuration tool will not produce the equivalent headers for the gateway T805. I had to provide all the information in the table for the gateway T805 in Figure 4.10 explicitly.

The headers for the T805 can be constructed from the following two pieces of information:

- The contents of the T9000 virtual link header tables shown in Figure 4.10.
- Device intervals for C104 links, also shown in Figure 4.10.

The T9000 virtual link headers for a link to send/receive messages and acknowledges are produced by *inconf* at the configuration stage, refer to Section 2.5, “The T9000 Toolset Development System,” for details of the T9000 tools. The configuration tool produces a binary file, the internal format of this file was not known. The configuration tool did not have options to output the headers. The binary file produced by *inconf* is used by *icollect*, the code collector. An unsupported undocumented feature of *icollect* was found that printed all the virtual link headers. The problem was that it did not match each table (set of headers for a processor) to a particular T9000 processor.

Another method had to be devised to relate these tables to the T9000 they represented. An occam program was written to read and output to the host the memory locations of the T9000 that contained the virtual link headers. The output of this program would provide a list of headers used by each T9000, which allowed the matching of tables to processors in the output of *icollect*. This program needed to have exactly the same virtual link configuration as the CPREAD workers, histogram collector, host multiplexer and calibration server that it was temporarily replacing.

These steps allowed us to obtain a table of virtual link headers for each of the T9000s. The device intervals for the C104 are contained in the hardware description file, the ndl file, which is a human readable source file. In the case of networks used in CPLEAR I had already produced the device intervals when I had written the ndl.

This provides the two pieces of information required to produce the table of virtual link headers for the T805: the virtual link headers for each T9000 and the device intervals for each C104 link. This information was stored in a file on a host system accessible by the T805 system. The T805 virtual link headers were read from the file by the gateway T805 software which performed packetisation, addition of headers and production of acknowledges.

4.7.4 Remaining problems

Four main problems remain:

- The process of interfacing the T805s and T9000s has not been automated. The user must still perform the various steps presented within this section.
- This procedure must be re-performed if the virtual link configuration is changed, any T9000 nodes are added or removed, or any of the C104 interval labelling scheme is changed. The addition of a single T9000 (that may not even communicate with the T805s) would be almost guaranteed to change all virtual link numbers in the DS link network.
- I require access to all interval labels used in the C104 network. A look-up table needs to be produced which gives the destination processor of any header injected into the network. In addition, I must also know the required header to reach a given processor.
- There are two ‘spy’ programs available, one for OS link networks and one for DS link networks. There is no combined spy for spying mixed OS/DS link networks and testing the connection between the two. There was no software that can verify that links over the C100 are actually connected correctly. A test program had to be developed to verify connections between the OS and DS links.

4.8 On-line Monitoring

The event reconstruction code (CPREAD) produces a standard set of histograms for monitoring purposes. A process on a T9000, the histogram collector, was used to collect these histograms from each worker and communicate them to the workstation host, where the PAW (Physics Analysis Workstation) package was used to view them. A schematic of the overall monitoring system is shown in Figure 4.11.

A PAW session connects to a PAW server to access histograms. The PAW server must use TCP/IP sockets to read histograms from the histogram collector. The software interface between the workstation and the Transputer system does not support Unix sockets. A socket Server program was written to allow communication between the PAW server and the Histogram collector.

As a method of checking the behaviour of the T9000 farm, the acceptance of CPREAD as seen by each processing node was monitored. This acceptance was defined as the fraction of events which passed the selection criteria of CPREAD. The acceptance as a function of the Worker Identifier is shown in Figure 4.12. This approach provided information on the state of an individual worker and an indication of the quality of the data. More detailed monitoring was performed by using the standard set of histograms produced by each worker.

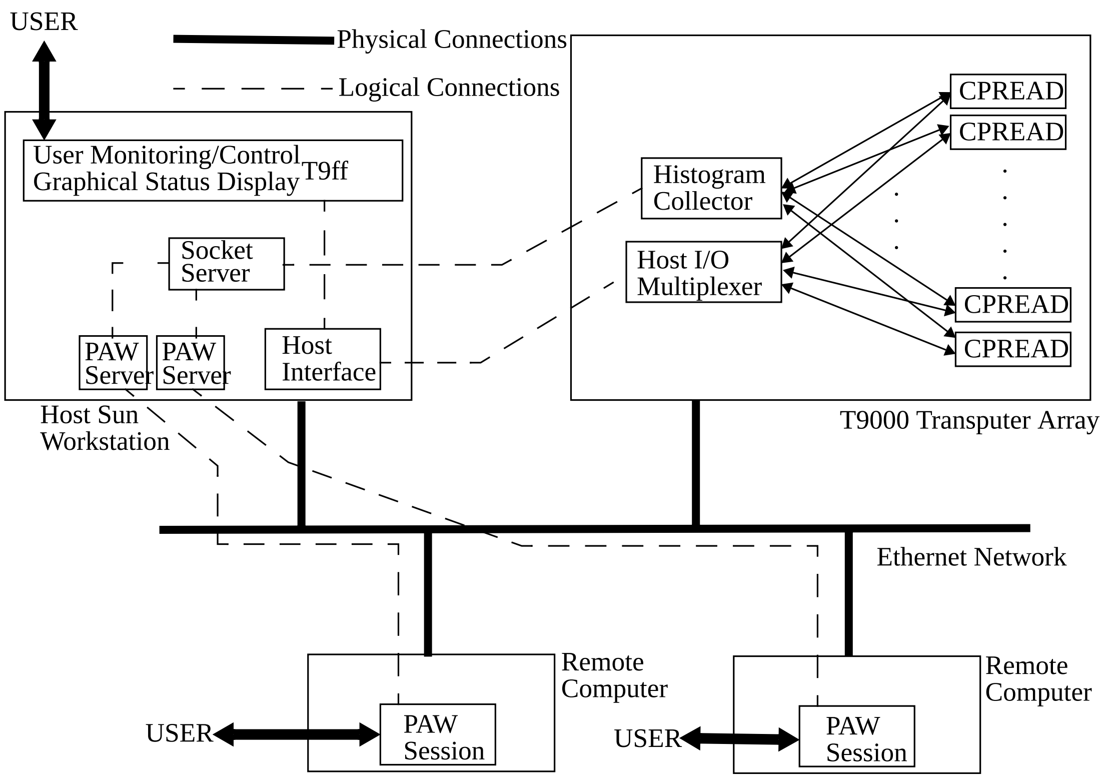


FIGURE 4.11 Monitoring

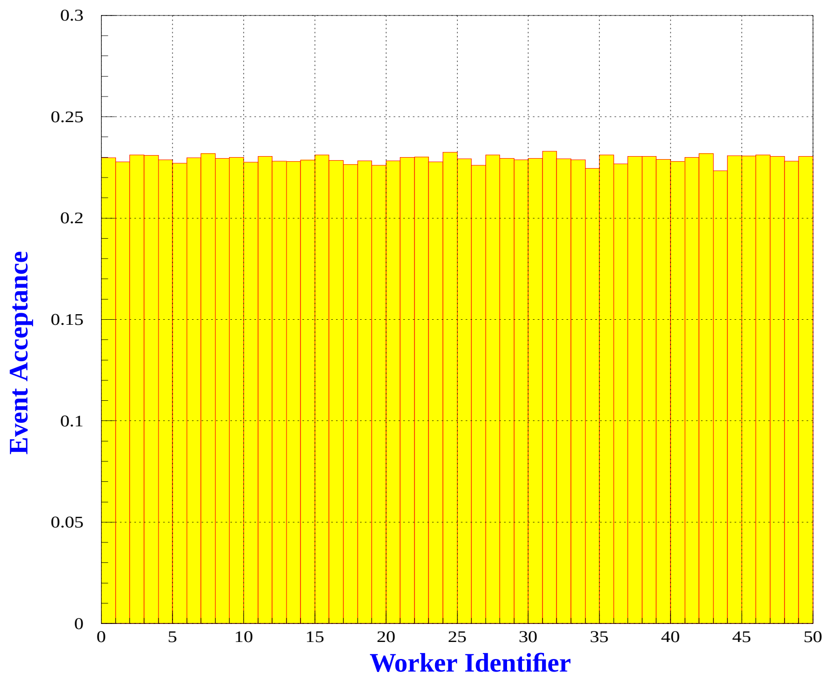
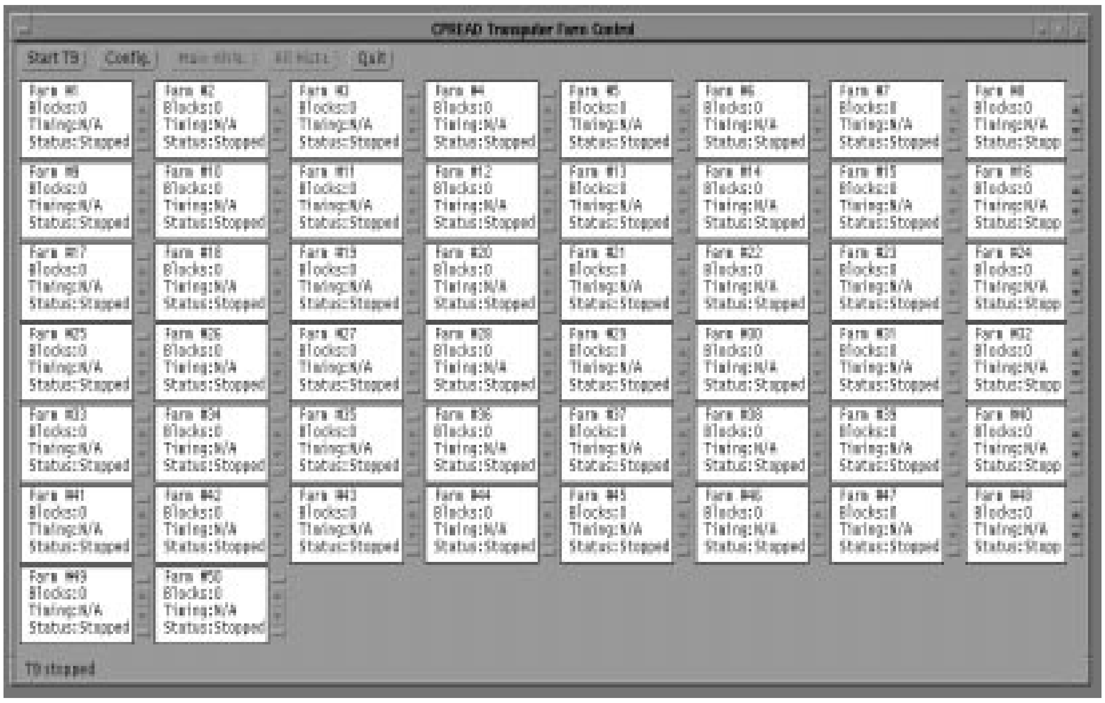


FIGURE 4.12 Acceptance Factor

User control and monitoring for the T9000 system was carried out via a graphical front-end called T9ff which is shown in Figure 4.13.



```
Farm #1          Farm #2          Farm #3          Farm #4          Farm #5          Farm #6          Farm #7          Farm #8
Blocks:0         Blocks:0         Blocks:0         Blocks:0         Blocks:0         Blocks:0         Blocks:0         Blocks:0
Timing:N/A       Timing:N/A       Timing:N/A       Timing:N/A       Timing:N/A       Timing:N/A       Timing:N/A       Timing:N/A
Status:Stopped   Status:Stopped   Status:Stopped   Status:Stopped   Status:Stopped   Status:Stopped   Status:Stopped   Status:Stopp
Farm #9          Farm #10         Farm #11         Farm #12         Farm #13         Farm #14         Farm #15         Farm #16
Blocks:0         Blocks:0         Blocks:0         Blocks:0         Blocks:0         Blocks:0         Blocks:0         Blocks:0
Timing:N/A       Timing:N/A       Timing:N/A       Timing:N/A       Timing:N/A       Timing:N/A       Timing:N/A       Timing:N/A
Status:Stopped   Status:Stopped   Status:Stopped   Status:Stopped   Status:Stopped   Status:Stopped   Status:Stopped   Status:Stopp
Farm #17         Farm #18         Farm #19         Farm #20         Farm #21         Farm #22         Farm #23         Farm #24
Blocks:0         Blocks:0         Blocks:0         Blocks:0         Blocks:0         Blocks:0         Blocks:0         Blocks:0
Timing:N/A       Timing:N/A       Timing:N/A       Timing:N/A       Timing:N/A       Timing:N/A       Timing:N/A       Timing:N/A
Status:Stopped   Status:Stopped   Status:Stopped   Status:Stopped   Status:Stopped   Status:Stopped   Status:Stopped   Status:Stopp
Farm #25         Farm #26         Farm #27         Farm #28         Farm #29         Farm #30         Farm #31         Farm #32
Blocks:0         Blocks:0         Blocks:0         Blocks:0         Blocks:0         Blocks:0         Blocks:0         Blocks:0
Timing:N/A       Timing:N/A       Timing:N/A       Timing:N/A       Timing:N/A       Timing:N/A       Timing:N/A       Timing:N/A
Status:Stopped   Status:Stopped   Status:Stopped   Status:Stopped   Status:Stopped   Status:Stopped   Status:Stopped   Status:Stopp
Farm #33         Farm #34         Farm #35         Farm #36         Farm #37         Farm #38         Farm #39         Farm #40
Blocks:0         Blocks:0         Blocks:0         Blocks:0         Blocks:0         Blocks:0         Blocks:0         Blocks:0
Timing:N/A       Timing:N/A       Timing:N/A       Timing:N/A       Timing:N/A       Timing:N/A       Timing:N/A       Timing:N/A
Status:Stopped   Status:Stopped   Status:Stopped   Status:Stopped   Status:Stopped   Status:Stopped   Status:Stopped   Status:Stopp
Farm #41         Farm #42         Farm #43         Farm #44         Farm #45         Farm #46         Farm #47         Farm #48
Blocks:0         Blocks:0         Blocks:0         Blocks:0         Blocks:0         Blocks:0         Blocks:0         Blocks:0
Timing:N/A       Timing:N/A       Timing:N/A       Timing:N/A       Timing:N/A       Timing:N/A       Timing:N/A       Timing:N/A
Status:Stopped   Status:Stopped   Status:Stopped   Status:Stopped   Status:Stopped   Status:Stopped   Status:Stopped   Status:Stopp
Farm #49         Farm #50
Blocks:0         Blocks:0
Timing:N/A       Timing:N/A
Status:Stopped   Status:Stopped
```

FIGURE 4.13 The Graphical Front End for User Control and Monitoring - T9ff

T9ff ran on the host workstation. T9ff automatically created and killed the PAW and socket server processes at start-up and end of runs respectively. The farm can be loaded, started and PAW sessions started and controlled via T9ff. Due to the use of multiple systems (T805, T9000 and monitoring on workstations) a complex start up system was required. The T805 network was hosted separately using a separate control interface, which was not connected to T9ff. There were three major steps to start the processor farm:

- Configure and load T805 system, then wait until T805 network has received data from experiment and all Exabytes initialised (approximately 3 minutes). This step is performed first to allow gateway T805s to initialise and start waiting for communication from the T9000s.
- Configure and load T9000 system via T9ff, approximately 5 minutes, the delay is due to loading the CPREAD binary (2 Mbytes) and calibration files (4 Mbytes) over the slow host connection. T9ff automatically starts socket servers.
- Run PAW via T9ff, this step must be performed last to ensure that socket servers have started, if a PAW server starts and tries to communicate when the socket server is not present an error will occur.

An end of run or shutdown was performed when Exabyte tapes were full or I wished to stop the system to perform further development work. The shutdown was performed on the T805 first, to leave Exabytes and all external I/O devices in a correct state. Then a T9000 shutdown was performed via T9ff. The T9ff program stopped the T9000 workers via the host interface and then killed all PAW server and socket server processes, to ensure a clean start-up for the next run.

4.9 History of Installations and associated problems

4.9.1 CPLEAR run June 1994

The pre-production prototypes of the T9000 (gamma D02 versions) capable of running large programs were available in small quantities at the start of 1994. A member of the group visited the chip manufacturer to select chips that were able to run CPREAD. A small Parsys SN9400 system, consisting of six T9000s and one C104 packet routing chip, was installed and successfully run in the CPLEAR experiment in June 1994, see Figure 4.14. Due to the low number of available processors the calibration server, histogram collector and host I/O multiplexer all ran on a single T9000. The June run was used to test the various system components, and prepare as stable a system as possible for the next experiment run.

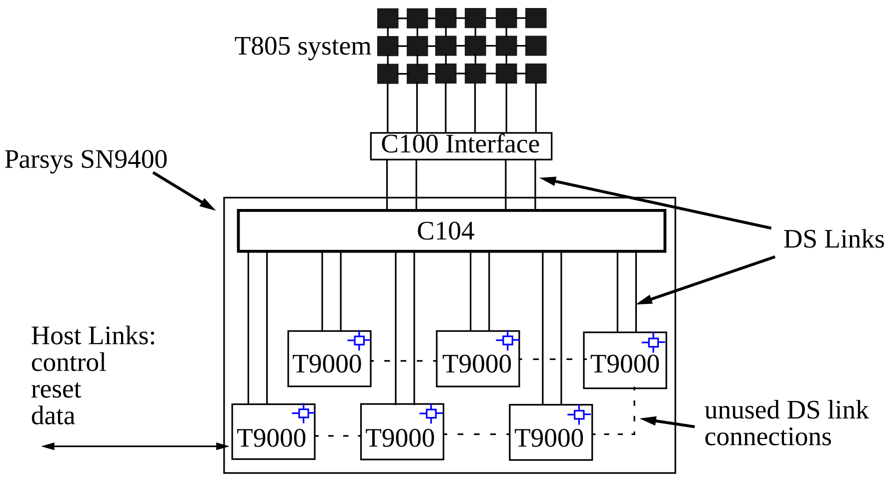


FIGURE 4.14 1994 June run - a single Parsys SN9400

The interface between the T9000 system and the T805 system was in development during the run. Continuous changes were required to both systems. The major limitation when hardware was installed in the CPLEAR experiment was that there was only room for a single workstation. Machines for two persons were required, so work had to be carried out remotely where more workstations were available. Some problems with the T9000 systems required a power cycle (required the power to be turned off and then on again), which could not be performed remotely. Work stopped while someone went to CPLEAR and power cycled the equipment. Development work during experiment running was unavoidable due to the late arrival of T9000s.

The main source of problems was the CPREAD program and errors that had been introduced in porting to the T9000. By the end of the run all CPREAD porting related problems had been solved, and I had a version of the code that was stable for the next experiment run.

The CERN computer service provides software via the ASIS server. Initially in the June run I used this server to obtain all our software for the host workstation, including PAW. During the June run ASIS was updated and PAW was changed to a version that did not work. I obtained the old version and made a local private copy of all software that the workstation used. The system was thereby isolated from any changes made by ASIS and controlled directly the update of software versions.

The June run allowed us to confirm that all components would scale to 64 T9000s at 20 MHz. I analysed the I/O requirements of the CPREAD workers and the rates the gateway T805s could provide. A single 20 MHz T9000 analysed events at 1.67 Hz, and for every ten events read approximately 2 events were accepted and written out. The reconstructed accepted events were a factor two greater in size (the raw event and the reconstructed event were both recorded). The result was that a worker writes out data at half the rate it read in. I knew the event sizes and was therefore able to calculate the bandwidth requirement between a 20 Mhz T9000 worker and a gateway T805. The requirement was 5.8 Kbytes/s.

A single link to a gateway T805 was benchmarked at 79.5 Kbytes/s, uni-directional[11], equivalent to 3.4 Zebra blocks or 36 to 37 events per second. Each gateway T805 link can service 13 T9000s at 20 MHz. If required 12 links could have been connected to the T805 network, corresponding to 156 T9000s, confirming that rates would have scaled past 64 T9000s.

In particular, it was important to measure the ability of the histogram collector to access all of the CPREAD histograms. The histogram collector was a single concentrated communication point for all workers, hence it may have been necessary to collect a subset of the full histogram set. The analysed event rates scaled linearly for one to five workers and the bottleneck was computation in the farm workers.

4.9.2 CPLEAR run September/October 1994

In September 1994, four more SN9400 systems were added to the original CPLEAR set-up making a total of 30 T9000 Transputers. The network is shown in Figure 4.15.

During the summer of 1994, components of the GPMIMD machine became available. A subset of the machine comprising 24 pre-production prototypes of the T9000 nodes (gamma D02 versions) was installed, independently of the SN9400 systems, in October of 1994. The resulting system contained 54 T9000 Transputers (all gamma D02 silicon) and 20 C104 switches (all revision alpha).

The two systems remained separate to maximise the possibility of at least one stable network, i.e. provide fault tolerance such that one network could remain running after the other had failed. Each system used two links to two gateway T805s. Each system had a separate histogram collector, calibration server, T9ff and host I/O multiplexer. A single PAW session collected histograms from a separate PAW server for each network, shown in Figure 4.16. PAW scripts were written to optionally combine the histograms from the two networks, for histogram viewing the two networks appeared to be one.

Figure 4.17 is a photograph of the T9000 system and Figure 4.18 is a key describing the various components of the photograph.

11.The gateway T805 links could only be used uni-directional, and communicate on a single virtual link at any one time. This was an implementation limit on the gateway T805 software.

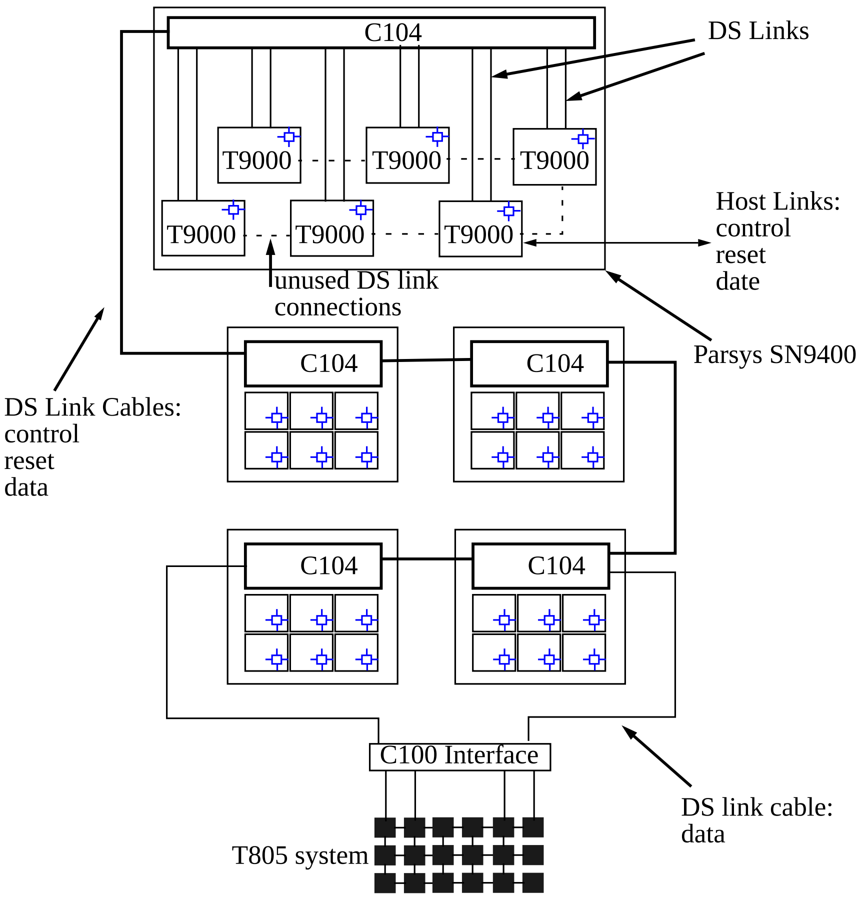


FIGURE 4.15 30 T9000 network of SN9400s

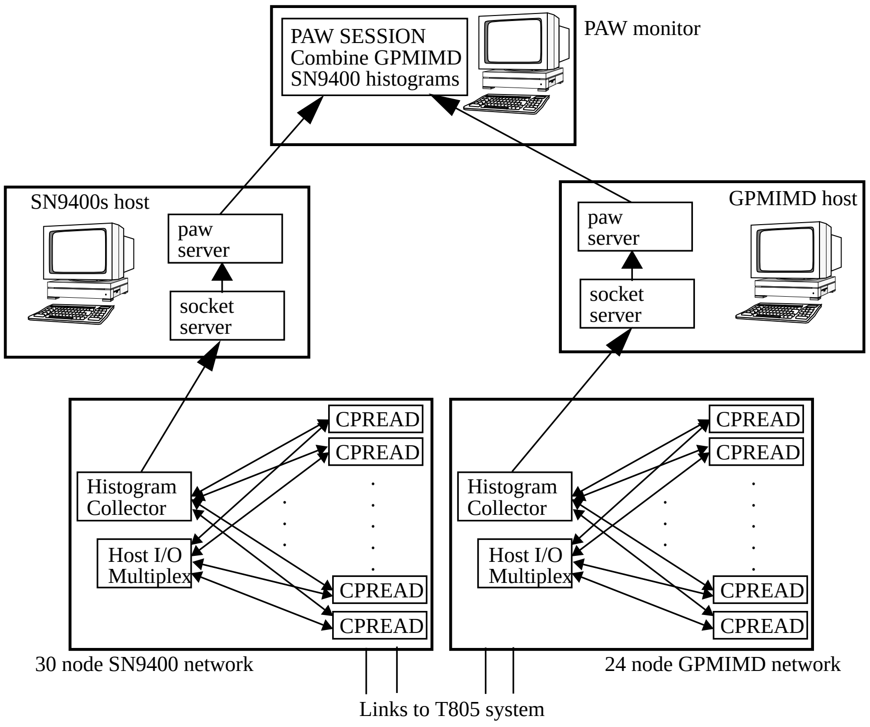


FIGURE 4.16 Final 54 T9000 PAW configuration

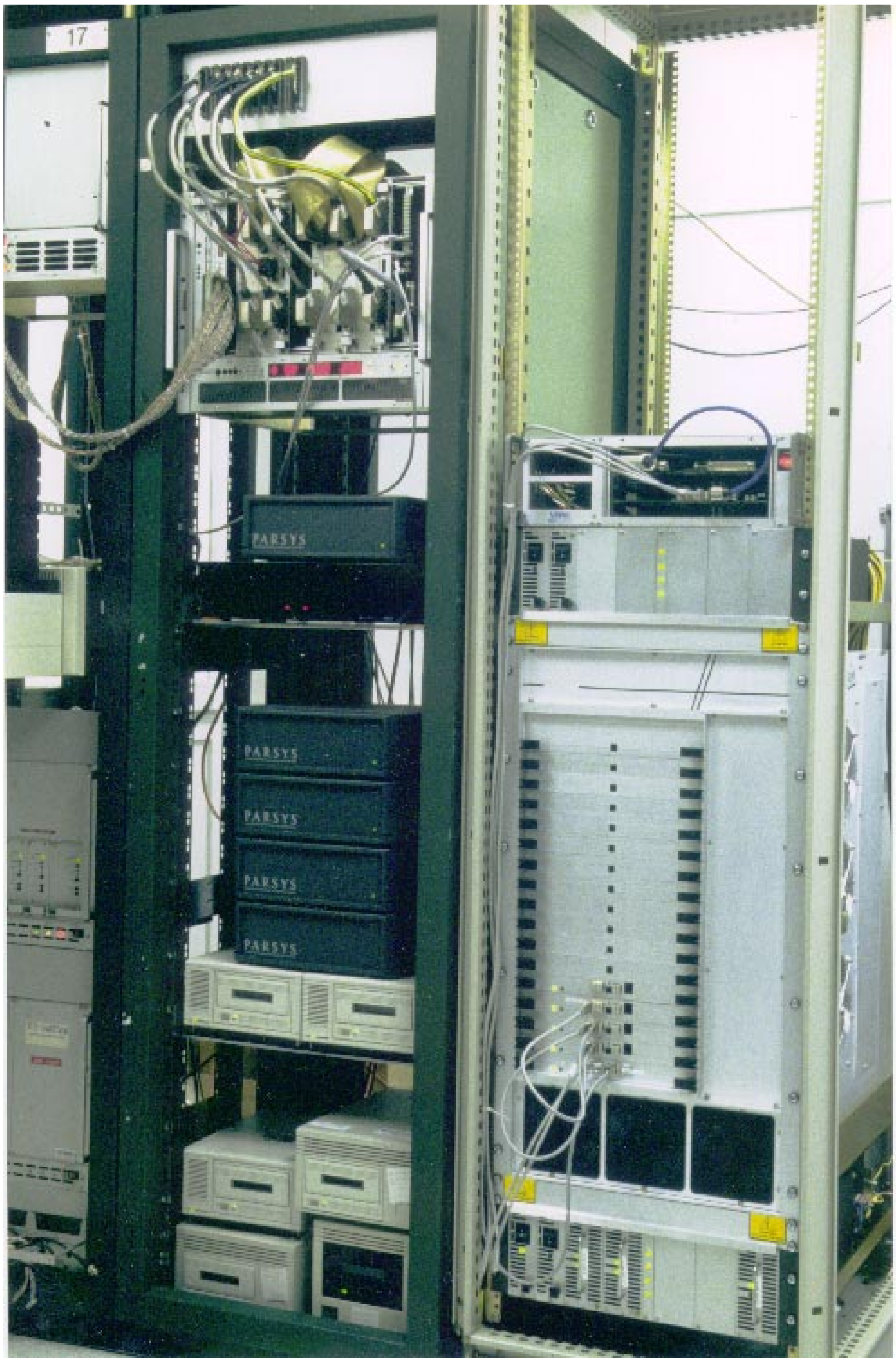


FIGURE 4.17 The T9000 system in CPLEAR, see Figure 4.18 for description of components

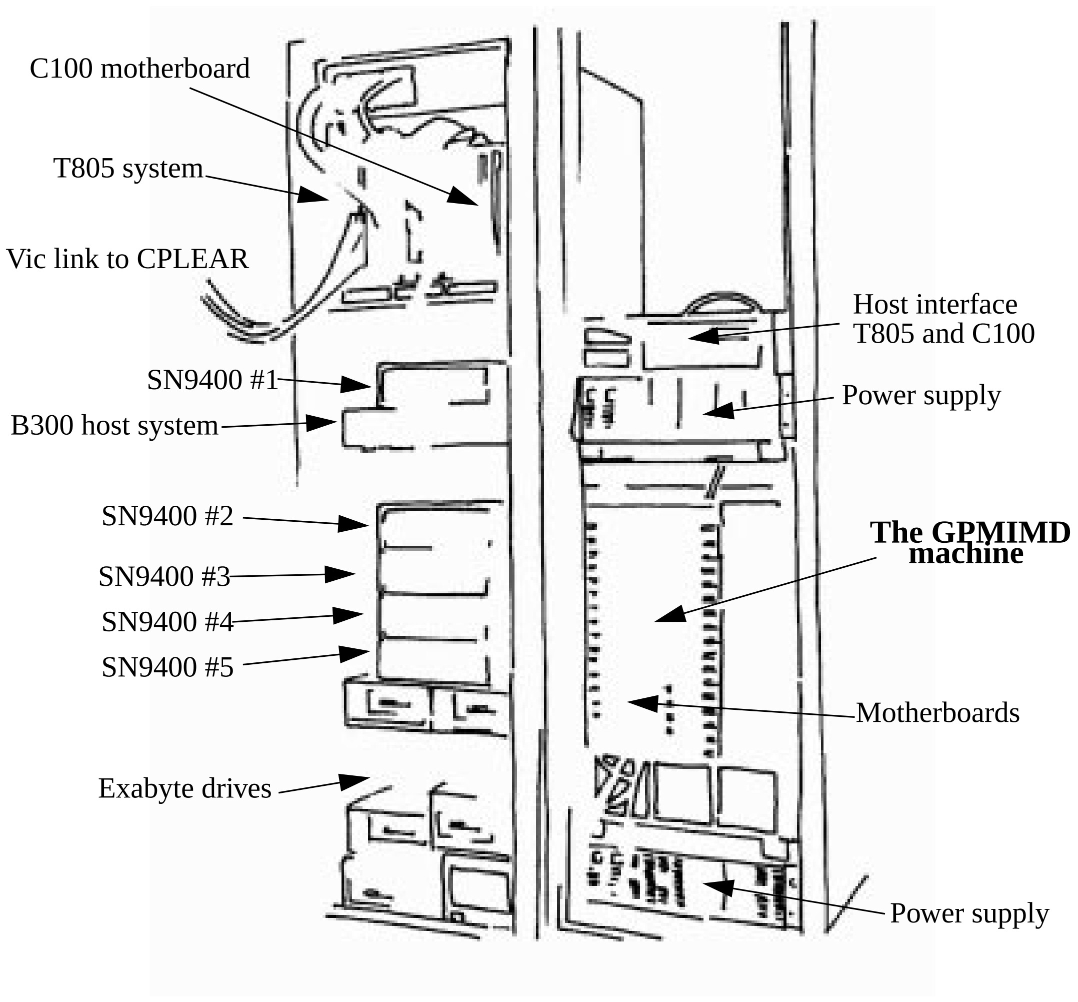


FIGURE 4.18 The Key to Figure 4.17

4.9.3 Practical problems and solutions

The T9000 REV D silicon suffered from multiple serious hardware bugs. The two which most affected the work are the double scheduling bug and the cache corruption bug. Both of these bugs occurred during communication. The problems caused by these two bugs, and the general problems encountered during the run are presented in the following sections.

4.9.3.1 Problems arising from the double scheduling bug

The bug was triggered when the final acknowledge packet was received before the last packet of the message had completely left the T9000. If this occurred, the communicating process was re-scheduled twice, once by the incoming final acknowledge and then again when the final packet left the T9000. This can cause a CPU error. All methods suggested to avoid the bug were based on increasing the probability that the final message packet left the T9000 before the acknowledge for it had arrived.

The bug is was most likely to occur for directly connected T9000s, where there was no C104 to add latency before the destination could produce the acknowledge. The simplest and most effective method was to ensure that the last packet of a message was very short. This ensured that the last packet was sent out quickly, and before the acknowledge could arrive.

None of these methods could guarantee total reliability for the CPLEAR application, as host I/O libraries did not allow the user to control the length of messages sent to the host.

Despite these limitations, all precautions possible were taken to try to avoid the bug. No direct links were used on any T9000 systems, in particular the Parsys SN9400 systems did not use any of their direct links, always communicating via the C104. In addition, where ever possible, restrictions were imposed on the message lengths that could be used. All messages had to be an exact multiple of 32 bytes plus an extra single byte. For example, the CPREAD workers read events in Zebra blocks, which are normally 23,040 bytes. For the CPREAD workers on the T9000s the Zebra blocks were extended to 23,041 bytes. Similar alterations were made in all software where it was possible. However, the host I/O libraries allowed no control on the message sizes of packets to the host, therefore this remained a possible source of instability.

4.9.3.2 Problems arising from the cache corruption bug

The cache corruption bug only occurs when the T9000 is performing inputs on two or more physical links in parallel, and the data is written to cacheable memory. The corruption usually occurs in the part of the cache line that follows the message, but words within a message can also be corrupted.

The bug will not occur if the cache is disabled. However, there were two major problems in not using the cache:

- The 64 bit memory interface on the 30 SN9400 T9000 HTRAMS would not allow the cache to be disabled. Altering the HTRAMS to 32 bit memory interfaces would have required hardware modifications. The 24 T9000s in the GPMIMD machine had 32 bit interfaces.
- The loss of the cache would have reduced the performance of the T9000 by a factor 4.

Other solutions were generally based on the use of checksums or sending multiple copies of the data. All methods of this type reduce communication performance and do not guarantee to make the T9000 stable when multiple links are in use with cached memory.

The result was that the hardware was completely unstable if more than one physical link of a T9000 was in use at any one time. The solution for the application to CPLEAR was where ever possible to only use one physical link of the T9000. This was possible for 53 of the 54 T9000s, the exception was the root T9000 [12] in the Parsys SN9400 system. This T9000 was swapped with a 32 bit memory interface T9000 from the GPMIMD machine, allowing the cache to be disabled. This node operated as a host I/O multiplexer and as such did not suffer from the reduced performance due to disabling the cache.

12.The root T9000 (connected to the host) of the SN9400 system must use at least two links: one to the host and a second into the C104 to connect to the other T9000s. The host link on the GPMIMD machine enters via a C104, so all T9000s can use a single link.

The communication performance of a single DS link was sufficient to provide raw data to, and read back reconstructed data from a T9000 running at 20 MHz. This approach completely eliminated the cache corruption bug.

4.9.3.3 Reset problems due to multiple networks

The Parsys SN9400 systems and the GPMIMD machine were independent, which meant there were two separate host systems. The result was that there was a separate reset for the two systems. The reset to a DS link system must be a common reset for all devices in that system. The C100 motherboard originally only had a single reset, so it had to be rewired to allow separate resets of each of the C100s for each system (SN9400s or GPMIMD machine), see Figure 4.19.

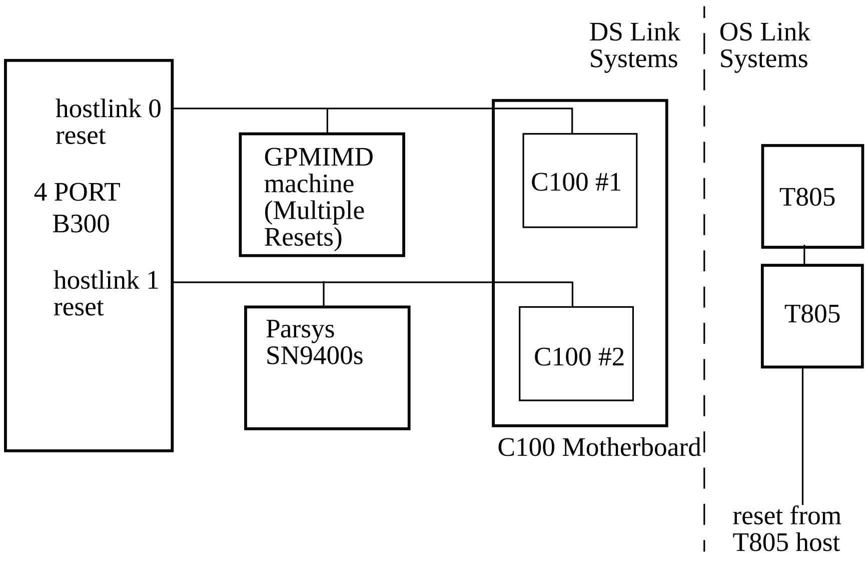


FIGURE 4.19 Reset System

4.9.3.4 Late delivery of GPMIMD machine

Due to late delivery, the GPMIMD machine was placed into the experiment during the September run, and the alterations to the C100 motherboard were also made during the run. This meant that the SN9400 systems were not actually tested with the GPMIMD machine in parallel until during the experiment run. The T805 to T9000 interface also needed to be tested during the experiment run.

4.9.3.5 GPMIMD machine problems

Before I could use the GPMIMD machine in the experiment there were various problems that needed to be solved. This was unfortunate considering that the late arrival of the machine meant that the experiment run had already started.

The version of the GPMIMD machine installed in the experiment is shown in Figure 4.20. There were no switch cards, all inter-board communications had to be performed via external cables between the motherboards.

This figure should be compared to Figure 2.19 on page 26, which shows the topology of the machine I had expected[13]. All hardware description files had to be re-written to accommodate this change. Two external links were connected to the T805 system. The backplane was used to connect the control chain between the motherboards.

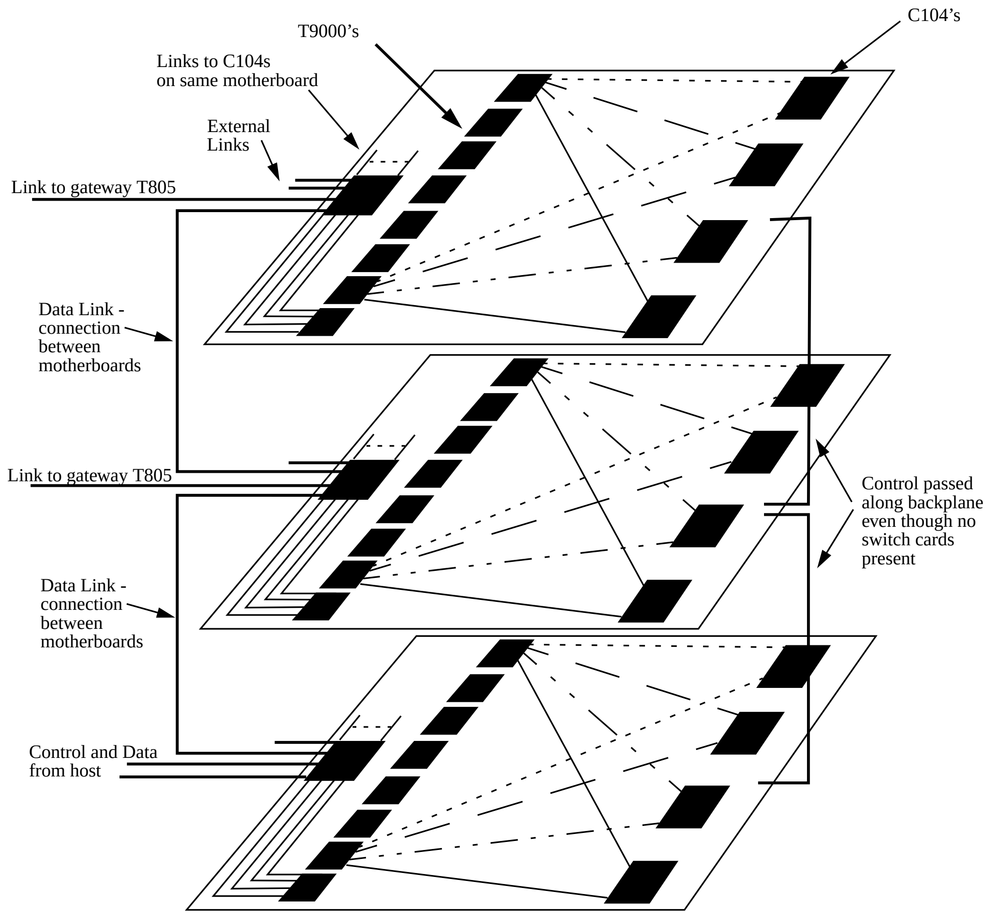

C104's
T9000's
Links to C104s on same motherboard
External Links
Link to gateway T805
Data Link - connection between motherboards
Link to gateway T805
Control passed along backplane even though no switch cards present
Data Link - connection between motherboards
Control and Data from host

FIGURE 4.20 3 motherboard GPMIMD machine for CPLEAR run September/October 1994

Multiple external links were not working, through a process of trial and error I uncovered all the operating links. A spy program to discover the topology of a network and tests all connections existed, but would not run on the GPMIMD machine. The spy program was produced by the chip manufacturer, it was unsupported and appeared untested.

T9000s in the GPMIMD machine would not run for greater than a few minutes at 20 MHz. I was reluctant to reduce the speed since all the processors ran at 20 MHz in SN9400 systems, there was finally no choice, and all T9000s in the GPMIMD machine ran at 10 MHz.

A 4th motherboard was unusable, and efforts to repair this board were abandoned in favour of getting the other 3 motherboards installed into the experiment.

13.At the time of writing the full machine topology, including four switch cards, 64 revision E03 T9000s and 58 beta C104s has been delivered to CERN.

The GPMIMD machine required multiple resets to avoid an approximate 10% failure rate that would occur when only a single reset was applied to the machine before loading. The problem still exists on the current GPMIMD machine installed at CERN. The latest revision of the machine includes hardware to perform multiple resets after a power up. There were also occasions (approximately every 48 hours) when the machine needed a full power cycle before loading could be successful.

4.9.3.6 Limited access to the C100 status

The host software did not support the C100 as part of the DS link control chain, the C100 could not be included in the hardware description file, even though the C100 has two control links (the same as the T9000). The result was that there was no access to the status of the C100s and errors on the C100 could not be reported on a control chain. If an error occurred on a C104 link connected to a C100, the error was only reported from the C104. If communication over the C100 failed, and there was no error from the T9000/C104 connected to it, the user had no information to diagnose the problem and no way to interrogate the status of the C100.

The C100 motherboard could hold 4 daughter boards, each daughter board with a single C100, giving a total of 16 OS links converted to 16 DS links. After continual failure to use 4 of these 16 links the board was returned to the lab for tests. A design fault was found, C100 link 3 was not working on all of the daughter boards. The result was that a maximum of 12 links were available between the T805 and T9000 systems.

4.9.3.7 Electromagnetic noise problems

Many of the T9000/C104 systems that I have used suffered from succeptability to electromagnetic noise. At the start of the September 1994 run I had hoped that these problems had been solved. I had placed copper foil between the casing of the GPMIMD machine and the external cable shields, and used copper foil to try to close off open VME boards. These actions were to provide a short return path to ground for any noise coupled onto the cable shields, and reduce the amount of electromagnetic noise entering the equipment. Despite these efforts the C100 motherboard proved to be a strong source of unreliability during this experiment run.

On repeated occasions (see Table 4.1) during the experiment run there were link errors reported from C104 links connected to the C100 motherboard. On at least one occasion I noticed that these failures occurred when the air-conditioning had automatically powered on. During the experiment run a new version of the C100 motherboard became available that connected the shields of the DS link cables to signal ground. Again this is to provide a short return path to ground for noise induced onto the cable shields. Previously the connection to ground from the cable shields was over a single printed circuit board track. After the new board was installed in place of the old, there were no further failures related to noise.

4.9.3.8 Summary of shift log from september/october run 1994

Table 4.1 is an outline of the experiment shift log during the CPLEAR running during the months of September/October 1994. Each end of run is detailed, and if problems caused the end of run then the final column details the problem. Additional errors occurred due to the T805 system and/or the actual CPLEAR experiment, these errors have been omitted.

There are two lengths given for each run, the first is the total run time, which is the time elapsed between problems on the T9000 system. A new run number is started after a failure on the T9000 system or Exabyte tapes need changing. The second length is the time which the T9000 system was actually running and taking data during this run. In each run there were T805 failures and time during which the experiment did not produce data, this accounts for the difference in the two times. 285 hours out of a possible 415.5 hours were available for data taking.

TABLE 4.1 Summary of September/October run 1994

Run Number and Length	Time of T9000 operation	Start Time	End Time	Reason For End of Run and network description
1 25 hrs.	24 hrs.	11:00 22/9	12:00 23/9	C100 board noise succeptability 30 node SN9400s network
2 168 hrs.	65 hrs.	13:00 23/9	12:00 30/9	stopped by user - change tapes 4 T805 Failures during run allowing tape changes. 30 node SN9400s network
3 2.5 hrs.	2.5 hrs.	12:00 30/9	14.30 30/9	C100 board noise succeptability 30 node SN9400s network
4 13 hrs.	13 hrs.	18:00 30/9	07:24 1/10	T9000 Host failure SN9400s network
5 9.5 hrs.	9.5 hrs.	08:00 1/10	17:30 1/10	C100 board noise succeptability 30 node SN9400s network
6 14 hrs.	14 hrs.	18:00 1/10	08:00 2/10	T9000 Host failure 30 node SN9400s network
7 29 hrs.	29 hrs.	09:00 2/10	14:00 3/10	stopped by user - change hardware 2 T805 failures during run allowed tape changes NEW C100 Motherboard introduced after run 30 node SN9400s network
8 40.5 hrs.	40.5 hrs.	18:00 3/10	10:30 5/10	stopped by user - change tapes RUN USED 24 node GPMIMD MACHINE ONLY
9 49 hrs.	38.5 hrs.	15:00 5/10	16:00 7/10	stopped by user - change tapes 1 T805 failure during run allowed tape changes Combined 54 node SN9400s and GPMIMD network

Run Number and Length	Time of T9000 operation	Start Time	End Time	Reason For End of Run and network description
10 48 hrs.	32.5 hrs.	16:00 7/10	16:00 9/10	stopped by user - change tapes 1 T805 failure during run allowed tape changes Combined 54 node SN9400s and GPMIMD network
11 17 hrs.	17 hrs.	16:00 9/10	09:00 10/10	stopped by user - end of CPLEAR running Combined 54 node SN9400s and GPMIMD network

4.10 Results

The results reported are based on a three-week run in September/October 1994. By the end of this period two T9000 processor farms had been combined into a 50-node processing farm which processed events at a rate of 65 Hz. Table 4.2 summarises the performance of the two networks, the results are consistent with a single event reconstruction time of 0.6 seconds on a 20 MHz T9000. The I/O requirement between a single 20 MHz worker and the T805 system, presented previously, is 5.8 Kbytes/s. A 10 MHz worker analyses data at half the rate, so requires 2.8 Kbytes/s. For both networks two links to the gateway T805s were adequate. If more than 28 workers at 20 MHz were in the SN94000 networks more links to the T805 network would have been required

The ability of the network to provide raw data, read back reconstructed data, connect to host, connect to calibration server and histogram server has not introduced a bottleneck. All I/O requirements have scaled with the addition of processors. The analysis rates have scaled linearly with the number of processors used, the computation bottleneck remaining.

TABLE 4.2 Event Rates

	Network description	Event rate	I/O requirement to T805 network	Time for single event on each T9000
SN9400s	28 T9000s at 20MHz	46 Hz	162 Kbytes/s	0.6 seconds
GPMIMD	22 T9000s at 10 MHz	19 Hz	61 Kbytes/s	1.2 seconds

Figure 4.21 shows the number of events analysed as a function of the number of hours on run. The time is the actual time the T9000 farm was operating. The points correspond to the end of runs shown in Table 4.1. The increase in reliability after the C100 motherboard change can be clearly seen in the spacing of the graph points. The reliability was improved despite the increased complexity of the system (combined GPMIMD and SN9400s). The

increased gradient corresponds to both the GPMIMD machine and SN9400 network used in parallel.

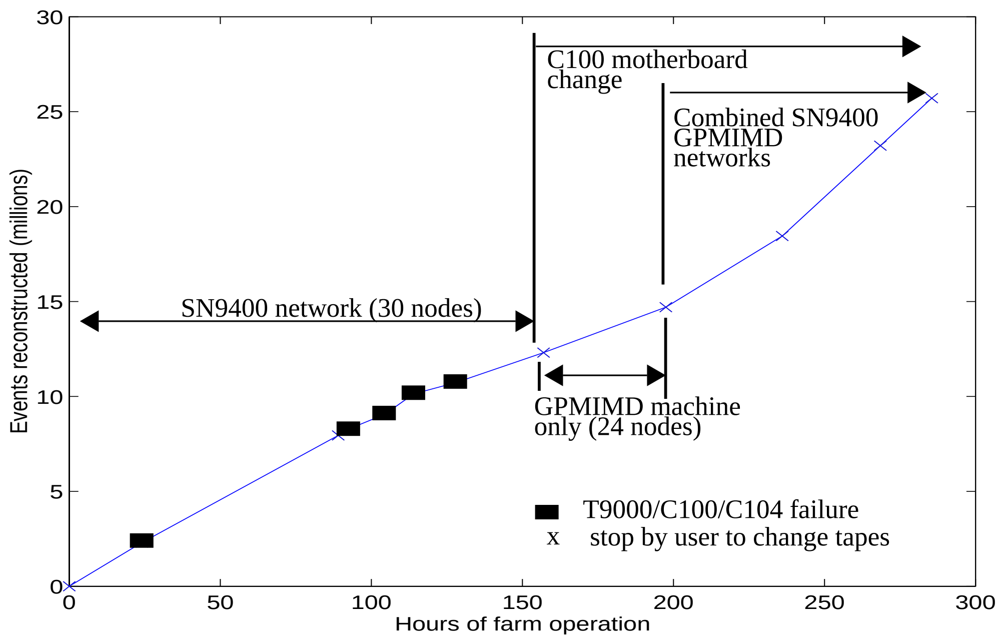


FIGURE 4.21 Number of events analysed and failures throughout experiment run

For the last 128 hours of the run a stable platform was maintained and no failures of the network were observed. During a total running time of 285 hours, 26 million events were processed. The experiment run lasted for 415.5 hours, 70% of run time was active data taking. The initial and main source of problems was the susceptibility of the DS links to noise, which were resolved with suitable hardware modifications.

Samples of the histograms produced by CPREAD were recorded (see Figure 4.22) and all output from the farm was written to Exabyte. The results have been verified against the standard CPLEAR off-line program. The first plot of Figure 4.22 'NBRE DE evts' shows the number of events passing each stage of the rejection and filtering process, known as the number of events plot. A different criteria is applied for each entry and results in more events being rejected. The event acceptance plot presented in the monitoring section can also be seen in Figure 4.22 with the title 'Acceptance factor'. The acceptance factor can be derived from the number of events plot, it is the last entry as a fraction of the first entry, approximately 20%.

The feasibility of a T9000 farm processing the full CPLEAR event rate (approx. several hundred Hz) was based on measurements made on the T805 system implemented in 1993. The 50 MHz T9000 was expected to be at least a factor of ten more performative than a 20 MHz T805 and a 64-node 50 MHz T9000 farm should have processed events at about 400 Hz. However, the current prototype T9000 performance falls short of its design goals and only a factor of four improvement over the T805 instead of the expected factor of ten has been measured.

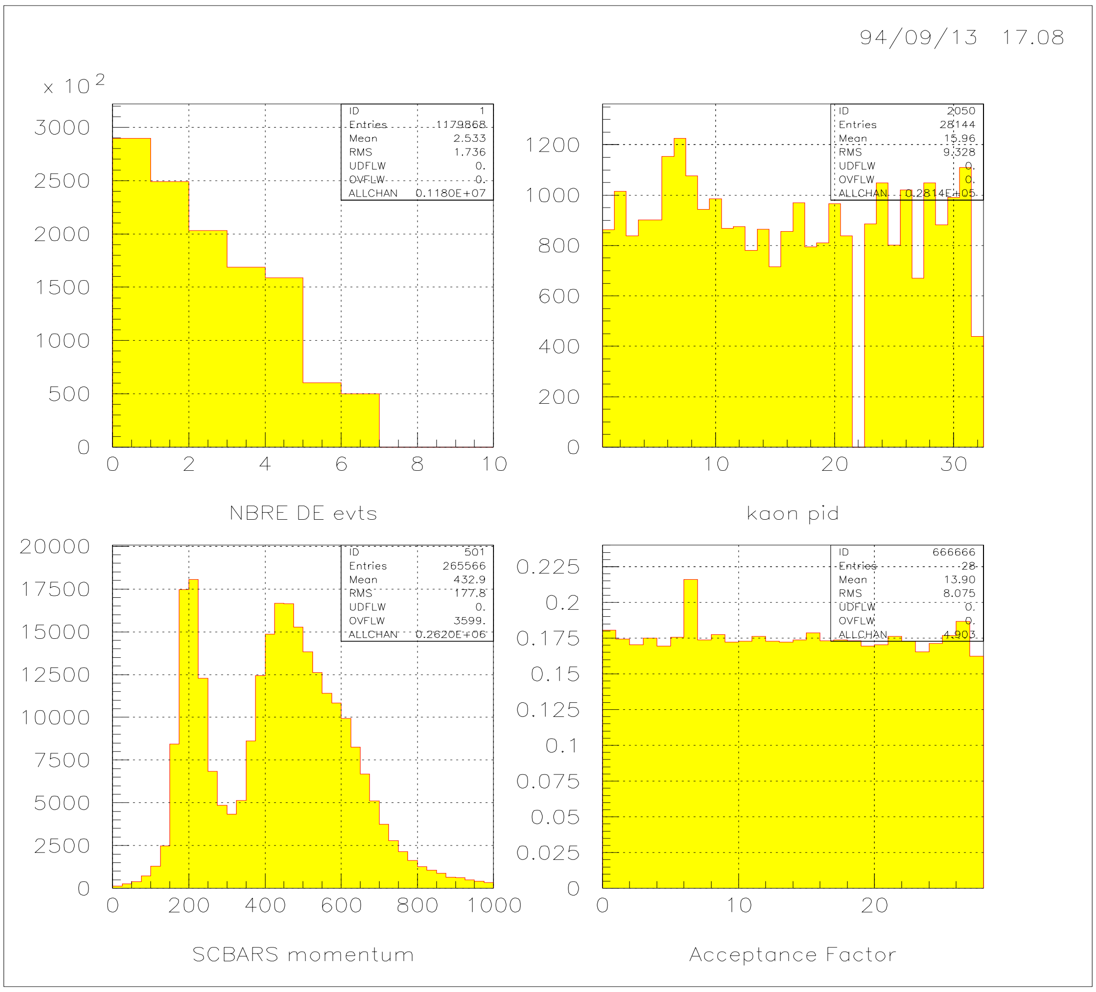


FIGURE 4.22 Example CPREAD histograms

4.11 Boosting computational power using TransAlpha modules

To analyse the current CPLEAR rate of 450 Hz it would require 274 T9000s running at 20 MHz. This is simply beyond the capacity of the GPMIMD machine. The bottleneck is the computational power of the T9000. In particular:

- The clock speed is 20 MHz and not 50 MHz (in some cases only 10 MHz)
- Many floating point operations take three cycles instead of two
- Some trigonometric functions run three times slower than expected
- The lack of a native Fortran compiler necessitates the use of f2c, which on a T805 and a SUN Sparc station gives a performance penalty equal to a factor of 1.5.

A system using the TransAlpha modules would restore the balance of communication and computation. For this reason a Particle Physics and Astronomy Research Council (PPARC) project was created to continue the work within CPLEAR. Replacing T9000 nodes with

TransAlpha nodes provided higher processing power and better cost versus price performance.

Initially the aim of the project was to implement a test system based on two T9000 Transputers and two TransAlpha modules running CPREAD. To simulate the CPLEAR environment, two T9000s provided raw event data to the two TransAlphas. The T9000s read raw event data from the host file system, which had been recorded by the T805 system during a previous CPLEAR run. The on-line histograms facilities via PAW were also implemented. The same method of porting CPREAD (via f2c) was employed.

A host I/O multiplexer allowed all the workers to output status information to the host workstation. The TransAlpha modules employed the same protocol to request raw event data from the T9000s as a T9000 worker used to request data from the T805s in the on-line experiment version. The TransAlphas also communicated with the histogram collector in the same way that CPREAD workers communicated with the histogram collector in the on-line experiment version. The result is that both the software and hardware (both use the HTRAM format) would allow us to unplug a T9000 and replace it with a TransAlpha module.

The two node TransAlpha system reconstructed events at a rate of 36 Hz. This is more than a factor 10 improvement in processor performance over the T9000 farm. The communications bandwidths required were well within the performance of a DS link, the requirement for each TransAlpha worker was only 36 Kilobytes/s. The system was tested for 48 hours with no failures, and processed over two million events.

With this performance it would be possible to construct a complete on-line event processing farm using 25 TransAlpha nodes mounted directly into the GPMIMD machine. The maximum capacity of the GPMIMD machine would be 64 TransAlpha nodes.

4.12 Summary

The aims of the work were detailed in Section 4.1, “Motivation,”. There were three main aims:

- Show feasibility of reconstructing full CPLEAR event rate on-line.
- Proof of existence and successful operation of the T9000 and C104 in an experimental environment.
- To obtain information for data acquisition at the LHC.

The limited computational power of the T9000 means that a 64 node 20 MHz T9000 system can only analyse a fraction of the full event rate. I have shown that networks of TransAlpha modules could be used to reconstruct the full event rate on-line.

Despite the prototype nature of the revision D02 T9000 and the known hardware bugs, some of them severe, I have been able to install and operate a network of 54 T9000s and 20 C104s. The system has been operated reliably at the CPLEAR experiment, running a very large event reconstruction program in real-time. Proof of existence, successful and reliable operation of large T9000 and C104 systems has been achieved. The construction of a substantial array of processors operating in an experimental environment uncovered problems that would have never been found for ‘small’ laboratory test systems. The problems that have been presented will be typical for future experiments at the LHC, for example, interfacing different link technologies, the succeptability of high speed links to electromagnetic noise and monitoring large distributed systems.

The work into the application of the T9000 and C104 to future generation HEP experiments is now being carried out as a continuation of this work. The work and its results form the next chapter of the thesis.

All three aims of the work in CPLEAR have thus been achieved.

The reliability of the D02 T9000 in the CPLEAR experiment has given a worst case result. The Gamma E03 T9000 and revision beta C104 are now commercially available and will improve reliability and more importantly remove the restrictions imposed by the D02 bugs. It is very important that application developers in general do not now have to implement the extreme workarounds I have made to avoid the T9000 hardware bugs. Initial tests with the GPMIMD machine upgraded to Gamma E03 and C104 revision beta suggest the problems of stability and reliability have been solved by the availability of the Gamma E03.

The L3 experiment [21] at CERN has installed a network of two C104 routers (initially revision alpha) interconnecting 48 T9000s (initially revision D02) which is operating reliably as a second level trigger. The L3 system has now been upgraded to beta C104s and revision E03 T9000s. Our work in CPLEAR and general experience with the T9000 and C104 was also of considerable use to L3 in the development of this system [22].

The new communication system using DS links and C104 switches shows considerable promise. The C104 in particular offers high density cost-effective commodity communications, which can be used to build very large switching networks. The full CPLEAR rate of approximately 1 Mbyte/s is a small fraction of the capacity of a single DS link.

Chapter 5
DS link technology applied to the LHC

In this chapter I investigate the extent to which DS link technology can satisfy the requirements of the Atlas experiment trigger levels 2 and 3. In Section 5.2, “Atlas,” an introduction to the Atlas experiment is given and the data acquisition system and requirements are presented. All information and requirements presented within the Atlas section are based on the Atlas technical proposal [7], and as such are not the work of the author. I am essentially investigating the extent to which DS link technology can satisfy the requirements of the Atlas experiment as presented within the Atlas technical proposal.

In Section 5.3, “GPMIMD machine applied to level 2 and level 3,” results are presented from mapping the Atlas level 2 and 3 triggers onto the GPMIMD machine. The extent to which DS link technology can meet the requirements presented in the Atlas technical proposal are assessed by extrapolating results to an Atlas subdetector.

A set of standard and Atlas specific communication benchmarks have been defined [23], in Section 5.4, “Comparison with other parallel platforms,” results are presented for the GPMIMD machine and compared to results on other platforms.

I have also identified a set of factors that will have a crucial effect on the performance of any future triggering systems using arrays of processors interconnected by point to point links and switches. These factors and conclusions are presented within Section 5.5, “Conclusions,”. Factors which I believe are important but have not yet been investigated are presented in Section 5.6, “Future Work,”.

5.1 Introduction

The bandwidth and connectivity requirements of trigger and data-acquisition systems for future HEP experiments at the CERN Large Hadron Collider (LHC), highlight the limitations in scaling multi-processor bus-based systems. The prominently featured alternative in the proposed systems is large switching networks using point to point links.

The performance of large switching fabrics has been studied within the context of message passing multi-processor computers [24]. However, these studies were based on telecommunications traffic patterns which differ from those expected in the triggering systems of HEP experiments. The modelling of switching networks under HEP traffic patterns is being performed by several groups [25]. The results presented in this chapter are actual performance measurements using the C104 and T9000. The motivation for the work was:

- Discover the extent to which currently available technology can meet the requirements of future generation experiments

- Define and investigate the crucial factors effecting the performance of multi-processor systems using point to point links and switches
- Compare DS links and the GPMIMD machine to alternative technologies and platforms

5.2 Atlas

All information and requirements for the Atlas detector presented within this section are based on the Atlas technical proposal.

5.2.1 Overview

Atlas is one of the approved experiments at the LHC. The construction work on the LHC machine will start in 1998. At present, the first experiments are scheduled for 2005. Studies are already investigating the structure and feasibility of experiments at the extremely high energies attained in LHC.

The detector consists of a number of subdetectors situated in concentric layers around the collision area. Each subdetector measures certain aspects or certain types of particles. The Atlas trigger system is organized into three levels. At each level, events can be rejected or accepted and passed on to the next level, triggering is the rejection/acceptance of events at these various points.

The level one trigger consists of purpose-built hardware which must reduce the event frequency from ~40 MHz to ~100 KHz. The level two trigger should reduce the event frequency further to ~1 KHz. The level three trigger should reduce the event frequency to 10-100 Hz, which can then be recorded by the data storage system. I have investigated the application of the T9000 and C104 to the second and third level triggers (level 2 and level 3).

An event selected by level 1 consists of multiple regions of interest (RoIs). An RoI corresponds to a physical area of the detector containing useful data generated by a particle. A single event will have variable numbers of RoIs, with an average of ~5 RoIs per event. The level 1 trigger passes information identifying the RoIs to the level 2 trigger. If the event is accepted by the level 2 trigger then all data from the event is passed to the level 3 trigger.

An RoI is defined as a cone starting from the collision area. For each RoI, data is present in several subdetectors. This is shown in Figure 5.1. The selection of subdetectors which contain data for an RoI depend on the particle type producing the RoI.

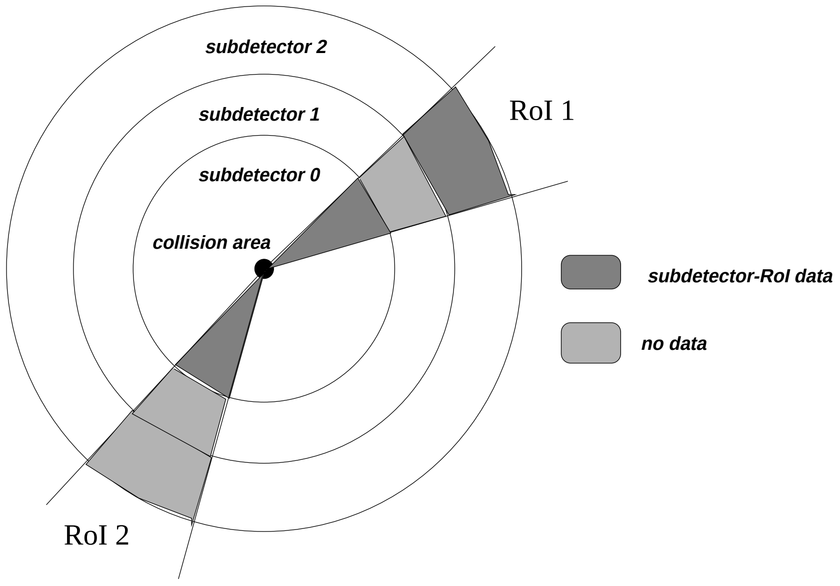


FIGURE 5.1 Two RoIs for a single event spanning several sub-detectors

5.2.2 The level 2 trigger

The level 2 trigger system only processes RoI data which is a small sub-set of the total data contained in an event. During this process the whole event (across all sub-detectors) is stored in Read Out Buffers (ROBs). It is expected that there will be order 1000 ROBs. ROBs must store this data throughout the decision latency for the event. The decision latency for the event is the time between the arrival of the event from level 1 trigger and the accept/reject decision from the level 2 trigger. If an event is accepted then all the event data is passed from the ROBs to the processors of the level 3 trigger system. If an event is rejected then all data contained in the ROBs is discarded.

At any point in time, each ROB holds event data corresponding to a fixed region of a subdetector (an event fragment), but for many events. The number of regions (buffers per subdetector) depends on the segmentation of the electronics connected to the subdetector. The number of buffered event fragments depends on the event frequency from the level 1 trigger and the level 2 decision latency.

The event rate into level 2 will be up to 100 KHz, which should be reduced to 1 KHz. The time between two events (~10 microseconds) is much smaller than the decision latency, therefore parallel processing is required. Multiple events must be analysed concurrently. The throughput of the level 2 trigger is expected to be approximately 10 Gbytes/s, assuming 10% of the total data for an event is analysed. A possible approach (local-global) is shown in Figure 5.2. In the local processing phase, physical quantities and geometric information called features are computed for each subdetector-RoI. A subdetector RoI is all the data from a single RoI originating from one subdetector, the total data for an RoI may span several subdetectors. The subdetector RoI data is collected together into a single feature extractor (FEX) processor. Different feature extraction algorithms are used for different sub-detectors.

All FEX computations are farmed on a set of general-purpose microprocessors. After feature extraction, data is passed through a network to a set of processors which perform the “global processing”. In the global processing phase, all features from one event are combined to make an accept/reject decision for the event.

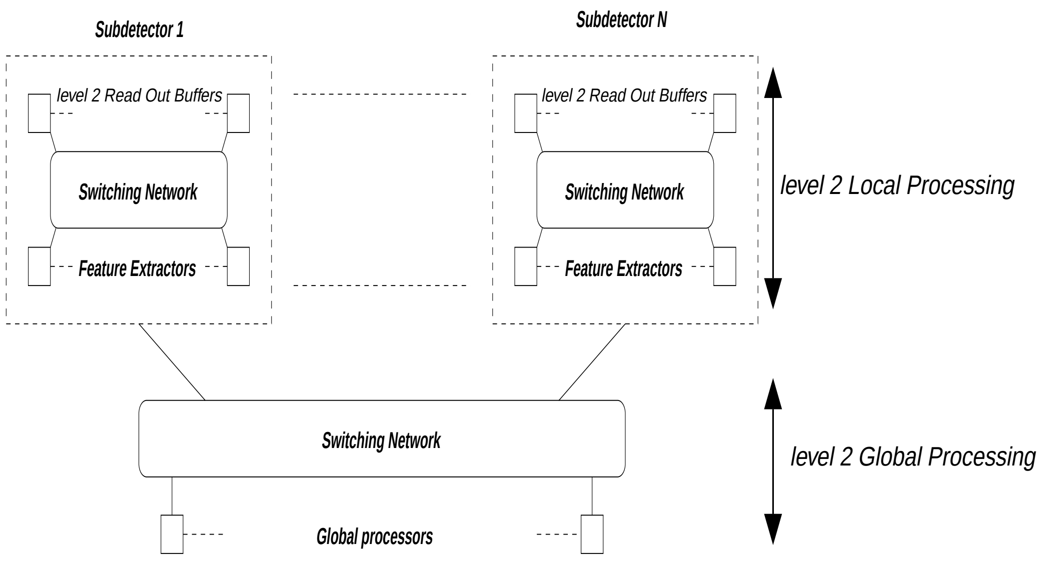


FIGURE 5.2 A local-global level 2 trigger architecture

A supervisor controls the operation of the level 2 trigger. It receives the RoI information from level 1, sends RoI information and accept/reject signals to ROBs. The supervisor must also assign processors to events. In general these functions have to be performed for an event input rate of 100 KHz. Information contained in the Atlas technical proposal suggests that 100 ROBs and 20 FEX processors will be active for each event. This requires messages at a rate of 10 MHz spread across to the ROBs just to inform them which FEX processors to use. Messages from the supervisor will also have to inform ROBs to discard data from rejected events. In addition, the supervisor will have to perform computation concurrently with communication to produce the format of the messages, for example, assigning processors. There will be a limit to the overall latency of a level 2 decision of approximately 10 milliseconds, all required communication and computation for a single event will have to completed within this limit.

The general requirements of the level 2 system can be summarised as:

- High message rates (MHz)
- Low latency limit on level 2 decision requires low latency communications and high processing power.
- High bandwidth links, with a level 2 throughput of 10 Gbytes/s
- Efficient support for concurrency of computation and communication
- The system performance must be scalable with the addition of processors, links and switches. It should be possible to build initial systems and upgrade in stages.

5.2.3 The level 3 trigger

All events that are accepted by level 2 are sent to a level 3 processor farm via an event builder. To achieve a further efficient cut in the event rate, level 3 should have access to the entire event for global processing. This requires a level 3 processor to be able to communicate with all ROBs, so the whole data for a single event can be collected into a single level 3 processor.

The rejection factor of the level 3 system is expected to be ~10, with an input rate of 1 KHz. This would correspond to a total throughput of up to 10 Gbytes/s for the level 3 system. Data must be recorded at a rate of 10 to 100 Mbytes/s, with the required reduction in event rate or event size performed by level 3.

A single Data Flow Manager (DFM) will be responsible for assigning destinations to event fragments from ROBs. It will also maintain information on the status of level 3 processors (free or busy), and balance load between the level 3 processors. The performance of the DFM will be crucial to the success of the level 3 system. The requirements are many short control messages at very high frequencies, between the ROBs and the DFM and between the level 3 processors and the DFM. The exact number of ROBs active for each event is not known, and would be variable from event to event. In the worse case all 1000 ROBs would need to be informed of which level 3 processor to send their data to. At a rate of 1 KHz this corresponds to a message rate of 1 MHz. The DFM would also need to perform computation concurrently with communication, for example, assign a level 3 processor to each event. The requirements for level 3 can be summarised as:

- High message rates (MHz)
- High bandwidth links, the throughput of the event builder is expected to be up to 10 Gbytes/s
- High computational power, level 3 processor farm will require 10^6 MIPS
- Efficient support for concurrency of computation and communication
- The system performance must be scalable with the addition of processors, links and switches. It should be possible to build initial systems and upgrade in stages.

5.3 GPMIMD machine applied to level 2 and level 3

Results for level 2 and level 3 are taken from a paper I co-authored entitled ‘Triggering and Event Building Results Using the C104 Packet Routing Chip’ [26].

The maximum amount of grouped adaptive routing was enabled (12 links) between the centre and terminal stages of the Clos. The machine is identical to the machine used to produce the results in Figure 3.6 on page 37, where the bandwidth for fixed pairs and random traffic patterns are presented. The results in Figure 3.6 can be used to calculate the fraction of the maximum achievable bandwidth obtained under level 2 and level 3 traffic patterns.

5.3.1 level 2 results

A level 2 system was mapped onto a 48 node GPMIMD machine, using 20 MHz revision D02 T9000s, containing six motherboards and two switch cards. The slot zero T9000s were not used to allow better interpretation of results: the slot zero T9000s are not part of the folded Clos network. In addition, a further six T9000s were excluded from the measurements, some D02 T9000s were a source of un-reliability, even when a single GPMIMD network was in use. The result was a maximum of 36 processors. All sources were on a separate motherboard to all destinations so a maximum of 18 sources and destinations was used. The measurements presented use only one of the four GPMIMD networks to avoid the D02 T9000 cache corruption bug.

The traffic patterns corresponding to the local processing of a single subdetector were mapped onto the 18 sources and destinations. Essentially one of the upper left or right components of Figure 5.2 is being implemented. This is the local processing phase of subdetector RoIs by feature extraction processors (FEX processors). The sources in the GPMIMD machine correspond to ROBs and the destinations correspond to features extractors.

The level 2 global processing has not been implemented. A level 2 supervisor has not been implemented, all ROBs are loaded with pre-calculated look-up tables that contain the event sequences. A ROB uses it's look-up file to discover whether it must send data for a certain event. It also uses it's look-up file to decide which feature extractor processor to send an event fragment. The T9000s only acted as sources and destination of event data. No computation was performed by the T9000s replacing feature extractors. The result is that the limit of the network communications performance is being investigated, not the limit of the T9000 computational power or a centralised supervisor.

The traffic patterns inside a single subdetector are defined by the following variable parameters[14]:

- Number of features for each event (1 to 5)
- Feature size (0.2 to 1.2 Kbytes)
- Number of ROBs per feature (1 to 5)
- Selection of ROBs to participate in each event (clusters at random or round robin)
- Total number of ROBs and feature extractors in system (3 to 18)

Figure 5.3 shows the traffic pattern for two example events. The messages are those required for the event within a single subdetector for the local processing phase. Each RoI may contain more features to be collected from different subdetectors. For event 0, at time t=100, subdetector A has to compute features for two RoIs. The FEX processors are allocated round robin, so these two computations are performed by the first two FEX processors (0 and 1). FEX processors 2, 3 and 4 compute the features for event 1. After FEX processor 5 has been used, FEX processor 0 will be used.

The possible traffic patterns for consecutive events must fit within the parameters presented above, i.e. 1 to 5 features per event, 0.2 to 1.2 Kbyte feature size and 1 to 5 features for each feature. Despite the large number of possibilities that this allows I will show that a network of a given size has a constant maximum achieved throughput regardless of any of these parameters.

The performance of a single GPMIMD machine network in terms of achieved event frequencies and bandwidths has been measured under level 2 traffic patterns.

14.The numbers were based on the latest information available during the summer of 1995.

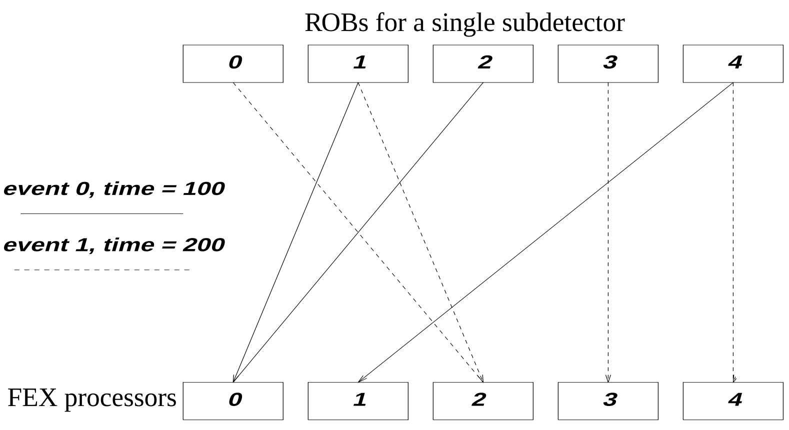


FIGURE 5.3 A level 2 traffic pattern

An important factor is the use of virtual links. A buffer can use different virtual links to send messages to different FEX processors at the same time. If the participating buffers in each event are chosen at random, then a buffer may participate in a number of consecutive events. If a buffer has not completed the send of event n when the time for the send of event n+1 is reached, the two events can be sent in parallel. An extra parameter is defined: the number of virtual links that a buffer can use in parallel. A destination FEX processor (T9000) has a separate process (and virtual link) to communicate with each buffer, this allows the destinations to read from all buffers in parallel.

The relevance of all level 2 results are presented with each plot and are also summarised at the end of the section in Table 5.2. In Figure 5.4 the achieved frequency and bandwidth as a function of the attempted event rate is shown. The plots show results for three different feature sizes: 0.2, 0.6 and 1.2 Kbytes per feature. Each event contains three features and each feature is spread over a single buffer. A total of three sources is active for each event. A random selection of buffers participate in each event. At the maximum achievable event frequencies a total of approximately 70 Mbytes/s out of a possible 111 Mbytes/s (see Figure 3.6 on page 37) is obtained. The result shows that performance scales with feature size and that performance is limited by an asymptotic bandwidth, regardless of the feature size.

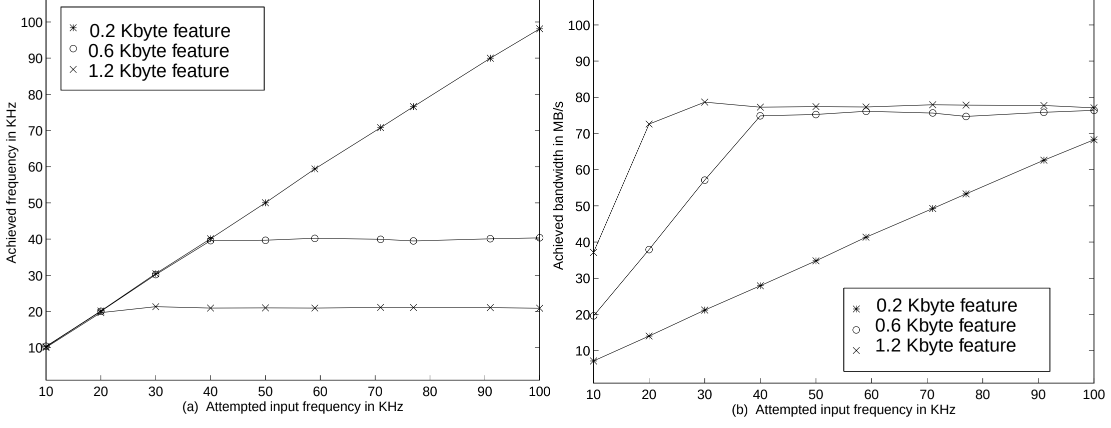


FIGURE 5.4 Achieved event frequency and bandwidth for a level 2 traffic pattern. Participating buffers chosen at random.

The loss of bandwidth (70 Mbytes/s compared to a theoretical maximum of 111 Mbytes/s) is due to the random selection of which sources participate in each event. Sources may have to participate in a number of consecutive events, causing some events to be delayed as the temporarily required bandwidth at a buffer exceeds that available. The queuing of events in the sources results in network under utilisation: messages are queued on a busy source while other sources may be idle. This effect is demonstrated by selecting buffers for each event on a round robin criteria, thus removing the queuing of consecutive events in the sources. For example, if 5 ROBs are active for each event, then event 1 will use buffers 1 to 5, event 2 will use buffers 6 to 10, continuing until the buffers wrap around. The results in Figure 5.5 for this round robin selection criteria show a 97% bandwidth utilisation and a corresponding increase in event rate, confirming event queuing in the sources.

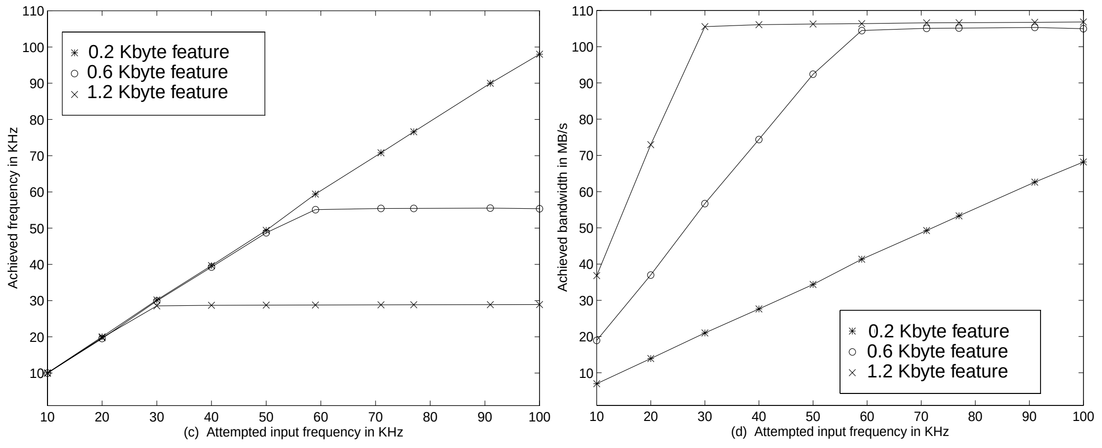


FIGURE 5.5 Achieved event frequency and bandwidth for a level 2 traffic pattern. Participating buffers chosen on round robin criteria

In Figure 5.6 the achieved event frequency and bandwidth as a function of the number of features for an event is shown. The three lines use a fixed feature size of 1 Kbyte but show different numbers of buffer for each feature. There is only a small dependence of the achieved bandwidth across the network on any of these parameters. When a feature is distributed across 5 buffers and not 1, there are 5 smaller messages of 0.2 Kbyte instead of a single message of 1 Kbyte. The range of values from 63 to 77 Mbytes/s can be attributed to lower effective bandwidths for smaller messages. This result shows that the number of ROBs that a feature is distributed over has a small effect on performance and the asymptotic bandwidth remains constant for different numbers of features.

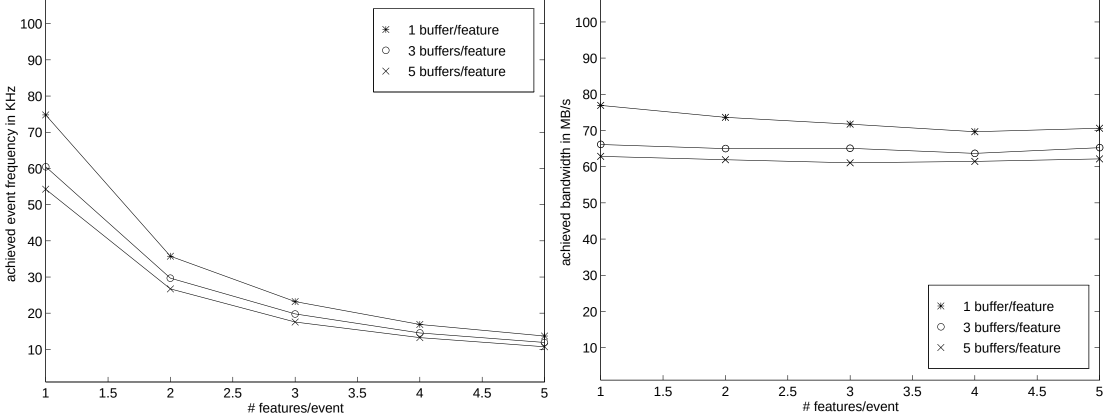


FIGURE 5.6 The achieved frequency and bandwidth as a function of the number of features per event for a level 2 traffic pattern

The performance of the network as a function of the number of sources and destinations using the network is shown in Figure 5.7. The achieved event frequency is shown as a function of the number of active buffers (=FEX processors) for feature sizes of 0.2 and 1.2 Kbytes. In all measurements there are 3 features per event and each feature is spread across 2 buffers. The performance scales linearly from 3 to 18 buffers. Event rates of 100 and 20 KHz are achieved for data block sizes of 0.2 and 1.2 Kbytes respectively. These results only use one of the four GPMIMD networks, hence the performance if all networks were used would therefore be 400 and 80 KHz respectively.

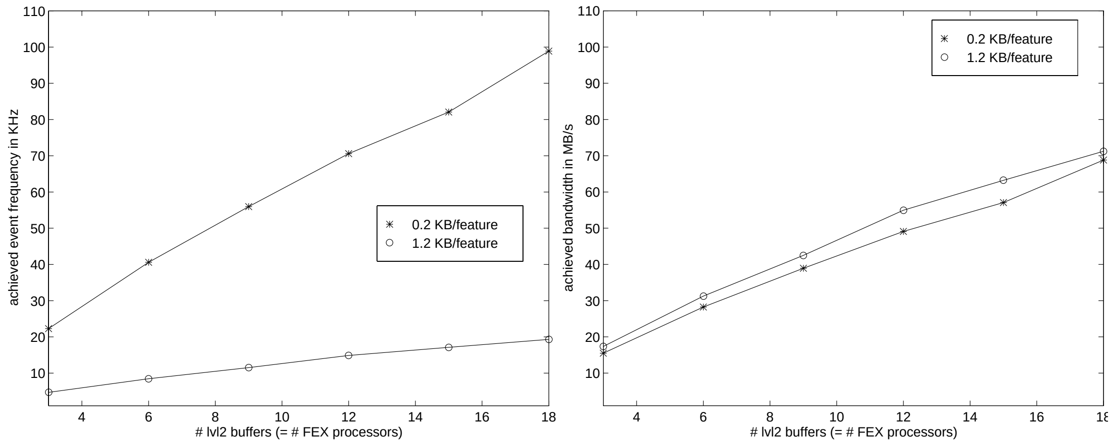


FIGURE 5.7 The achieved event rate and bandwidth as a function of connected sources (=destinations)

In Figure 5.8 the effect of allowing the buffers to communicate on more virtual links in parallel is investigated. It can be seen that the achieved event frequency depends on the number of virtual links used in parallel but the performance does not increase for more than 5 virtual links per physical link. This agrees with the general communications benchmarks presented in chapter 3, which show performance saturating for 5 virtual links.

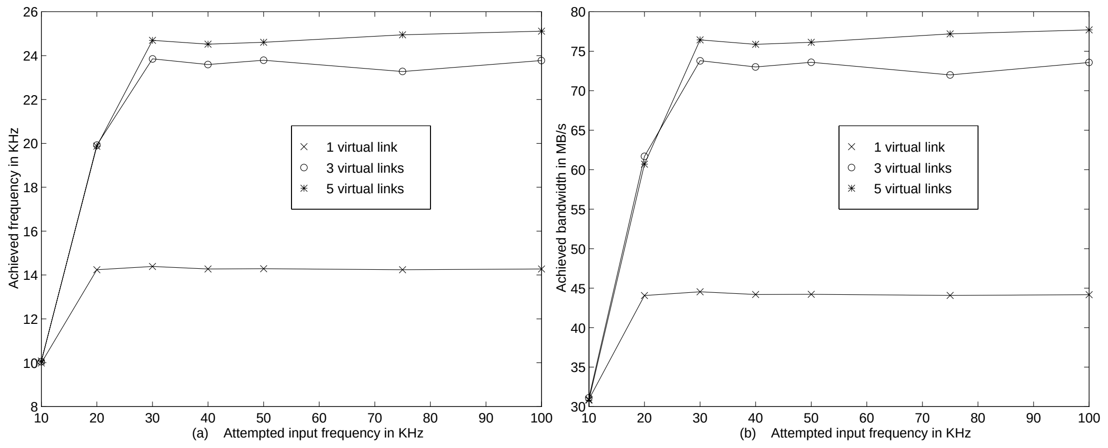


FIGURE 5.8 Effect of virtual links on achieved frequency and bandwidth for level 2 traffic patterns

The limit with a single virtual link is similar to other measurements presented throughout the thesis. A source must wait for acknowledges from a destination. The limitations of this effect are reduced by using multiple virtual links in parallel, while waiting for a packet acknowledge on one virtual link, a source may send a packet on another virtual link to another destination. All other results presented on level 2 traffic allowed 5 virtual links to be used in parallel on each buffer.

5.3.2 level 3 results

The relevance of all level 3 results are summarised at the end of the section in Table 5.2. A level 3 system was mapped onto the same machine configuration. However, only the slot zero T9000s had to be excluded, the six T9000s causing reliability problems for the level 2 traffic patterns did not cause problems for the level 3 measurements. This allowed a maximum of 21 sources and destinations. 21 of the T9000s correspond to ROBs and the other 21 to the level 3 processors.

For each event accepted by level 2 a subset of the ROBs have data which should be collected into a single destination level 3 processor. Figure 5.9 shows the traffic pattern for two example events. The selection of ROBs that have data for a single event have been chosen randomly. The level 3 processor for event 1(2) has data from 4(3) ROBs.

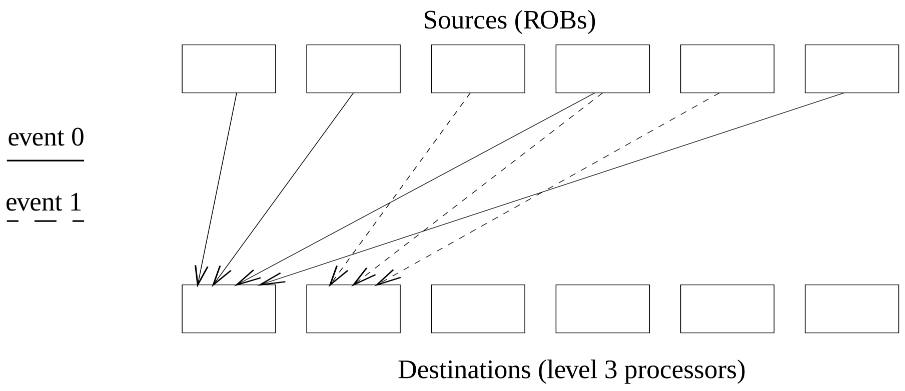


FIGURE 5.9 A level 3 traffic pattern

The data flow manager has not been implemented, all ROBs load pre-calculated look-up tables that contain the event sequences. A ROB uses it’s look-up file to discover whether it must send data for a certain event. It also uses it’s look-up file to decide which level 3 processor to send event data to. As for the level 2 results, the T9000s only acted as sources and destination of event data. No computation was performed by the T9000s replacing level 3 processors. The result is that the limit of the network communications performance is being investigated, not the limit of the T9000 computational power or a centralised supervisor.

The traffic patterns are defined by the following variable parameters, which were the best available estimates during the summer of 1995:

- Total event size within subdetector (10 to 50 Kbytes), obtained by scaling the Atlas requirements to the fraction of the system being implemented.
- Number of ROBs participating in each event (5 to 21)
- Selection of the ROBs to participate in each event (random or round robin)

- Additional latencies in communications libraries (0 to 400 microseconds)
- Total number of ROBs and level 3 processors in system (5 to 21)

The performance, in terms of achievable event rates and bandwidths, of a single GPMIMD machine network has been measured.

Figure 5.10 shows the achieved event frequency for three different event sizes, whilst the number of active sources (ROBs) for each event is varied between 5 and 21. There are a constant number of 14 destinations (level 3 global processors). The total event data is divided equally amongst the active sources.

For each event size there are two lines: solid and dashed. The solid line uses a random selection of ROBs to be active in each event, the dash line corresponds to a round robin selection of ROBs for each event. For example, the round robin selection, if 5 ROBs are active for each event, then event 1 will use buffers 1 to 5, event 2 will use buffers 6 to 10, continuing until the buffers wrap around. When 21 sources are active, all sources are active, so the selection of sources is identical for the round robin or random selection criteria. This explains why the dotted and dashed lines on Figure 5.10 coincide for 21 sources per event.

There is a clear reduction in performance when the buffers are chosen at random for each event. This effect was also seen in the level 2 results. Buffers may have to participate in a number of consecutive events, causing some events to be delayed as the temporarily required bandwidth at the buffer exceeds that available. This queuing of events in the buffers results in network under utilisation: messages are queued on a busy buffer while idle buffers may exist. The effect is demonstrated by selecting buffers round robin, which ensures that buffers are not active in consecutive events whenever possible. For the three event sizes considered, all buffers active, approximately 50 Mbytes/s out of a theoretical limit of 111 Mbytes/s is achieved. A bandwidth of approximately 50 Mbytes/s is achieved regardless of the event size. The results demonstrate reduction in performance due to queuing in sources and show that performance is limited by a maximum achieved bandwidth for all event sizes.

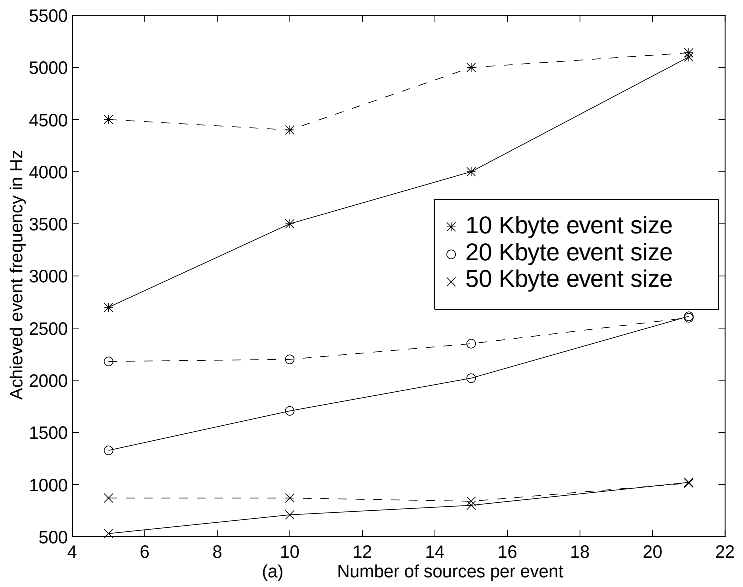


FIGURE 5.10 Achieved event frequency versus number of buffers per event. The solid line shows random source selection and the dashed round robin selection of buffers participating in each event.

In Figure 5.11 the event latency per source (elapsed time to output the data for an event from a source) as a function of event number is shown for three sources in the system used to produce Figure 5.10. The results are for 21 active buffers per event and a total event size of 20 Kbytes. For the first event, all buffers are competing for the first destination link (the level 3 processor to analyse the first event) resulting in a high latency for the first event. The 3 milliseconds corresponds to the time to read 20 Kbytes/s on a single destination T9000 link running at 7 Mbytes/s since that is the limit of a 20 MHz revision D02 T9000. The latency then decreases with event number, settling down to a steady state, as different sources are sending different events to different destinations. The steady state of ~400 microseconds source latency corresponds to the achieved frequency for 20 Kbytes of ~2.5 KHz shown in Figure 5.10.

The total throughput of the system is ~50 Mbytes/s, and the maximum rate a destination link can transfer is 7 Mbytes/s. I assume therefore that multiple destination links are in use in parallel. The throughput of the system is initially 7 Mbytes/s (one destination link active) and then increases to 50 Mbytes/s (whole system with congestion) as multiple destination links are used in parallel. The results demonstrate the reduced event latency per source as gradually more destination links are used concurrently.

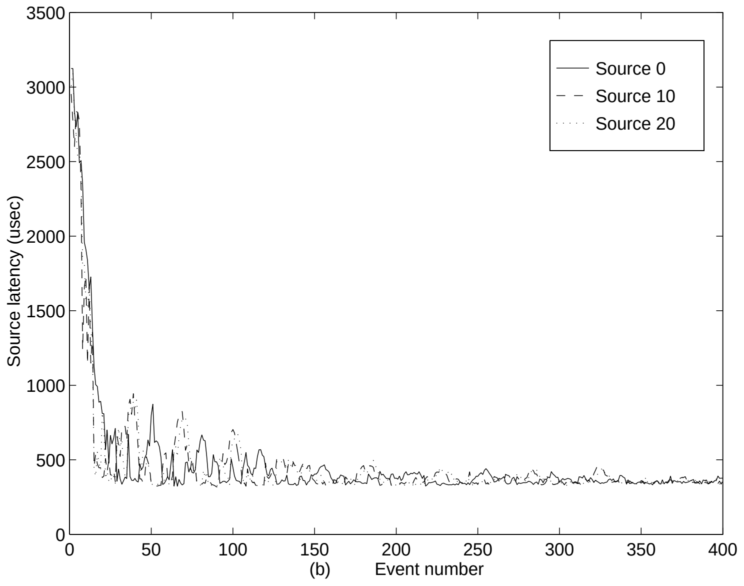


FIGURE 5.11 Source event latency versus event number for 3 of the 21 buffers active per event

A destination T9000 has a separate process (and virtual link) to communicate with each buffer, this allows the destinations to read from all buffers in parallel. A buffer cannot transmit an event until the previous event has been completely transmitted, a single process is used on the buffer.

In order to investigate the effect of communications overhead on overall communications performance the node message latency (I/O initiation time) in the buffers was artificially increased. The results are shown in Figure 5.12 for 21 sources and 14 destinations. The total event size is varied and the achieved event frequency measured. There is a very strong degradation in performance for small increases in latency. An efficient node to network interface, together with low node message latencies is vital to efficiently utilise the available network bandwidth.

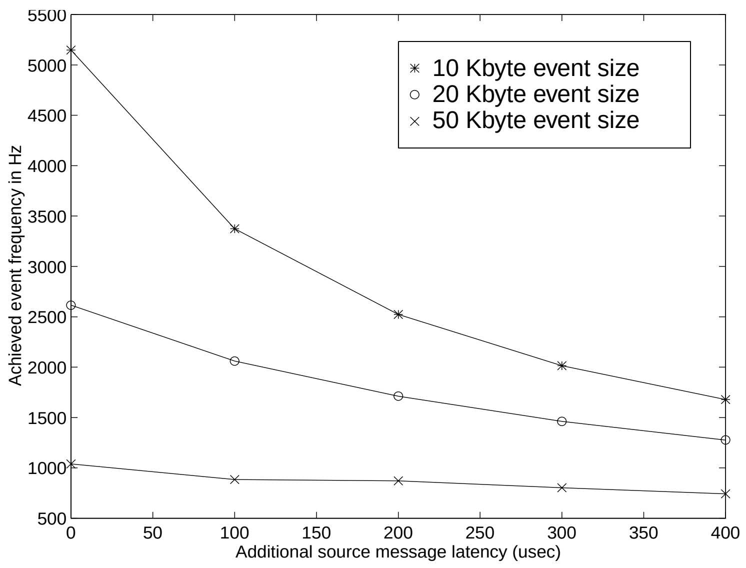


FIGURE 5.12 The achieved frequency versus the additional source (node) message latency

5.3.3 Extrapolation for level 2 to the Silicon Tracker (SCT) subdetector

All extrapolation (for both level 2 and level 3) is based on results contained in the CHEP 95’ paper [26]. The limitation is that these results were produced with a GPMIMD machine which had limited interconnection between the centre and terminal stages, presented in Figure 2.20 on page 27. A fully balanced Clos would have twice as many links between the centre and terminal stages. In all extrapolations I assume this limited interconnectivity, which is half of the complete system. The result is that the extrapolations produce a lower limit. The effect of increasing the number of links ‘inside’ the Clos will be investigated in future work, presented at the end of the chapter.

Figure 5.4 and Figure 5.6 can be used to produce a relation between the number and size of features per event and the rate of events through the network. The plots show that there is an achieved bandwidth of ~75 Mbytes/s regardless of the number or size of features. Reducing the feature size by a factor 3 increases the achieved event rate by a factor 3. In addition, Figure 5.6 shows that the number of ROBs the features are spread over does not have a large effect on the achieved frequency (the spread of results is less than 10%). The variance is due to a larger message passing overhead for multiple small messages compared to a single large message. The conclusion is that the maximum achieved frequency can be calculated from the size and number of features per event to within a 10% error,

$$eventrate = \frac{AchievedBandwidth}{featuresize \times numberfeatures} \qquad \text{where AchievedBandwidth is constant}$$

The three stage Clos network that has been used within the GPMIMD machine will connect a maximum of 512 nodes. The performance of the 36 node Clos can be extrapolated to the performance of a 512 node Clos using C104s and T9000s to drive the links. Figure 5.7 shows that for 5 to 18 sources and destinations the performance scales linearly, i.e. the percentage of the achieved bandwidth compared to the theoretical maximum bandwidth remains constant at 68%. The theoretical bandwidth is taken from Figure 3.6 on page 37, and is equal to 111 Mbytes/s, sources and destinations are paired and there is no contention on destination links. This measurement is the only experimental evidence that I can provide to show scalability to 512 nodes. 512 node physical networks are currently being produced within the MACRAME project (ESPRIT project 8603) and will assess whether the extrapolation presented in this section is valid. The project will also shortly produce network simulators which will allow the performance of larger networks to be predicted.

Results from analytical models combined with simulation are available [24] for Clos networks using random traffic and show approximate scaling for 64 to 512 nodes. However, in these cases all nodes send to all others, as opposed to the level 2 and level 3 traffic patterns which divide the available nodes in two and one half sends to the other. These simulation and analysis results are discussed later in Section 5.4.3, "Benchmark 1.3 (all to all)," which is a benchmark where all nodes send to all others. The conclusion is that to ensure the extrapolation is correct the results must be verified with realistic sized systems within MACRAME.

The SCT subdetector is expected to have 256 ROBs, and if a similar number of FEX processors is assumed, a 512 node Clos will provide all the level 2 local processing interconnectivity required for the SCT subdetector. All information on the SCT requirements are taken from [27].

To extrapolate the performance to a 512 node Clos the theoretical maximum is calculated under the same conditions. I assume that the proportion of links 'inside' the Clos to the number of external links remains the same, the theoretical maximum bandwidth of a Clos scales linearly and the achieved bandwidth is constant at 68% of the theoretical maximum. The theoretical uni-directional maximum of the 512 node Clos is 1.2 Gbytes/s (128 * 9.26), 128 DS links available between the terminal and central stage of the Clos. The new AchievedBandwidth value for the above equation is then 816 Mbytes/s (68% of 1.2 Gbytes/s). I can then use the equation to predict the performance of a 512 node C104 Clos applied to the SCT.

For each feature within the SCT there will be 1.85 Kbytes in each endcap, and 2.31 Kbytes in the barrel, producing an average feature size of 2.0 Kbytes. This allows us to produce a table relating the number of features per event to the achieved event frequency, see Table 5.1
.

TABLE 5.1. Projected event rates for a 512 node Clos applied to the level 2 SCT

Number of features in SCT per event (=RoIs per event)	Projected achieved event frequency
1	408 KHz
2	204 KHz
3	135 KHz
4	102 KHz
5	81 KHz

The number of features clearly determines the extent to which the network can achieve the required event rate of 100 KHz. The Atlas technical proposal states the number of RoIs per event as ~5. If 4 RoIs per event is assumed, a single Clos network could achieve the required rate for level 2, if there are 5 RoIs per event the network can almost achieve the required rate. However, there are three possible ways that performance may be improved:

- The T9000 could be used to drive four independent Clos networks.
- The link speed may be increased by up to a factor 2, my initial tests suggest this will be possible
- Increase by a factor 2 the number of links ‘inside’ the Clos network

It should be noted that I have not included the supervisor or the global level 2 processing in the extrapolation.

5.3.4 Extrapolation for level 3 to the SCT subdetector

The same method is used to extrapolate the level 3 performance as was used for level 2. I assume that the number of level 3 processors is the same as the number of ROBs, this is the ratio of level 3 processor to ROBs for the whole Atlas trigger system (taken from the Atlas technical proposal). In the extrapolation it is assumed that all ROBs send data to the level 3 processor. The result is that there are 14 ROBs and 14 level 3 processors, the number of level 3 processors remained constant at 14 for all measurements. A relation can be produced between the event rate and event size from Figure 5.10, there is an achieved bandwidth of 40 Mbytes/s regardless of the event size. This produces a similar equation to the level 2 extrapolation:

$$eventrate = \frac{AchievedBandwidth}{EventSize}$$

The 14 to 14 network gives a theoretical maximum of 74 Mbytes/s (8 DS links to the central stage), under level 3 traffic patterns 54% of the theoretical maximum is obtained. This figure is lower than the 68% exploitation under level 2 traffic, but the level 3 implementation does not allow a source to use multiple virtual links in parallel.

The bandwidth achieved for a 512 node Clos will be 54% of 1.2 Gbytes/s, which is 648 Mbytes/s. The event size for the SCT can be calculated from [27], which is 256 ROBs each containing 0.5 Kbytes, giving a total of 128 Kbytes per event. The above equation then gives us an achieved level 3 event rate of 5.1 KHz for the SCT sub-detector, a factor 5 greater than the trigger requirements.

5.3.5 Conclusions

In Table 5.2 the relevance of each plot for all level 2 and level 3 results are presented, the contents of the table are a summary of the various explanations given with the plots.

TABLE 5.2. Summary of relevance of each plot for level 2 and level 3

Plot	Relevance
figure 5.4 level 2	Shows that performance scales with feature size and that performance is limited by an asymptotic bandwidth, this result is used for the level 2 extrapolation
figure 5.5 level 2	Shows the congestion effect of queuing in the sources and the reduction of effective bandwidth that this causes
figure 5.6 level 2	Shows the number of ROBs that a feature is distributed over does not have a major effect on performance and that the asymptotic bandwidth is obtained for different numbers of features. The result is used for the level 2 extrapolation

TABLE 5.2. Summary of relevance of each plot for level 2 and level 3

Plot	Relevance
figure 5.7 level 2	Shows the network performance scales for 3 to 18 connected sources and destinations
figure 5.8 level 2	Shows the number of virtual links driving the network in the sources improves performance
figure 5.10 level 3	Shows congestion effect of queuing in sources and the reduction of effective bandwidth that this causes. Also shows the performance is limited by maximum achieved bandwidth regardless of the event size, this result is used in the extrapolation for level 3
figure 5.11 level 3	Shows the reduced source latency with increasing event number, due to more destination links being used concurrently
figure 5.12 level 3	Shows the effect of increased node message latency on performance

Measurements have been presented on a 42 node Clos switching fabric using the C104 and T9000. Under the currently expected traffic patterns of subdetectors in the level 2 and level 3 triggers of the Atlas experiment bandwidths of 50 to 75 Mbytes/s have been achieved, corresponding to ~50 to 70% utilisation of the theoretical bandwidth. The performance relied upon the concurrent use of multiple virtual links, in particular, destinations reading from multiple sources in parallel.

I have shown for level 2 and level 3 traffic patterns the limit of performance is queuing in the sources. This is demonstrated by 97% bandwidth utilisation for round-robin choice of sources in the level 2 traffic pattern. In the actual experiment it will be difficult to have any control of the selection of sources for each event.

For level 3 traffic patterns I have shown very large initial latencies when all sources try to send to a single destination. Performance quickly reaches a steady state after a few hundred events. I have shown the effect of increased message latency on network performance, showing that low message overheads are crucial.

Results have been extrapolated for a full size SCT subdetector. I have shown that a single Clos network of C104s should provide the required level 2 event rate for 4 RoIs per event. If five RoIs must be analysed per event then a small improvement must be found. The improvement could come from increased link speeds, multiple networks and adding a factor two more links between the centre and terminal stages of the Clos. More importantly there is uncertainty of a factor 5 in requirements (1 to 5 RoIs per event) depending on the final Atlas processing strategy for events. Results extrapolated to the level 3 system show that a 512 node Clos should perform five times the required event rate for the SCT.

5.4 Comparison with other parallel platforms

The Atlas communication benchmarks have been defined to allow the comparison of performance of parallel computing platforms applied to triggering in HEP experiments. The benchmarks are divided into two parts: a set of general benchmarks which may be relevant for other applications and an application specific part which is closer to the expected traffic patterns in HEP trigger systems. The Atlas communication benchmarks have been run on the GPMIMD machine, a selection are presented to allow comparison to other platforms.

The only available platforms with numbers of nodes comparable to the GPMIMD machine are commercial parallel computing systems. The Atlas benchmarks have been run on the GPMIMD machine, the SGI challenge [28], the IBM SP-2 [29] and the Meiko CS-2 [30]. The interconnects on the SGI, SP-2 and Meiko are proprietary. Interconnect standards such as SCI, ATM and Fibre Channel are of considerable interest to Atlas but:

- There are only results for very small systems
- The Atlas benchmarks have not been reported

To allow a direct comparison the results are reported in identical form to those already available for other platforms. The following sections give a brief outline of the benchmarks, full details are available in [23]. Each section gives details of the quantity measured and some notes specific to the implementation on the GPMIMD machine. All T9000s are 20 MHz GAMMA revision E03 T9000s using 8Kbyte internal memory and 8Kbyte cache. All C104s are revision beta, all links run at 100 Mbits/s. All results use only a single network of the GPMIMD machine.

The results are compared with MPI [31] versions for the other machines for two reasons: it is the only version for which results are available for all three platforms and it uses the message passing concept at the same level of abstraction as the T9000 toolset. For the Meiko platform programming at a lower level of abstraction is possible, using the Elan widget library [30], however, the functionality of the T9000 toolset is not provided by these libraries. For many of the benchmarks there are limited results available for comparison, results that are not available are shown as a ‘-’.

5.4.1 Benchmark 1.1 (one-way)

One receiver and one sender. A single message is sent from the sender and then once it is received a single message is sent back from the receiver (ping-pong). The time reported is the time for the ping-pong divided by 2. This benchmark measures the basic uni-directional performance, and is identical to an international standard benchmark: Parkbench COMM1 [32].

On the GPMIMD implementation two T9000s on a single motherboard communicated via a separate virtual link in each direction. A single C104 connected the two T9000s. Results are presented in Table 5.3.

TABLE 5.3. Benchmark 1.1 (one way) Results are: message time (effective bandwidth)

Message Size (bytes)[a]	**GPMIMD** µs (Mbytes/s)	**SGI-MPI** µs (Mbytes/s)	**SP2-MPI** µs (Mbytes/s)	**Meiko-MPI** µs (Mbytes/s)
1	6.6(0.15)	132.2(0.008)	164.10(0.006)	257[b](0.004)
64	16.9(3.79)	145.9(0.44)	168.8(0.38)	262(0.244)
256	56.9(4.5)	150.4(1.71)	214.1(1.2)	286(0.9)
1024	216.9(4.72)	239.6(4.28)	326.2(3.14)	392(2.61)

a. All 1 byte messages are actually sent as four bytes messages for the GPMIMD implementation of all benchmarks

b. For the Elan widget library the value is 40 microseconds

The benchmark assumes that the elapsed time for a message is composed of two parts: a fixed message passing overhead (T_0) and the time to transfer the message, which is the message length (L) divided by the effective transfer rate (Reffective). The relation can be expressed as

$$ElapsedTime = T_0 + \frac{L}{Reffective}$$

The message passing overhead is defined as the elapsed time to transfer a zero length (1 byte) message from an application program in the source node to an application program in the destination node. This value is the benchmark 1.1 result for a 1 byte message, for the T9000 implementation this value is 6.6 microseconds. The results from benchmark 1.1 produce a value for the affective transfer rate by fitting results to the above equation. The asymptotic bandwidth achieved is calculated using the benchmark 1.1 result for very large messages.

The message passing overhead can be divided into two further components: the network latency and the node message latency. The network latency is defined as the time to propagate a single byte through the switching network. The node message latency is defined as the I/O initiation time in a source or destination.

The equation allows the calculation of two very important parameters for quantifying the performance of parallel computers: the message passing overhead and the achieved asymptotic bandwidth.

The message passing overhead for the T9000 is considerably lower than the SGI, SP2 and Meiko. This is due to a combination of the low network latency provided by the C104 and low node message latency of the T9000. It should be noted that the link bandwidth of the other platforms is higher than the T9000 links. The links on the Meiko platform can transmit data at 50 Mbytes/s, and the links on the SP2 at 40 Mbytes/s. For longer messages a high message passing overhead can be compensated for with higher link speeds, although the raw link performance is not approached.

For ideal performance the Reffective in the above equation should be the raw transmission speed of the interconnection medium. In Figure 5.13 the benchmark 1.1 results for the GPMIMD implementation are plotted against message length. In addition, the ideal performance is shown where Reffective is the raw transmission speed of the link. The raw transmission rate of a 100 Mbits/s link is approximately 10 Mbytes/s (one byte sent as a 10 bit token over link, see Figure 2.7).

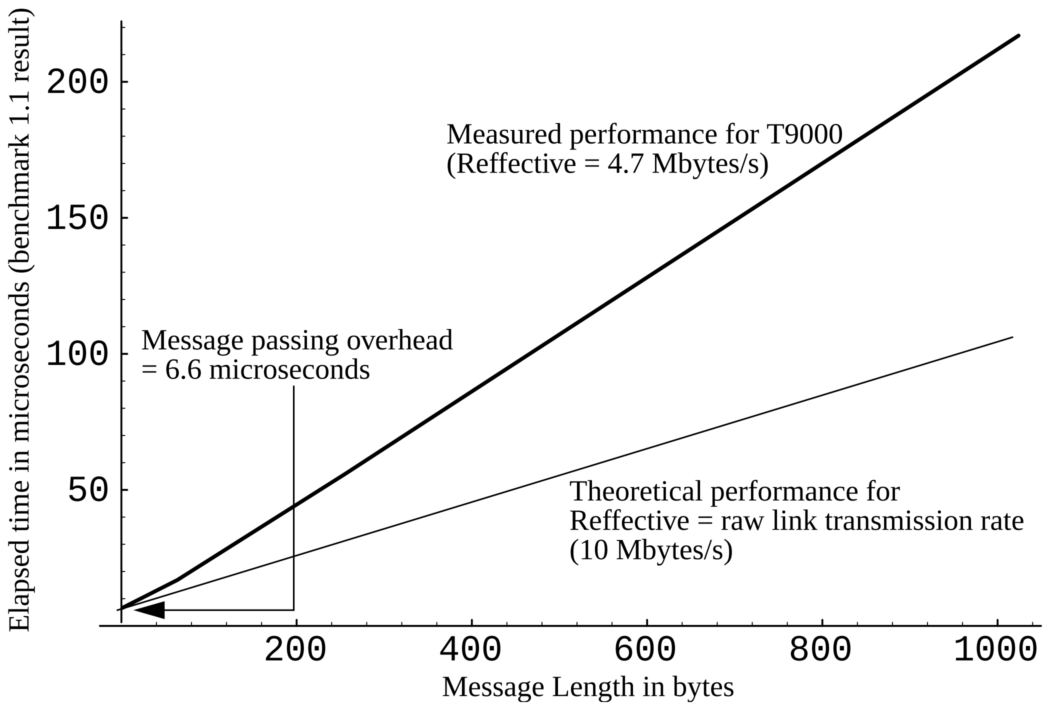


FIGURE 5.13 Measured and theoretical results for the T9000 implementation of benchmark 1.1

It is clear that for all benchmark 1.1 results the achieved Reffective is not equal to the raw transmission rate of the link. Reasons for this are:

- Node interface to the raw link, i.e. the rate at which data can be transferred into the application program from the link
- Protocol implementation overheads, e.g. packetisation of messages and production of acknowledges

5.4.2 Benchmark 1.2 (two-way)

Identical to benchmark 1.1 but with the communications performed in parallel. The time reported is the time for a transmit in both directions divided by 2. The benchmark measures basic bi-directional performance. i.e. the extent to which a link can be used in both directions. This benchmark is identical to Parkbench COMM2 [32].

On the GPMIMD implementation two T9000s on a single motherboard communicated via a separate virtual link, both virtual links used in parallel. A single C104 connected the two T9000s. Results are presented in Table 5.4. This benchmark measures the extent to which

TABLE 5.4. Benchmark 1.2 (two way). Results are: time for message (effective bandwidth)

Message Size (bytes)	**GPMIMD** µs (Mbytes/s)	**SGI-MPI** µs (Mbytes/s)	**SP2-MPI** µs (Mbytes/s)	**Meiko-MPI** µs (Mbytes/s)
1	4.9 (0.2)	81.4(0.012)	103.7(0.01)	137(0.007)
64	10.1(6.34)	85.4(0.75)	101.5(0.63)	146(0.44)
256	33.6(7.62)	109.4(2.34)	121.2(2.12)	171(1.5)
1024	127.2(8.05)	183.7(5.57)	202.6(5.05)	286(3.58)

links can be used bi-directional. A simple comparison would be the fraction bi = COMM2/ COMM1, with bi = 0.5 the ideal result where both directions are used in parallel without affecting each other, and bi = 1 the worst case where there is no bi-directional use of the link at all. For the GPMIMD implementation and 1024 byte message size bi = 0.59, for SGI bi = 0.77, for SP2 bi = 0.62 and for Meiko bi = 0.73. The GPMIMD implementation is the most efficient at using the link bi-directional.

5.4.3 Benchmark 1.3 (all to all)

All processors are simultaneously senders and receivers. All processors send to all other processors. The benchmarks investigates the scalability of the switching network when there is contention on destination links. The time reported is the time for all processors to send a message to all other processors. This benchmark corresponds to Parkbench COMM3 [32].

For the GPMIMD implementation every T9000 has a separate virtual link to every other processor. Every T9000 also has a separate process for each virtual link, to allow it to drive all virtual links in parallel. The slot 0 T9000s were not included. To interpret these values the system wide bandwidth is required, which allows an investigation of scalability with the number of processors. The amount of data passing through the entire network is:

(message length) x number of processors x (number of processors -1)

The system wide bandwidth is then this amount divided by the elapsed time. Figure 5.14 presents the system wide bandwidth for the GPMIMD machine and SP2.

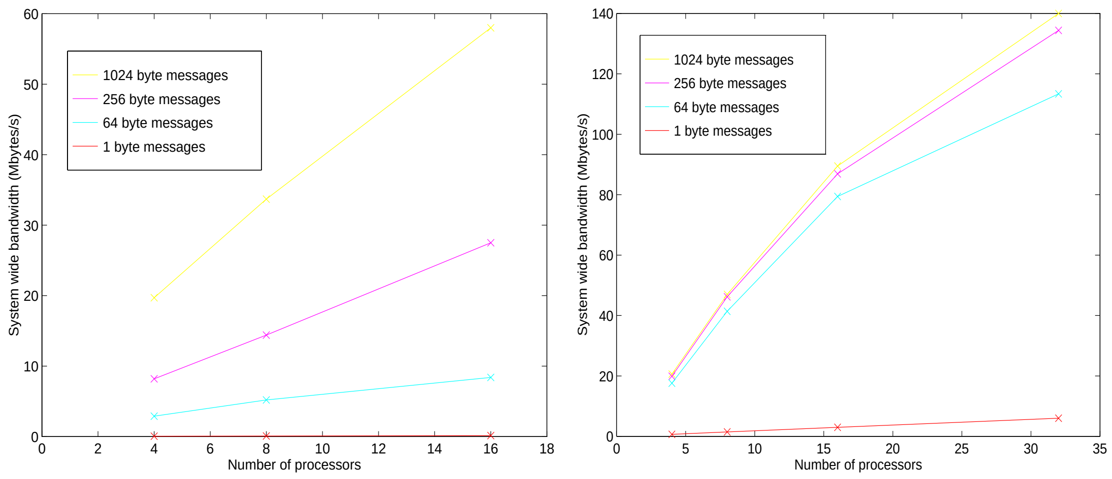


FIGURE 5.14 System wide bandwidth for benchmark 1.3: left plot is SP2, right is GPMIMD

The system wide bandwidth does not scale linearly with the number of processors for the SP-2 or GPMIMD machine. The performance increase from 4 to 16 nodes is close to linear for both implementations. The overall performance is again better for the GPMIMD implementation especially for short messages, where the low message passing overhead is dominant.

The degradation of performance with increasing numbers of processors agrees with analytical models presented within [24]. The work predicts the performance of Clos networks under sustained random traffic, where all nodes randomly send to another node picked from all possible destinations. Results cannot be compared quantitatively as the analysis starts with networks of at least 64 processors. The analysis predicts a 25% degradation (compared to linear scalability) from 64 to 512 nodes.

It should be noted that this is a different traffic pattern to the level 2 traffic pattern, which I have shown scales from 3 to 18 sources and destinations, and extrapolated to a 512 node Clos for that traffic pattern. For the level 2 and level 3 traffic pattern the nodes are split into two groups and one half sends to the other, i.e. all links are used uni-directional and there is no communication between processors on the same motherboard. In contrast, the all to all benchmark requires processors on the same motherboard to communicate with each other and uses all links to the nodes bi-directional.

5.4.4 Benchmark 2.1 (push farm with supervisor)

This benchmark is a more detailed implementation of the level 2 system already used to produce results earlier in this chapter. Figure 5.15 shows the configuration of senders and receivers. A supervisor processor communicates to all sources a receiver (destination) address. The destinations are selected using a round robin criteria. The source processors then send data to the indicated destination processor. When the destination processor has collected data from all sources it informs the supervisor of a decision. All messages exchanged with the supervisor are 16 bytes. The benchmark measures basic push-farming performance assuming processor control is from the supervisor through the switching network. The time reported is the time on the supervisor between the sending of the messages to the sources and the reception of a decision from a receiver. The sources correspond to ROBs and the receivers to FEX processors.

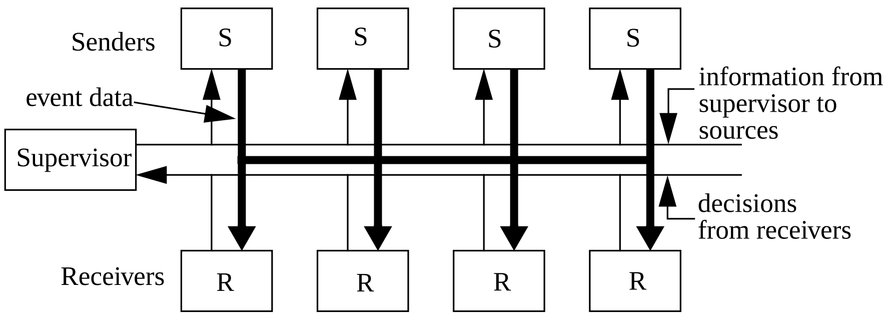


FIGURE 5.15 Benchmark 2.1 (push farm with supervisor)

For the GPMIMD implementation, all messages between all processors were 16 bytes. This is because all measurements available for comparison used 16 byte messages between senders and receivers. The supervisor is always on a separate motherboard to all senders and receivers, and no sender is on the same motherboard as a receiver. Results are shown in Table 5.5. The results for the GPMIMD implementation show far lower times than for the other platforms (where results are available). This is due to the lower message passing overheads on the GPMIMD implementation.

TABLE 5.5. Benchmark 2.1 (push farm with supervisor). Results are time for event (event rate).

Total no. of Senders	**Number of receivers**	**GPMIMD** μs (KHz)	**SGI-MPI** μs (KHz)	**SP2-MPI** μs (KHz)	**Meiko-MPI** μs (KHz)
2	4	41.2(24.4)	-	271(3.69)	217(4.61)
4	4	50.5(19.8)	-	-	286(3.5)
6	4	60.8(16.4)	-	-	-
8	4	70.4(14.2)	-	-	453(2.21)

The performance of a pull farm was also measured, the supervisor communicates to a receiver and the receiver fetches data from all sources. The difference is due to one extra message between the supervisor and the receivers. This accounts for an approximate difference of 10 microseconds between a push farm and pull farm.

5.4.5 Summary of comparisons to other platforms

Despite the lower link bandwidth of the T9000 the results have been generally better than the alternative platforms. The performance difference is particularly wide for smaller messages. This is due to the dominance of message passing overhead for sending short messages, the T9000 has an extremely low message passing overhead of 6.6 microseconds. In addition the T9000 exploits the VCP, virtual channels and efficient context switching to provide efficient support for concurrent communication/computation and high message rates. The high message rates are particularly important for the supervisor in benchmark 2.1. It should also be noted that the GPMIMD results used only one of the four available networks.

5.5 Conclusions

Measurements have been presented on a 42 node Clos switching fabric using the C104 and T9000. Under the currently expected traffic patterns of subdetectors in the level 2 and level 3 triggers of the Atlas experiment, bandwidths of 50 to 75 Mbytes/s have been achieved, corresponding to ~50 to 70% utilisation of the theoretical bandwidth. Results have been extrapolated for a full size SCT subdetector. These extrapolations will be further investigated in future work, presented in the next section.

The results from the Atlas communication benchmarks have shown the advantages of low message passing overheads, especially for short messages. For example, the 2.1 Atlas benchmark shows far better performance for the GPMIMD implementation because lots of small messages are required from the supervisor or destinations. The latency for these small messages has been more influential on the results than the raw link bandwidth, which is lower for the GPMIMD implementation than all other platforms used for comparison.

At the end of chapter 3 I had identified critical factors which effect the performance of multiprocessor systems (Table 3.6 and Table 3.7) using point to point links and switches. After analysing the requirements of the Atlas trigger system I am able to quantify the importance of each factor to Atlas. The requirements of level 2 and level 3 can be summarised as follows:

- High message rates for messages of ~1 Kbyte (implies low I/O loading on CPU)
- Efficient support for concurrent communication and computation (implies fast context switch and low I/O loading on CPU)
- Low latency communications
- Scalability to large systems
- High bandwidth and computational power

The factors presented in Table 3.6 and Table 3.7 are also in Table 5.6 and Table 5.7, but the relevance is now applied to Atlas and the support they provide for the above requirements. The factors are split into two groups: factors vital to network performance and factors vital for nodes driving the network (see Table 5.6 and Table 5.7). Other additional factors which I believe are important, but I have not yet investigated will be presented in the future work section at the end of the chapter.

TABLE 5.6. Critical factors affecting network performance and their relevance to Atlas

Factor	General relevance or importance to Atlas data acquisition
C104 adaptive routing	High bandwidth requirements of level 2 and level 3 rely on the efficient use of available network links. Adaptive routing allows the efficient use of all links available between the terminal and centre stages of a Clos network. A possible alternative of grouped adaptive routing is to have nodes pre-determine the route which a packet will take before it entered the network, adding load and complexity to the nodes. Another alternative would be a fixed route through the network for all packets which would increase contention.
Scalability to large networks	The performance of level 2 and level 3 must scale to meet the increasing requirements of the experiment. Scalability must allow initial systems to be upgraded in stages.
Network latency	The time to switch packets through the network is a component of the message passing overhead. Therefore it should be minimised, the C104 switches packets in one microsecond.
Link bandwidth	To provide high throughput the network will require high speed component links. However, this is a necessary but not sufficient condition for high throughput, the performance of the node interfacing to the network will also be crucial.

TABLE 5.7. Critical factors affecting nodes driving the network and their relevance to Atlas

Factor	General relevance or importance to Atlas data acquisition
Node message latency	Low node message latencies are important to allow low message passing overheads. level 3 results with increased node message latency demonstrate the affect on performance. Atlas benchmark results show better performance of GPMIMD machine even though link bandwidths are lower than the other platforms.
Link interface bandwidths	High link interface bandwidths between the raw link and the application program are crucial to achieve the high bandwidth requirements of level 2 and level 3.

TABLE 5.7. Critical factors affecting nodes driving the network and their relevance to Atlas

Factor	General relevance or importance to Atlas data acquisition
A separate communications controller, e.g. The VCP and virtual channels (facilitating low node message latency and high link interface bandwidths)	Low I/O loading due to the VCP is crucial for high message rates, efficient support for concurrent computation/communication and low message latencies. Virtual channels and the VCP allow: Better exploitation of link bandwidth in C104 networks by multiplexing multiple virtual channels onto a single physical link. Multiplexing/demultiplexing in hardware (of up to 64,000 channels) avoids requirement of software (and the CPU) to handle communications received on a physical link to multiple possible processes. The VCP uses multiple concurrent DMAs to transfer data from messages directly to the required location when they arrive, no further memory copies are required. This reduces latency and I/O load on the CPU. The result is that the VCP facilitates low node message latency and high link interface bandwidths. The VCP is essentially an efficient tightly coupled on-chip communications controller for the T9000.
Fast context switch times	Fast context switch times are important for multi-processing and exploiting low I/O loading of the CPU to provide efficient support for concurrency of communications and computation. A process must be scheduled quickly and efficiently to exploit the CPU time made available by the low I/O load on the CPU. In addition, interrupt times are crucial in any system with real-time constraints.
Multiple physical links	The supervisor processor may use a separate network to control system components. Additional performance may be gained from using multiple networks.
Computational performance	level 3 computational requirements are 10^6 MIPS. This performance requires multiple high performance processors. A possible solution is hybrid nodes, e.g. the TransAlpha, a high performance RISC processor using a T9000 as a dedicated communications controller.
Language support for parallel processing	The programming interface and environment are crucial to allow the user to exploit the performance of the hardware. Occam provides an example of the advantages of a language designed for parallel processing, see comparison in Section 2.2, “Occam,”.

The following points summarise the conclusions corresponding to each of the three goals set out at the start of this chapter:

- I have presented the projected performance of DS link technology using the T9000 and C104 applied to the SCT detector for level 2 and level 3.
- I have listed the factors which I believe are crucial for high performance in trigger systems, related to the nodes driving the network and the network performance. Not all of the factors have been investigated, those which have not are presented in the future work section below.
- A comparison to other parallel platforms has been presented. When more results are available using other technologies further comparisons will be allowed.

5.6 Future Work

I expect the future work to be carried out on three platforms: the GPMIMD machine, ARCHES and the MACRAME testbeds. The MACRAME project is building a 1000 node variable topology and packet length switching network. ARCHES will build a network of HS links using the Rcube [33] router. Network simulators will also be available to model HS and DS link networks. Table 5.8 shows the factors which will be investigated (along with the existing factors that have been presented in Section 5.5, “Conclusions,”), and the platform to be used.

TABLE 5.8. Crucial factors to be investigated in future work

Factor	Affects node or network performance	Platform to be investigated on
Universal routing	Network	MACRAME
Multiple networks	Network	GPMIMD/MACRAME
Higher link speeds	Network	ARCHES/MACRAME
Network topology	Network	MACRAME
Packet size	Network	MACRAME
Cost	Network and Node	ARCHES/MACRAME
Node interfaces to network	Node	ARCHES/MACRAME
Noise, errors and fault tolerance	Network and Node	GPMIMD/MACRAME/ARCHES

Both platforms will also be used to investigate the extrapolations made within this thesis to a 512 node Clos, both with modelling and by direct measurement. When results for other parallel platforms and interconnect technologies become available a more detailed comparison to DS link technology can be made.

Chapter 6
Concluding remarks

6.1 Original goals

This thesis has presented work in three parts: a general technology evaluation of IEEE 1355, DS links and the T9000, on-line event reconstruction in CPLEAR and the application of DS links to future generation experiments.

The goal of the technology evaluation presented in chapter 3 was to compare the T9000 and C104 to other technologies and identify a set of factors crucial to performance.

The goals of the CPLEAR work presented in chapter 4 were:

- Demonstrate the feasibility of reconstructing and filtering the full CPLEAR event rate on-line.
- Proof of existence and successful operation of the T9000 and C104 in a demanding experimental environment.

The application of DS link technology to the Atlas detector at the LHC presented in chapter 5 had the following goals:

- Discover the extent to which currently available technology, i.e. DS link technology, can meet the requirements of future generation experiments.
- Identify and investigate the crucial factors affecting the performance of point to point links and switches applied to future generation experiments.
- Compare the performance of DS links to other relevant platforms

6.2 Summary of results

The communication results are very promising and can be summarised as follows:

- Single link results show low message passing overheads and low message latencies. The C104 has been shown to switch packets in a single microsecond.
- I have demonstrated the ability of the T9000 using the VCP to concurrently perform communications and computation due to low I/O loading on the CPU.
- The control/supervisor benchmarks show high message rates from a single T9000 node (greater than 400,000 per second).
- The C104 performs grouped adaptive routing with no measurable overhead, and the achieved bandwidth has scaled linearly with the number of links grouped.
- The uni-directional communication results for the T9000 scale linearly for one to four physical links.

- The T9000 revision D02 suffered from serious external memory problems, the 64 bit interface T9000 could only write data at 16.4 Mbytes/s to the links from external memory. The situation is improving with the availability of the revision E03.
- Context switching performance and interrupt response have been measured and compared to other platforms. Context switch times are better by orders of magnitude.
- The 20 MHz T9000 suffers from a fundamental shortfall in computational power, which can be remedied by the use of the TransAlpha module
- The prototype revision D02 T9000 suffers from multiple serious hardware bugs which have fortunately been fixed with the production revision E03.

After analysing the requirements of the CPLEAR experiment I have shown that a 25 node TransAlpha farm could analyse the full event rate produced by CPLEAR. A network of 54 T9000s and 20 C104s has been installed in the experiment and a stable platform was obtained which ran without failures for the last 128 hours of the experiment run.

Measurements have been presented on Atlas level 2 and level 3 traffic patterns using a 42 node Clos switching fabric. Bandwidths of 50 to 75 Mbytes/s have been achieved, corresponding to ~50 to 70% utilisation of the theoretical bandwidth. Results have been extrapolated for a full size SCT subdetector. Results for a set of standard and Atlas specific benchmarks have been presented and compared to other parallel platforms. The performance of the T9000 and C104 was generally better than the alternative platforms.

6.3 Achievements

An evaluation of the technology has been presented and results understood and explained. I have identified and evaluated a set of critical factors which affect the performance of multiprocessor systems. Thus the goals of the technology evaluation have been achieved.

I have demonstrated the feasibility of analysing the full CPLEAR event rate on-line. Despite the prototype nature of the revision D02 T9000 and the known hardware bugs, some of them severe, we have been able to construct a network of 54 T9000s and 20 C104s. The system has been installed and operated reliably in the CPLEAR experiment at CERN, running a very large event reconstruction program in real-time. Proof of existence, successful and reliable operation of large T9000 and C104 systems has been achieved. The construction of a substantial array of processors operating in an experimental environment uncovered problems that would have never been found for ‘small’ laboratory test systems. The problems that have been presented will be typical for future experiments at the LHC, for example, interfacing different link technologies, the succeptability of high speed links to electromagnetic noise and monitoring large distributed systems. Thus the goals for the application of the technology to the CPLEAR experiment have been achieved.

The reliability of the revision D02 T9000 in the CPLEAR experiment has given a worst case result. The Gamma E03 T9000 and revision beta C104 are now commercially available and will improve reliability and more importantly remove the restrictions imposed by the D02 bugs. Initial tests with the GPMIMD machine upgraded to Gamma E03 and C104 revision beta have shown that the problems of stability and reliability have been solved.

The L3 experiment at CERN has installed a network of two C104 routers (initially revision alpha) interconnecting 48 T9000s (initially revision D02) which is operating reliably as a second level trigger. The L3 system has now been upgraded to beta C104s and revision E03 T9000s. Our work in CPLEAR and general experience with the T9000 and C104 was of considerable use to L3 in the development of this system [22].

To address applicability of DS links to LHC and in particular Atlas, I have extrapolated results to the SCT subdetector of Atlas. The conclusion is that DS link technology can satisfy a significant proportion of the requirements of future generation experiments at the LHC. These extrapolations will be further investigated in future work.

The results from the Atlas communication benchmarks have shown the advantages of low node message passing overheads and network latencies of the T9000 and C104. A comparison to other parallel platforms has been presented. When more results are available using other technologies further comparisons will be allowed.

After analysing the requirements of the Atlas trigger system I have been able to identify a list of all factors crucial to the performance of point to point links and switches in future experiments. This list quantifies and presents the relevance to Atlas for each of the factors given at end of chapter 3: the technology evaluation.

The factors are split into two groups: those vital to network performance and those vital for nodes driving the network.

Factors vital for network performance:

- C104 adaptive routing
- Scalability to large networks
- Network latency
- Link bandwidth

Factors vital for nodes driving the network:

- Node message latency
- Link interface bandwidths
- A communications controller, i.e. the Virtual Channel Processor and Virtual channels
- Fast context switch times
- Multiple physical links
- Computational performance
- Language support for parallel processing

The extrapolation to the SCT subdetector, the comparison to other parallel platforms and the list of crucial factors have satisfied the goals for the application of DS links to future generation experiments.

6.4 Future work

Other additional factors which I believe are important and effect both network and node performance, but have not yet been investigated are:

- Universal routing
- Multiple networks
- Link speed
- Network topology
- Packet size
- Cost
- Noise, errors and fault tolerance

I expect future work investigating these factors to be carried out on the existing GPMIMD machine, and the testbeds for DS and HS links developed in the MACRAME and ARCHES ESPRIT projects.

6.5 Final remarks

The late delivery of the T9000 has meant that in the meantime many vendors have passed the 50 MHz full specification computational performance of the T9000. The T9000s that finally became available are only 20 MHz, in the near future perhaps 30 MHz, which has created a situation where the T9000 has a fundamental shortfall in computational power. This has been demonstrated in comparisons to the PowerPC and DEC Alpha. The solution to this particular shortfall has been the development of the TransAlpha board, a combination of T9000 communications performance and DEC Alpha computation.

The Gamma E03 T9000 is now commercially available. At the time of writing the GPMIMD machine has been upgraded to 64 Gamma E03 T9000s and 58 beta C104s. The main improvement over the D02 is the increased stability due to the removal of hardware bugs. The external memory interface has also been improved.

The T9000 is still interesting as a communications controller due to its specialised design for use in multiprocessing, in particular the communications performance through on chip hardware and software development environment. It is being considered for such a role in several major research and development projects, and is already being shipped in existing products.

The T9000 Transputer was brilliantly conceived yet poorly implemented and marketed. Its legacy is in the field of high speed communications, IEEE 1355 may yet succeed.

References

[1] M. Turala and S. Cittolin, Detector Techniques and Data Acquisition for LHC experiments, CERN Academic Training, March 1996.

[2] IEEE Std. 1355, Standard for Heterogeneous Inter-Connect (HIC). Low Cost Low Latency Scalable Serial Interconnect for Parallel System Construction. IEEE Inc., 1995.
http://www.omimo.be/standard/omi303/omi303.htm

[3] The T9000 Transputer Hardware Reference Manual. Inmos Ltd., Inmos document number 72 TRN 238 01.

[4] The STC104 Asynchronous Packet Switch, Preliminary Data Sheet, June 1994, SGS-Thomson Microelectronics.
http://www.cern.ch/HSI/dshs/Welcome.html

[5] GPMIMD Consortium, GPMIMD P5404, Technical Annex for ESPRIT, revised 1994.

[6] L. Adiels et al., Proposals for the experiment PS195. CERN/PSCC/85–6/P82, PSCC/86–34/M263, PSCC/87–14/M272.
http://www.cern.ch/cplear/Welcome.html

[7] The ATLAS Technical Proposal, CERN/LHCC/94–43, LHCC/P2, ISBN: 92–9083–067–0.
ftp://www.cern.ch/pub/Atlas/TP/tp.html

[8] C.A.R. Hoare, Communicating Sequential Processes, Prentice Hall, 1986.

[9] John Galletly, Occam 2, Pitman, 1990.

[10] The ESPRIT project ARCHES, Application, Refinement and Consolidation of HIC exploiting standards, ESPRIT P 20693.
http://www.omimo.be/projects/20693/20693.htm

[11] The Transputer Data Book, 2nd ed., SGS-Thomson microelectronics, 1989.

[12] C. Clos, A Study of Non-blocking Switching Networks, Bell Syst. Tech. J. 32 (1953) 406-424.

[13] R.L. Sites, R. Witek et al., Alpha Architecture Reference manual, Digital Press, 1992, ISBN 1-55558-098-X.

[14] Networks, Routers and Transputers, edited by M.D. May, P.W. Thomson and P.H. Welch, pp. 90, ISBN 90 5199 129 0.
http://talisker.pact.srf.ac.uk/~andyj/inmos/bluebook.html

[15] S.Fisher, Low Level Benchmarking of the T9000 Transputer, M.Sc dissertation, the University of Liverpool, February 1995.

[16] M. Joos et. al., An Evaluation of VMEbus PowerPC Based Processor Boards Running the LynxOS Operating System, CERN, Electronic Systems Support, 1996.

[17] Curnow/Wichmann, Computer Journal, Vol. 19#1, Feb 1976.

[18] R. Heeley et. al., The Application of the T9000 Transputer to the CPLEAR Experiment at CERN, Nuclear Instruments and Methods in Physics Research Section A, 368 (1996) 666-674.
http://www.cern.ch/HSI/dshs/publications/t9paper/T9paper_1.html

[19] M.P. Ward, A Transputer Based Scalable Data Acquisition system, Ph.D. dissertation, the University of Liverpool, September 1995.

[20] B. Martin et. al., New Generation Transputer Components for HEP Applications, TRI-93-1, Conference Record of the Eighth Conference on Real-time Computer Applications in Nuclear, Particle and Plasma Physics, Vancouver, 1993.

[21] L3 Collaboration, Letter of Intent, CERN/LEPC 82-5, March 1982.

[22] Alain Masserot, Mise en oeurve et intégration dans l’expérience L3 d’un déclenchement de deuxième niveau avec assemblage de l’événement, développé autour d’un réseau de routeurs dynamiques C104 et de Transputers T9000, Ph.D. discertation, Université de Savoie Laboratoire d’Annecy-Le-Vieux de Physique des Particules, 1995.

[23] R. Bock, F. Chantemargue, R. Dobinson and R. Hauser, Benchmarking Communications Systems for Trigger Applications, ATLAS DAQ note 48, 1995
http://atlasinfo.cern.ch/Atlas/GROUPS/DAQTRIG/NOTES/notes_ps.html

[24] A. Klein, Interconnection Networks for Universal Message-Passing Systems, Proc. ESPRIT Conference ‘91, pp. 336-351, Commision for the European Communities, Nov. 1991, ISBN 92-826-2905-8.

[25] Evaluation and Simulation of Event Building Techniques for a Detector at the LHC, Ph.D. Dissertation, Universität Dortmund, October 1995.

[26] R.W. Dobinson, D.Francis, R. Heeley and J.R. Moonen, Triggering and Event Building Results Using the C104 Packet Routing Chip, Nuclear Instruments and Methods in Physics Research Section A, 376 (1996) 59-66.
http://www.cern.ch/HSI/dshs/Welcome.html

[27] Rudolf Bock and Patrick LeDu, Detector and Readout Specifications for the Level-2 Trigger Demonstrator Program, ATLAS DAQ note 53, June 1996.
http://atlasinfo.cern.ch/Atlas/GROUPS/DAQTRIG/NOTES/notes_ps.html

[28] The Power Challenge Technical Report, Silicon Graphics, 1994.

[29] IBM Systems Journal, Vol. 34, No.2, 1995.

[30] Eric Barton, James Cownie, Moray McLaren, Message Passing on the Meiko CS-2, Parallel Computing 20 (1994) 497-507.

[31] MPI: Message Passing Initiative,
http://www.mcs.anl.gov

[32] Roger Hockney, Michael Berry, Public Inernational Benchmarks for Parallel Computers (Report 1), PARKBENCH Committee Report 1, February 1994.

[33] The Rcube Hardware Specification, Université de Pierre et Marie Curie, Mètholodigie et Architecture de Systémes Informatiques, 1995.
http://humanus.ibp.fr/rcube.html