The Hathi-2

Multiprocessor System

M. Aspnäs, R.J.R. Back, T-E. Malén

Ser. A, No 80, 1989

Inst. för Informationsbehandling
Lemminkäinengatan 14
SF-20520 Åbo, Finland

June 1989

ISBN 951-649-604-0
ISSN 0358-0563

The Hathi-2 Multiprocessor System

M. Aspnäs * R.J.R. Back * T-E. Malén *

June 30, 1989

Abstract

Hathi-2 is a reconfigurable general purpose loosely coupled MIMD multiprocessor system consisting of 100 32-bit IMS T800 transputers and 25 16-bit IMS T212 transputers. Hathi-2 has a distributed switching network and a distributed control system wich contains an interrupt system and hardware support for distributed monitoring. This report describes the architecture of the Hathi-2 system and the design goals of the system. The system software that has been developed for the system is also presented.

1 Introduction

Hathi-2 is a reconfigurable general purpose multiprocessor system built of 100 Inmos T800 transputers. The system can be characterized as a loosely coupled MIMD multiprocessor, with a reconfigurable distributed interconnection network and a modular design. Due to its modularity, Hathi-2 can be expanded by adding more modules, i.e., processor boards, to the system.

The Hathi-2 multiprocessor system was developed in the Hathi project by the Department of Computer Science at Åbo Akademi and the Technical Research Centre of Finland (VTT/TKO) in Oulu. The Hathi project started in August 1986 and ended in December 1988. One goal of the project was to build a massively parallel multiprocessor system based on transputers. The design of the Hathi-2 system started early in 1987 and the system was delivered to Åbo Akademi in April 1988. The Technical Research Centre of Finland in Oulu was responsible for designing and building the hardware system, while Åbo Akademi has designed the system software and application programs for Hathi-2. The theoretical parallel performance of the Hathi-2 multiprocessor system is 150 MFLOPS or 1000 MIPS.

A number of applications have been implemented on Hathi-2 as a part of the project. Among the applications can be mentioned full-text retreival in a distributed database [WS], fluid dynamics [ÖMK], geometric image transformation of satellite data [RRS], cluster identification in a three dimensional data space and parallel execution of production systems [Bom], [Eri]. The hardware design of the the Hathi-2 system is described in [Peh1], [Peh2] and [Äij]. The use of the Hathi-2 multiprocessor system is described in [AsMa].

The reader is assumed to be familiar with the transputer arcitecture and the Occam language in this paper, so this will not be presented here. A presentation of the transputers

*Åbo Akademi, Department of Computer Science, Lemminkäisenkatu 14, SF-20520 Turku, Finland

architecture and its instruction set can be found in [Inm1] and [Inm2]. The programming language *Occam* is presented in [Inm3] and [JG].

The paper is organized as follows: In Section 2 we give an overview of other transputer based multiprocessor systems. In Section 3 we describe the architecture of the Hathi-2 system and the motivations for choosing this arcitecture. Section 4 describes the hardware environment in wich Hathi-2 is operated, i.e., the host computers from which it is accessed and the peripheral units attached to it, while Section 5 describes the software environment of Hathi-2, i.e., the programming languages and other utilities that can be used to write parallel programs for the system and the software developed for the control system. In Section 6 we describe the Transputer Network Topology administrator, an utility with wich the Hathi-2 system can be (statically) reconfigured. Section 7 describes the Transputer Network Monitoring System, with which the user can monitor the activities in a parallel program running on Hathi-2. Finally, in Section 8, we try to evaluate the architecture of the Hathi-2 system and identify its strengths and weaknesses.

2 Related work

Several transputer-based multiprocessor systems have been announced during the last few years. In this section we give a short description of the architecture of two other and somewhat similar transputer-based multiprocessor systems.

2.1 Meiko Computing Surface

The Meiko Computing Surface is a modular, reconfigurable, general purpose transputer-based multiprocessor system, manufactured by Meiko Ltd [BoKe], [BBKPW]. A Computing Surface consists of one or more modules each containing 10 to 40 boards. At least one of the boards in a module is a *local host* board, carrying one transputer, 3 Mb of memory and various interfaces for connection with external devices. The local host is responsible for carrying out maintainance tasks such as controlling the electronic routing network, hardware reset, monitoring for hardware errors and hosting the interactive program development environment. The transputers in a module are connected to the local host through a supervisor bus and the local hosts are connected to each other in a chain using two of the links, thus logically forming a global supervisor bus. The supervisor bus is used as a link-independent debug channel, through which messages can be routed to the standard output device from every processor. The other boards in a module can be *compute boards*, which contain four T800 transputers with 256 Kb to 4 Mb of memory per transputer, *mass store boards* wich carries a single transputer with 8 Mb of memory and a disk and peripheral interface and *display boards* with a single transputer and dual-ported video RAM.

The Meiko Computing Surface uses a custom VLSI switching chip for reconfiguring the topology of the transputers in a system. The interconnection network, formed by the switching elements and the transputer links, allows the transputers in a system to be configured as they were all in the same module. The reconfiguration capability is mainly restricted by the fact that each transputer can be connected to at most four other transputers.

The operating system on the Computing Surface is a multi-user Unix-like operating system. Each user is allocated a smaller, logically independent part of the system, called a *domain*, which is totally controlled by the user. The user logs in to the system and

executes the program development environment on a *user-seat*, which is a transputer dedicated to this task. The user-seats share access to the resources in the system, i.e., to the compute servers (the domains) and the disk servers.

2.2 Esprit P1085 (Supernode)

In Esprit project P1085 a reconfigurable transputer-based multiprocessor system, called the Supernode, has been developed [Nicole]. The Supernode architecture has been commercially exploited by Parsys, which manufactures and sells general-purpose multiprocessor systems based on this architecture.

The Supernode has a hierarchical architecture. The basic building block is a *computing module* consisting of one T800 transputer, 256 Kb to 4 Mb of memory, a control bus interface and a performance monitoring module. Eight computing modules are mounted on a *worker board*. Six worker boards can be plugged in to a switching backplane, which connects the worker transputer links to a custom designed 72-way crossbar switch. Each backplane requires a controller card with one transputer, which is responsible for controlling the switch. Other types of boards can also be plugged into a backplane, i.e., mass storage boards with a disk interface, link buffer cards for extending links away from the backplane or boards with one transputer and a very large amount of memory. A single backplane can support up to 36 transputers, including the contol transputer, disk servers and external links.

Two backplanes can be connected to each other through a switch in a tandem rack, containing up to 72 transputers. For larger systems, several tandem racks can be connected by replacing half of the worker cards in each rack with link buffer cards, whose links are connected to an inter-rack switch. The inter-rack switch is controlled by a transputer, thus forming a three layer switching network. The switching network in the Supernode architecture is universal, i.e., it is able to establish any graph of transputer interconnections. However, the switching network is not nonblocking, but reconfigurable.

All working transputers in the Supernode architecture are connected to a separate low-bandwidth byte-wide *control bus*, which is used for system houskeeping (such as reset, analyse, error and link speed selection), and for sending monitoring data about link and CPU utilization to a controlling transputer. The control bus can also be used for synchronizing a group of modules with the controlling transputer. The maximal data rate on the control bus is around 100 Kb/s.

3 Hathi-2 architecture

Hathi-2 is a transputer based multiprocessor system. A transputer based system generally consists of three main hardware components:

1. A *host computer* system, which can be a personal computer (e.g. IBM-PC AT or Apple Macintosh) or a graphical workstation (e.g. Sun-3). The host computer is a normal computer system with disks, screen and keyboard and an operating system (e.g MS-DOS or Unix). The processor in the host computer (e.g. Intel 80286 or Motorola 68020) is refered to as the *host processor*. The host computer provides I/O to the multiprocessor system and interaction with the user through a *server process*, which executes on the host processor.

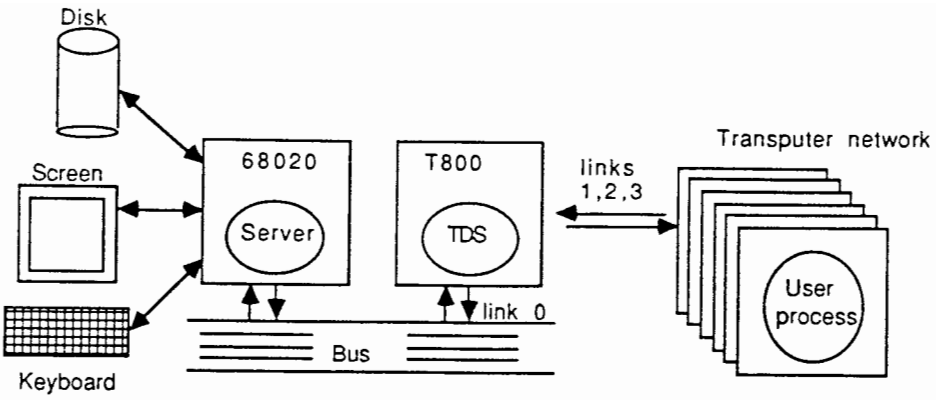

Figure 1: The components in a transputer-based multiprocessor system

2. A *host transputer* , which is installed in the host computer system. The host transputer is connected to the bus of the host computer, usually via one of its links and is thus able to communicate with the host processor. The host transputer is normally mounted on a transputer board (e.g. Inmos B004, Niche NT1000 or Levco Translink), which is plugged into one of the available extension slots in the host computer system. The programming environment TDS and the compilers for transputers are executed on the host transputer and uses the host computer as an I/O unit.

3. A *transputer network* , consisting of a number of transputers connected to each other via links. The transputers can be connected to almost any desired topology (e.g. ring, mesh, tree, cube). One restriction on the topology of the network is that the number of links on each transputer is limited, typically four links per transputer. The parallel program is executed on the transputer network and input and output is handled by the server process running on the host processor.

Hathi-2 extends this basic scheme with some additional features, such as a reconfigurable switching network, a possibility to partition the system between several users and hardware support for distributed monitoring of the system.

3.1 Design goals

When the design of the Hathi-2 system system started in the beginning of 1987, the T800 transputer had recently been published and only preliminary data was available about the C004 crossbar switch. Large general-purpose transputer based multiprocessor systems were not commercially available to the same extent as today.

Before the Hathi-2 system was built, a smaller 16 transputer multiprocessor system, called Hathi-1, was constructed from Inmos B003 boards. This system was used for over one year, and much of the system software was designed on this system, using a Hathi-2 architecture simulator developed in the project.

Hathi-2 was intended to be used as a general-purpose multiprocessor system, mainly for research and program development. It was expected that the users would have very different and even contradicting demands on the system. To meet these demands the

architecture of the system must be flexible, so that each user can adapt the system to his specific demands. Especially important was to give the users a possibility to reconfigure the topology of the system. The system must also be able to support more than one user at a time.

The proposed design should be modular and as simple and straightforward as possible, in order to minimize the costs of building the system and improve its reliability. The architecture of the system should also be fully compatible with the existing program development software (e.g. TDS). This requires that the Reset, Analyse and Error control signals are treated in the same way as in Inmos transputer board products.

The experiences from the first multiprocessor system, Hathi-1, showed that debugging and load balancing of parallel programs was a difficult task and some tools for monitoring the activities in the multiprocessor system was needed. However, monitoring a distributed system produces a large amount of data and can require extensive amounts of computation. In order not to interfere with the main computation in the system, the architecture contains hardware dedicated to monitoring system activities during program execution. Monitoring data need to be processed and presented by a separate subsystem, otherwise monitoring will affect the behaviour of the distributed program.

Based on these requirements the following design desicions were made for the architecture of the Hathi-2 system:

To keep the system design simple and modular, the multiprocessor system is built of *one single type of boards*. The number of processors in the system can thus be expanded simply by adding new boards. To extend the system with peripheral units, every board has a number of input/output links dedicated to this purpose.

The communication links between the processors in the system are connected to each other via a *reconfigurable switching network*. Because there is only one type of boards and all boards are identical, there is no central switching system. The switching network is distributed over the boards and built outof Inmos C004 crossbar switches.

The system is made compatible with the program development tools from Inmos by connecting the control signals Reset, Analyze and Error as a pipeline through all transputers in the system. The system can be shared between a number of users by partitioning the system into a number of independent subsystems. This is done by disconnecting the control signal between the partitions and connecting the control signals from the users host to the partition. This gives the user full control over his partition of the system, but prevents him from affecting other partitions.

To control the distributed switching network, a separate *distributed control system* was built. The control system consists of one additional transputer on each board, connected to each other in a ring. One of the links on the control system transputers is used for sending commands to the C004 switch. The control system can also handle other supervisory tasks, such as monitoring the behaviour of the system. The control system is separated from the rest of the system, i.e. it is controlled by a separate host computer.

Each processor in the system is equipped with some *hardware support for monitoring*. Gathering and forwarding of monitoring data may not affect the computations in the system, and must thus be performed by some hardware dedicated to this task. Monitoring information is forwarded from the calculating transputers to the control system by two additional hardware components: the FIFO buffers and the CPU load meter.

An additional *interrupt subsystem*, based on the transputers EVENT pin, is included in the control system. This can for instance be used to provide a global clock for the system.

3.2 The Hathi-2 board

Hathi-2 consists of 25 identical extended 3E eurocard boards. Each board contains four 32-bit IMS T800 transputers with up to 4.25 Mb of external memory, one IMS C004 crossbar link switch and one 16-bit IMS T212 transputer.

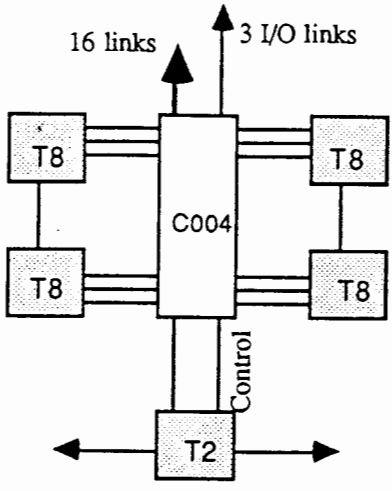

Figure 2: Hathi-2 board architecture

The T800 transputers are connected pairwise to each other via link 0. The three remaining links from each T800 and link 2 of the T212 are connected to the C004 crossbar switch. The C004 is controlled by link 3 of the T212 transputer (see Fig. 2). The T212 controls the setting of the C004 switch by sending commands on the control link. The two remaining links on the T212 (links 0 and 1) are used to connect the T212 transputers into a ring, thus forming the distributed control system.

Of the remaining 19 links on the switch, 3 links are used as I/O links, i.e., to connect users host computers and peripheral units to the system. The remaining 16 links are used to form a statical torus connection between the boards in the Hathi-2 system.

3.3 The distributed switching network

Hathi-2 has a two-level connection scheme, with a static and a reconfigurable level. The static connection scheme is formed by the physical interconnections between the boards. The boards are statically connected to each other in a torus connection, with 4 links between every pair of boards (see Fig. 3). From the switch on every board, 4 links are connected to the upper, lower, right and left neigbour switches respectively. This connection between boards is static and is implemented by a physical wiring on the backplane of Hathi-2. It can only be modified by manually changing the link connections between the boards on the backplane.

The second level of connection is through the C004 switches on each board. This connection scheme allows the processor interconnection topology to be changed by software. Two transputers on the same board can be connected to each other through the onboard switch. If the transputers reside on neighbouring boards, the link connecting the two

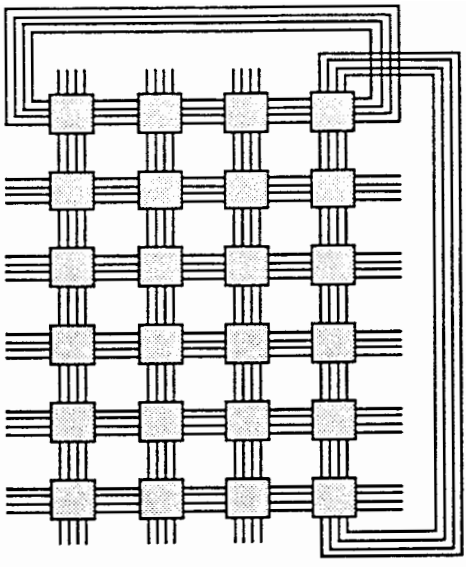

Figure 3: Hathi-2 board connections

transputers goes through the two switches on the corresponding boards, using one of the four links in the static torus connection between the boards. If the two transputers does not reside on neighbouring boards, the link connecting them can be routed through an arbitrary number of intervening switches (see fig. 4 a, b and c). The performance penalty associated with this is a 20% communication capacity reduction for each C004 switch the link is routed through.

3.4 Partitioning the system

Hathi-2 can be used by several users simultaneously. Each user gets an own subsystem, called a *partition*. A partition is allocated to one user and used as a separate multiprocessor system. The user can reconfigure his partition and execute code on it independently of all other users. The smallest unit that can form a partition in Hathi-2 is one board, i.e., four T800 transputers. Thus, Hathi-2 can be partitioned in at most 25 independent subsystems. The largest partition that can be formed consists of 96 T800 transputers. Partitioning is done by a set of manual switches on the backplane of Hathi-2 and can not be done during runtime, as the system has to be rebooted after partitioning.

All T800 transputers are connected into a pipeline by the control signals Reset, Analyse and Error (see Fig. 5). A partition is formed by disconnecting its control signals from the rest of the system, which is done by cutting off the RAE-signals by a manual switch on the backplane of Hathi-2. This isolates the partition from the rest of the system and prevents the RAE-signals to continue to the transputers outside the partition. The user is connected to his partition with the RAE-signals and at least one link from the users host transputer. Partitions must be formed so that the boards in one partition follow each other along the control signal line (see Fig. 5). The board which the user is connected to (the first board in the partition) is called the *master board* and the T212 transputer on

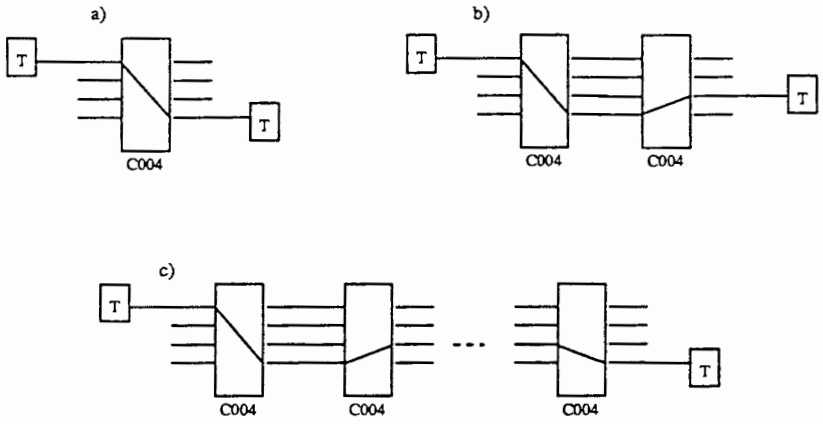

Figure 4: Connecting transputer links via C004 switches

this board is called the *master T212.*

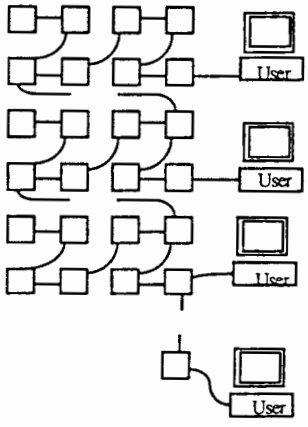

Figure 5: The Reset, Analyse and Error signals

Figures 3 and 5 show that the boards in Hathi-2 are organized as a 4 by 6 torus. One board (the $25th$) is not part of the torus connection, and its links are not connected to any other board. This board is used as a separate partition consisting of 4 T800 transputers. Therefore, Hathi-2 is always partioned in at least two subsystems.

3.5 The control system

Hathi-2 is logically divided into two separate subsystems: the multiprocessor system and the control system. The multiprocessor system consists of 100 T800 transputers. The contol system consists of 25 T212 transputers, the C004 crossbar switches, the monitoring hardware and the interrupt system. The T212 transputers in the control system are connected to form a ring and are controlled by a separate host computer, called the

control system host, (see Fig. 6). The T212 interconection topology is static and cannot be changed. Partitioning Hathi-2 does not affect the control system. The users host transputer is connected via one of the I/O links to the partitions master T212. This allows the user to communicate with the control system software using a set of predefined interface procedures.

The control system has two primary tasks:

1. To control the settings of the C004 switches, i.e., to manage the *reconfiguration* of the system. The reconfiguration software is described in Section 7.

2. To support *monitoring* of the activities in the system, i.e. to forward monitoring data from the T800 transputers to the control systems host. The monitoring software is described in Section 8.

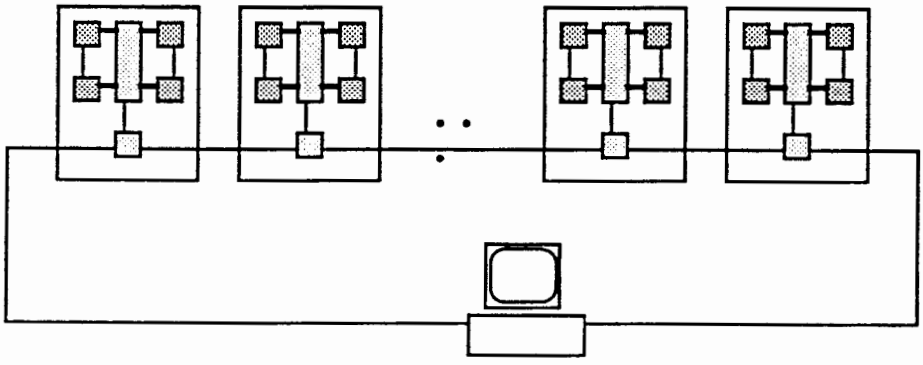

Figure 6: The control system

3.6 Hardware support for monitoring

The Hathi-2 architecture contains hardware support for monitoring the activities in the multiprocessor system. A process running on a T800 in the system can send information about its activities to the controlling T212 on the same board with a minimal overhead on the processor. Monitoring data is sent from a T800 to the T212 via a memory mapped FIFO buffer accessable by the two transputers. The T800 can only write to the FIFO buffer and the T212 can only read from it (see Fig. 7). The FIFO buffer is of size 512*9 bits.

A process running on a T800 can put data into the FIFO buffer by writing a byte (8 bits) to the memory location where the FIFO buffer is mapped. The T212 can get data from the FIFO by reading the buffers memory location. The T212 has an empty-flag for each FIFO, indicating that the FIFO queue is empty. The T800 transputer has a corresponding full-flag, indicating that the FIFO is full. Additionally, when the buffer is full, an interrupt is generated on the T800 (described in Section 3.7). If data is inserted into the FIFO when it is full, old data will be pushed out, thus causing loss of data.

When monitoring activities in a partition, a synchrounous clock pulse, called a *sync interrupt*, can be generated by the master T212 in the partition. The sync interrupt

reaches all processors in the partition simultaneously, both the T800 and the T212 transputers. It is used for generating a global clock pulse, dividing the time into short intervals. On each sync interrupt the *interval bit* on the transputer is complemented. The interval bit is used as the $9th$ bit in the FIFO, thus identifying which time interval the monitoring data in the FIFO belongs to. The T212 can also read the status of the interval bit from a register in its memory.

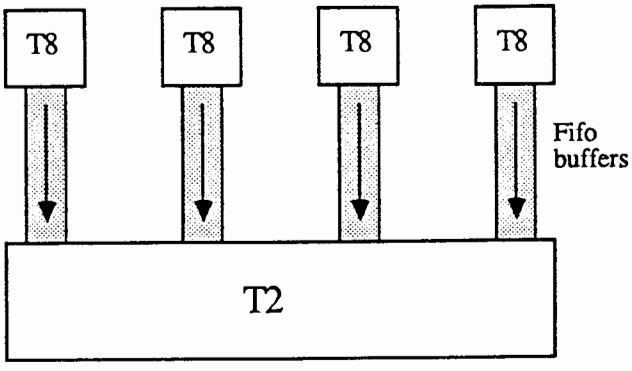

Figure 7: The FIFO buffers

Hathi-2 also contains hardware support for measuring the load of the CPUs. This is done by observing the activity on the transputers bus. A high activity on the bus indicates that the CPU is busy, and a low activity indicates that it is idle. The lowest address line (MemAD2) on the transputers memory interface is connected to a monostable multivibrator, which controls a 24 bit counter. On a rising edge on the bus line, the counter is incremented. If the counter is continously enabled, it overflows after 3.36 seconds. For each sync interrupt, the 16 most significant bits of the counter are stored into a counter data register, which can be read by the T212 transputer.

Monitoring data is gathered from the T800 transputers by the onboard T212, which reads data from the FIFO buffer and the processor-load register. The monitoring data is then forwarded along the T212 ring to the control systems host, where it can be analysed, stored and presented to the user.

3.7 The interrupt system

The transputer architecture only supports one type of external interrupt, called an EVENT. The interrupt must be served by a process waiting on input from the EVENT pin. If there is no process waiting on an EVENT, the interrupt will have no effect on the processor. Hathi-2 has an extended interrupt system which is implemented using the transputers EVENT pin. On Hathi-2, a T800 transputer can receive three different interrupts, called *Full Flag (FF)*, *T2int* and *Sync*. Upon all interrupts an EVENT signal is generated to the transputer, and the process waiting for an event is scheduled. This process must then read an interrupt status register to identify the cause of the interrupt.

The *FF* interrupt is raised by hardware when the FIFO buffer on a transputer becomes full. Additionally, the full flag is also set on the T800 transputer. The process handling the interrupt can then take measures to avoid loss of data in the FIFOs.

The *T2int* interrupt is an interrupt from the T212 transputer on the same board. The T212 can send interrupts to any of the fourT800 transputers on the same board.

The *Sync* interrupt is generated by the master T212 in a partition and reaches all transputers in that paration. It is used by the monitoring software (described in Section 8) for generating synchronization signals and global clock pulses.

4 Hathi-2 hardware environment

The Hathi-2 system is located in the Department of Computer Science at Åbo Akademi. It is installed in a standard instrument cabinet, containing three 19 inch racks, each with room for 14 boards. The cabinet contains the multiprocessor system, which consists of 25 boards, power supply and fans for cooling. The system consumes about 500 W of electric power and is operated in a normal laboratory environment.

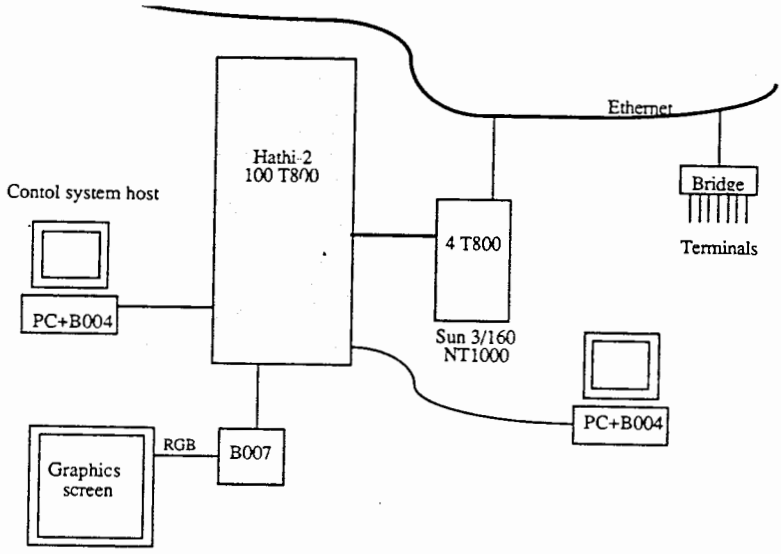

Figure 8: Hathi-2 hosts and peripheral units

In this section we present the host computers that can be used in the Hathi-2 system and the peripheral units that can be accessed from it. All connections between Hathi-2 and other units are via transputer links. The equipment connected to Hathi-2 must be located in the same room as the Hathi-2 system, as the maximum length of a link cable running on 10 Mbits/s is 6 meters and for links running on 20 Mbits/s is 3 meters.

4.1 Host computers

Hathi-2 is used as a back-end computing resource. The user edits, compiles and links his programs on a host computer. The users host must be equipped with at least one transputer to be able to run software which loads programs onto and controls the Hathi-2.

The following host computers are connected to the Hathi-2 system:

Sun 3/160M : Hathi-2 is connected to a Sun-3/160M workstation. This is equipped with a Niche NT1000 transputer board, which contains four T800 transputers, each

with 2 Mb of memory. The Sun is connected to the local Ethernet network in Åbo Akademi, which in turn can be accessed from the national network connecting the Finnish universities, as well as from other international networks. This means that Hathi-2 can be accessed from any terminal, regardless of where the user physically is located.

IBM-PC AT : Two IBM-PC AT computers with Inmos B004 transputer boards are connected to Hathi-2. One of them is used as the host computer in the control system (see Section 3.5), and the other is used as a users host.

Macintosh II : An Apple Macintosh II with a Levco transputer board will also be used as a host computer.

Normally, Hathi-2 is partitioned into five partitions, of which four are connected to the host transputers (nap0 to nap3) in the Sun workstation and one is connected to an IBM PC host. The standard partitioning of Hathi-2 is shown in Figure 9. The numbers indicate how many T800 transputers the partitions contain.

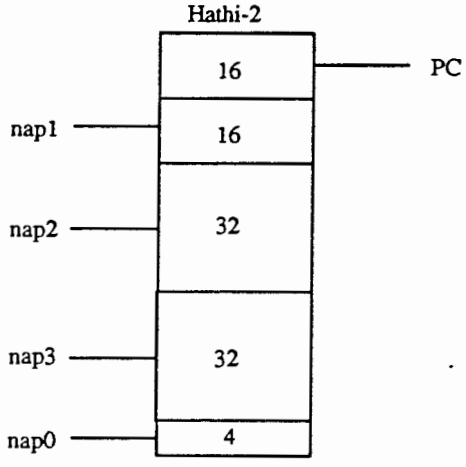

Figure 9: Hathi-2 standard partitioning

4.2 Peripheral units

The following peripheral units are attatched to Hathi-2 and can be used in programs running on the system.

Graphics system : The graphics controller is an Inmos B007 graphics board containing one T414 transputer with 512 Kb of dual-ported video RAM, 512 Kb of program memory and an IMS G170 colour palette. The resolution of the graphics controller is 512 by 512 8-bit pixels. A 20 inch Salora 445A RGB monitor is used as a graphics display.

Mass storage system : An Inmos B005 20 Mb hard disk can be directly connected to any of the I/O links in Hathi-2. The disk can be used for storing intermediate data or results from a computation. The hard disks in the host computers (Sun, PC and Mac) can also be used as mass storage systems.

5 Hathi-2 software environment

Hathi-2 is normally programmed in Occam, using *TDS (Transputer Development System)* [Inm4]. TDS is is an integrated programming environment for transputer networks and contains a folding editor, an Occam-2 compiler, loader, configurer, debugger and several other utilities. TDS is executed on the host transputer and I/O to the host system is provided through a server process running on the host processor.

Occam programs can also be written using the *stand-alone Occam toolset* [Inm5], which consists of a standalone Occam compiler, linker, configurer and several other utilities. Using this system, occam programs can be developed under the host computers operating system (MS-DOS, Unix, etc.) without using the more closed programming environment TDS.

Hathi-2 can also be programmed using C [Inm6] or Fortran [Inm7]. Parallel programs are written as a collection of sequential processes, which can communicate with each other by sending and receiving messages through channels. Communication is implemented as calls to predefined procedures. To execute the processes in parallel, they have to be called from an *occam harness*, which is an Occam program skeleton in which the parallel structure of the task is described. There is also a *parallel Fortran* compiler, in which the Fortran language has been extended with facilities for process creation and communication via channels.

Also available on Hathi-2 are the distributed operating systems *Helios* [Per] and Trollius [Bra], [Bur]. The operating systems can be executed on a partition of Hathi-2, using either a PC or the Sun workstation as a host processor.

5.1 The control system software

The control system in Hathi-2 is responsible for carrying out system administration and service routines. The user can request service from the control system by calling service request procedures, which communicate with the control system by sending messages over the link connecting the users host to the control system. The system service functions do not impose any overhead on the computation in the multiprocessor system, as they are executed by the T212 transputers in the control system.

The user is connected to the control system through a link from the users host transputer to the master T212 transputer in the partition. Communication between the user and the control system follows a specific protocol and should only be accessed through the service request procedures.

The control system software is based on a general message-passing system, which handles transparent forwarding of messages between processes in the control system. This is implemented by a simple communication kernel executing on each transputer in the control system. The communication kernel supports any type of messages and any number of processes. Furthermore, the kernel is simple and efficient, both with regard to execution time and memory requirements.

The kernel forwards messages from a source process either to one destination process or to a group of processes using broadcasting or multicasting. Messages can be of any type and length. A message consists of an address header, the message data and a end-of-message indicator. Addresses consist of two fields: a processor number identifying a processor in the control system and a process number identifying one of the system service processes executing on the control system.

Messages flow simultaneously in both directions in the ring. The communication kernel calculates the shortest route to the destination node and forwards the message in that direction. The kernel prevents starvation of any single process by alternating the priority for communication between messages to be forwarded from a neighbouring process and messages from a service process on the same processor. The message passing scheme is deadlock free.

Currently, there are three types of service processes executing on each T212 transputer in the control system:

User host interface : The user host interface process handles communication between the users host and the control system. It continously listens to messages from the user or responses from the control system and forwards these either to the users host or to the communication kernel. The process checks that all communication over the link between the users host and the control system conforms to the defined protocol.

Switch controller : The switch contoller process controls the setting of the C004 switch on the board by sending commands to it over the switch control link. The process does not perform any extensive checks on the switch commands, but merely passes on the received commands to the C004 switch. The switch controller process is a part of the Transputer Network Topology administrator software.

Monitor controller : The monitor controller process handles generation of interval synchronization signals and gathering of monitoring data from the FIFO queues and from the load meter. The process polls the FIFO registers and the load meter register and collects monitoring data from one time interval to a packet, which is sent via the message passing system to a node where the data can be stored in a file. The monitor process is part of the Transputer Network Monitoring system.

6 The Transputer Network Topology Administrator

The Transputer Network Topology (TNT) administrator is a utility that supports the reconfiguration capabilties of the distributed switching network in Hathi-2. The TNT administrator allows the user to change the interconnection of the processors in the partition. Only a static reconfiguration capability is supported, i.e., the network configuration can be changed only between separate program executions, not during runtime.

6.1 Program structure

The TNT administrator software consists of three different parts: the *user interface* which is executed on the users host transputer, a *switch controller process* executing on each transputer in the control system and the *central topology administrator* which is executed on the control systems host.

User interface : The user interface process executes on the users host transputer. It presents a menu of topology alternatives to the user, and sends a description of the chosen topology to the control system. The user interface process checks the size of the partition it is connected to and allows the users to select only those alternatives that can be established on this partition. The descriptions of the topologies are stored as files on the users host.

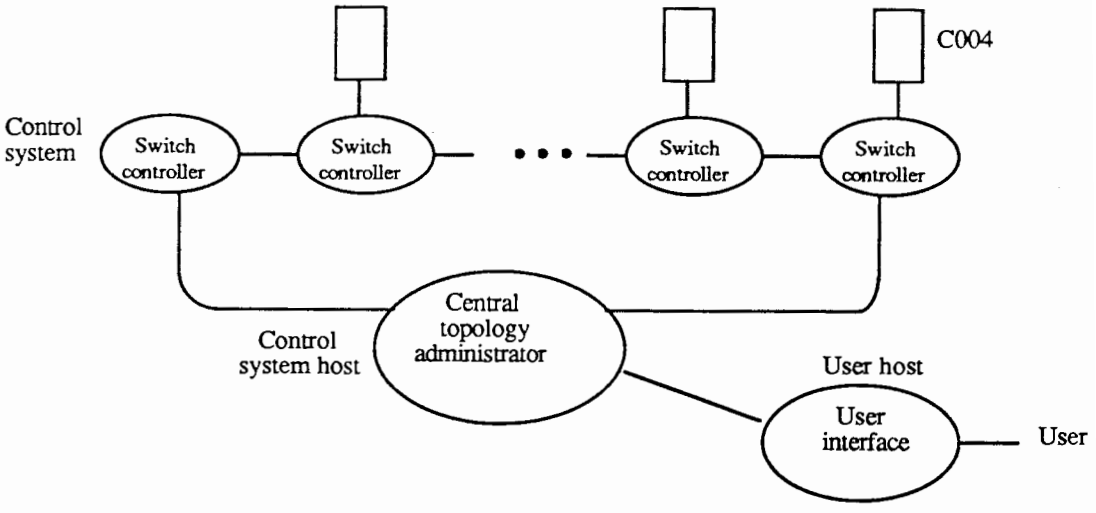

Figure 10: The structure of the TNT administrator

Switch controller : The switch controller process is one of the system service processes in the control system. It receives switch control commands from the message passing system, and forwards them to the C004 switch.

Central topology administrator : The central topology administrator process is executed on the control system host. This process is responsible for keeping record of all established link connections in Hathi-2. It receives topology descriptions from the message passing system and translates theese to switch commands, using a shortest path routing algorithm. It also takes into consideration the limitations imposed by the limited number of links between neighbouring boards. If it is not able to establish a topology, it sends an error message back to the user.

The user selects a processor interconnection topology from a library of predefined topologies. The user interface process sends a reconfiguration request to the central topology administrator and waits for an acknowledgement. If the acknowledgement is positive, the process sends the topology description via the message passing system to the central topology administrator process on the control systems host, otherwise it cannot reconfigure its partition. The description of the selected topology is then interpreted and translated to switch commands. These are sent via the message passing system to the switch controller processes, which in their turn pass them to the C004 crossbar switches. The tool does not allow a user to change the interconnection of other processors than those belonging to his own partition. The central topology administrator queues the incomming reconfiguration requests and serves one requst at a time, thus allowing several users to request reconfiguration simultaneously.

The TNT administrator is menu driven, i.e., the user selects what to do from a menu. The tool is designed to have minimal interaction with the user. Therefore, it automatically checks the size of the users partition and displays only those topologies that can be established on the partition.

The user can extend the topology library with new topology descriptions. These consist of a number of connection statements where a connection statement specifies that two processors should be connected to each other with a link. A connection statements has the form:

Connect processor 2 link 1 to processor 5 link 3

The processor numbering scheme used in the connection statements is obtained from a partition map, which assigns a unique logical number to all processors in a partition. The link numbers corresponds to the transputer link numbers, 0 - 3.

6.2 Link routing

The routing mechanism used in the TNT Administrator is a simple routing algorithm based on Djikstras shortest path algoithm [Sch]. The algorithm takes as input a number of node pair connections, described by connection statements, which are to be established on the Hathi-2 switching network. For each connection the link routing algorithm finds the shortest path through the network. The algorithm is sequential, i.e. the connections are established sequentially.

The distributed switching network allows every processor to be connected to any other processor in the network. However, there are limitations on which connections can be established simultaneously, due to the limited number of links available between neighbouring boards. It is possible that processors on different boards cannot be connected because there is no path available between the respective boards, i.e., the path is blocked by other connections using this path.

The algorithm uses no backtracking or heuristics to find the optimal path through the network. It will either find a connection between a pair of nodes or be unable to connect the nodes, depending on which connections are already established in the switching network. The connections are initially sorted in ascending order on the minimum distance between the nodes, to allow the shorter connections to be established first. The TNT administrator checks that all connections are established within a users partition and thus prevents a user from corrupting another partitions topology.

7 The Transputer Network Monitoring System

The Transputer Network Monitoring system is a utility with which the user can *monitor* the performance of a parallel program executing on Hathi-2. Monitoring is done by collecting information about the utilization of the CPU and the communication links from the processor load meter and the FIFO buffers described in Section 3.6.

During monitoring, time is divided into equally long time intervals by a global synchronization signal, which is generated by the master T212 transputer in the users partition. Monitoring data is sampled once every time interval, and sent via the message passing system to a hard disk for storage. This creates a trace of the execution, which describes the dynamic behaviour of the executed program. The monitoring data is presented to the user in a graphical form.

Information about processor load is generated automatically by the load meter hardware and does not affect the computation on the T800 transputers. Information about link communication has to be written by the T800 transputers to the FIFO buffers, and thus causes some overhead on the computation. The user has to insert som additional code into his program to write monitoring information into the FIFO buffers. This information does not necesserily have to concern link communication, it can also report on other events the user wishes to monitor during program execution. However, the amount of additional computation introduced by these monitoring statements is very small and can be neglected.

Monitoring can be done either as off-line monitoring or as a on-line monitoring. In off-line monitoring, data is gathered during the program execution and the generated data is examined after execution. This allows the user to examine the generated data on his own pace by stepping through the monitoring data. In on-line monitoring, data is presented graphically or numerically during program execution without any intermediate storage. On-line and off-line monitoring can also be combined, so that monitoring data is both presented and stored in a file simultaneously.

7.1 Program structure

The Transputer Network Monitoring system consists of four different parts: a *monitoring process* executing on each T800 transputer being monitored, a *monitor control process* executing on each T212 transputer in the control system, a *collector process* executing on the control system host and a *presentation process* executing on the users host transputer (see Fig. 11).

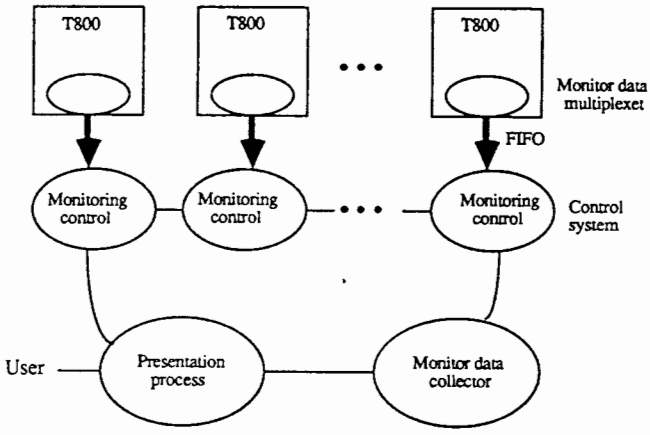

Figure 11: The Transputer Network Monitoring system

The monitor data multiplexer : The monitor data multiplexer process runs in parallel with the monitored user program on the T800 transputers. The process acts as a multiplexer, which collects raw monitoring data, e.g., messages about link usage, from the user processes and outputs the data as packets to the FIFO buffer once every time interval.

The monitor controller : The monitor control process is one of the system service processes in the control system. The process continously polls the FIFO buffers and reads the processor load meter register every time interval. The monitoring data is sent via the message passing system to the collector process. The monitor control process executing on the master T212 transputer in the partition is also responsible for generating a synchronization signal every time interval. The length of the interval can be varied from 50 milliseconds to 3 seconds.

The collector : The monitor data collector process runs on the control system host, or on some other processor connected to a hard disk unit. The process collects the monitoring data sent from the monitor control process and saves the data onto

disk. The data file can be retrieved by the presentation process on the users host transputer for presentation to the user.

The presentation process : The presentation process presents the monitoring data to the user after the execution. This is thus the second phase in off-line monitoring. The monitoring data is read from the hard disk and the neccesary calculations are performed on the data. The user can for instance wish to view the results of the monitoring with a longer time interval than the sampling interval. In this case, data from a number of sampling intervals are read from the disk and the mean values for the utilization degrees are calculated. The presentation is based on the processor structure of the monitored program, i.e., the utilization of the transputers CPUs and links are shown on a graphical representation of the processor topology.

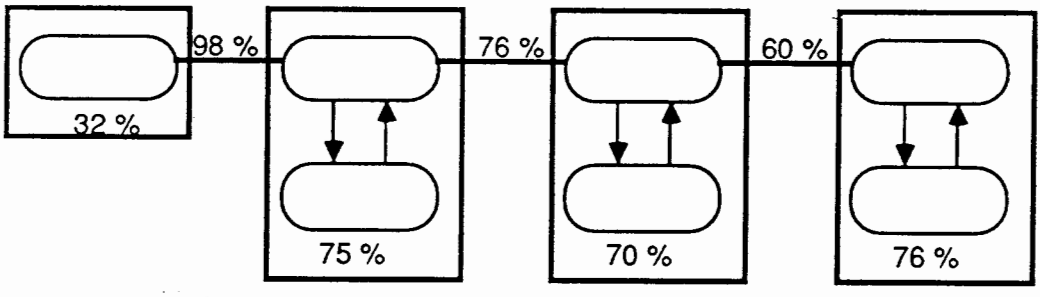

Figure 12: Presentation of monitoring data

7.2 Monitoring a program

To monitor the resource utilization of a parallel program executed on Hathi-2, the user has to modify his program by adding some monitoring code to it. First, a multiplexing monitor process has to be executed in parallel with the other processes on each monitored T800 transputer. Second, the user has to call a pair of predefined procedures for every communication statement that is to be reported to the monitoring system. The procedures are always called pairwise, one before the communication statement and one after the communication statement.

To monitor a communication statement *C ! X* (or eqvivalently for an input statement *C ? X*) the following code has to be inserted into the users program:

StartCommunication (linknumber)
C ! X
EndCommunication (linknumber, SIZE(X))

The call to the procedure *StartCommunication* will start a timer for the channel specified by the parameter *linknumber*. The value of this timer is read by the call to *EndCommunication*. The time for the communication and the number of bytes sent over the channel

is passed as a message to the multiplexor process on the monitored T800 transputer. The communication time and number of transfered bytes is added together for each time interval, and then sent through the FIFO buffer to the control system. Time intervals are controlled by the *interval* interrupt signal generated by the control system.

8 Conclusions and future work

The experiences with using the Hathi-2 system have this far been mainly positive. The system has been in use for about one year, and no hardware faults has yet occured. This indicates that the transputer technology used in Hathi-2 is very reliable.

The system has proved to be very flexible to use, as it is possible for the user to adapt the system to his demands. The flexibility of the system is based on the following properties of Hathi-2:

1. Hathi-2 can be used from different types of host systems (e.g. Sun, IBM-PC, Macintosh). This gives the user a possibility to choose the environment he thinks is best suited to this demands. This also gives the user access to a large number of different programming environments and languages for transputer based systems.

2. Hathi-2 can be partitioned into a number of smaller independent subsystems and used by several users at the same time. This makes it possible to give a larger number of users access to the system and to make more efficient use of it. This is a very important feature, as Hathi-2 is mainly used for program development and the users are normally satisfied with a small number of transputers.

3. The size of the partitions can be varied. Normally, the Hathi-2 system is partitioned into five partitions, of which the largest partition has 32 T800 transputers. However, for larger test runs a partition with up to 96 T800 transputers can be formed.

4. The Hathi-2 system can easily be extended. More processors can be added to the system just by adding new boards. To maintain the torus connection between the boards, the system should be incremented in steps of four boards, i.e., 16 T800 transputers. However, if new boards are added as separate partitions (like the $25th$ board), the system can be enhanced in steps of one board (4 T800 transputers). Also, commercially available transputer hardware, like graphics systems and hard disks, can be connected to the Hathi-2 system via the I/O links.

5. Most commercially available software for transputer based systems can be executed on Hathi-2. Basically, the system consists of a number of transputers connected to each other via links. The additional features in the Hathi-2 architecture, like the distributed switching network and the control system, do not have to be taken into consideration by a program running on Hathi-2.

In the future, the Hathi-2 system will be upgraded with more memory for the T800 transputers. Each T800 transputer is likeley to have 1.25 Mb of memory, raising the total memory capacity of the system from 25 Mb to 125 Mb. Furthermore, a number of hard disks will be added to Hathi-2. The disks will be directly connected to an I/O link in Hathi-2 and will not be dependent on a host computer, thus allowing much faster disk access.

A number of program development tools are currently being developed for the Hathi-2 system. The aim of this development is to make the system easier to use and to program. Under development is a *mapping tool*, that accepts a description of a parallel program in form of a processor graph, and automatically maps the program onto the processor network and connects the processors into the desired topology.

The development of the user interface to the *monitoring tool* will also continue. The monitoring information gathered via the FIFO buffers in the control system will be presented to the user by a tool that allows the user to browse through the monitoring data, which is presented in a graphical form.

A similar tool under development is the *program animation tool.* An animation of a parallel program is a graphical visualization of the execution of the program. This tool allows the user to follow the execution of a parallel program and serves as a high level debugging tool.

All these tools will be integrated under a common graphical user interface, executing on a Sun graphical workstation.

Acknowledgements

The Hathi-2 multiprocessor system was designed and built in the Hathi project, wich was financed by The Technology Development Center (TEKES), The Academy of Finland, Åbo Akademi and The Technical Research Centre of Finland (VTT). Part of the work has been done in the FINSOFT III research program, financed by TEKES. The authors wish to thank Kari Leppälä, Kari Pehkonen and Kyösti Rautiola at VTT/TKO in Oulu and Tapani Äijänen at Jyväskylä Institute of Technology for their central contribution in developing the Hathi-2 hardware system.

The authors also wish to thank everybody that have participated in the software development for the Hathi-2 system: Tom Björkholm, Jonny Boman, Edward Eriksson, Jens Granlund, Pekka Kuusela, Stefan Levander, Yngve Nyman, Aarne Rantala, Antti Raunio, Kaisa Sere, Ulla Solin, Lena Ståhl, Dan-Johan Still, Marina Waldén, Patrik Waxlax, Sami Viitanen and Göran Öhman.

References

[ALM] M. Aspnäs and T-E. Malén, Hathi-2 users guide, Reports on Computer Science, Ser B, No. 6, Åbo Akademi, 1989.

[Ba] R.J.R. Back (ed.), Final report on the Hathi-project 1986-1988, Åbo Akademi, 1989 (in preparation).

[BBKPW] K.C. Bowler, A.D. Bruce, R.D. Kenway, G.S. Pawley, D.J. Wallace, Exploiting highly concurrent computers for physics, Physics Today, Oct. 1987.

[BoKe] K.C. Bowler, R.D. Kenway, Physics on parallel computers, Part 1: the new technology, Contemp. Phys., 28:6, 1987, pp. 573–598.

[Bom] J. Boman, Computer architectures for execution of production systems, M.Sc. thesis, Åbo Akademi, Department of Computer Science, 1988 (in swedish).

[Bra] M. Braner, Trollius users reference manual, Parallel Systems Group, Cornell Theory Center, 1988.

[Bur] G. Burns et al, Trillium operating system, Proc. Third Conference on Hypercube Concurrent Computers and Applications, ACM, 1988, pp. 374-376.

[Eri] E. Eriksson, Parallel implementation of the rete-algorithm, M.Sc. thesis, Åbo Akademi, Department of Computer Science, 1988 (in swedish).

[Inm1] Inmos Limited, *Transputer Instruction Set: a compiler writers guide*, Prentice–Hall, 1988.

[Inm2] Inmos Limited, *Transputer Reference Manual*, Prentice–Hall, 1988.

[Inm3] Inmos Limited, *occam 2 Reference Manual*, Prentice–Hall, 1988.

[Inm4] Inmos Limited, *Transputer Development System*, Prentice–Hall, 1988.

[Inm5] Inmos Limited, occam Toolset, User Manual, Inmos Limited, 1987.

[Inm6] Inmos Limited, 3L C Reference Manual, Inmos Limited, 1987.

[Inm7] Inmos Limited, 3L Fortran Reference Manual, Inmos Limited, 1987.

[JG] G. Jones and M. Goldsmith, *Programming in occam 2*, Prentice–Hall, 1988.

[KBW] A. Kilpinen, T. Björkholm and T. Westerlund, Parallel calculation of a packed bed reactor, Proc. 8th IASTED International Conference, Grindelwald, Switzerland, 1989.

[Lev] S. Levander, A monitoring tool for a multiprocessor system, M.Sc. thesis, Åbo Akademi, Department of Computer Science, 1989 (in swedish).

[Mal] T-E. Malén, A configuration tool for a multiprocessor system, M.Sc. thesis, Åbo Akademi, Department of Computer Science, 1988 (in swedish).

[Niche] Niche Data Systems Inc., Niche NT1000 System Reference Manual, 1988

[Nicole] D.A. Nicole, Esprit Project 1085, reconfigurable transputer processor architecture, Proc. CONPAR 88, Brittish Computer Society, Parallel Processing Specialist Group, UMIST, Manchester, 1988.

[Peh1] K. Pehkonen, Hathi-2 technical document, Technical Research Centre of Finland, Computer Technology Department, Oulu, 1988 (in Finnish).

[Peh2] K. Pehkonen, A dynamically reconfigurable parallel computer Hathi-2, Licentiate thesis, University of Oulu, Department of Electrical Engineering, 1989.

[Per] Perihelion Software Limited, *The Helios Operating System*, Prentice-Hall, 1989.

[Rau] K. Rautiala, , M.Sc. thesis, 1988 (in Finnish). ???

[RRS] A. Rantala, A. Raunio and D-J. Still, Some parallel implementations of a geometric image transformation, Proc. SCIA 89, The 6th Scandinavian Conference on Image Analysis, Oulu, Finland, 1989.

[Sch] M. Schwartz, *Telecommunication Networks Protocols, Modelling and Analysis*, Addison-Wesley, 1987.

[3L] 3L Ltd., Parallel Fortran User Guide, 3L Ltd., 1988

[WS] M. Walldén and K. Sere, Free-Text Retreival on Transputer Networks, Microprocessors and Microprocessing, Vol. 13, No. 3, April 1989, pp. 179 - 187.

[Äij] T. Äijänen, Distributed Interconnection of a Reconfigurable Multicomputer System, Microprocessing and Microprogramming, 3-1988, pp. 243-246.

[ÖMK] G.A. Öhman, T-E. Malén and P. Kuusela, Numerical Fluid Flow and Heat Transfer Calculations on Multiprocessor Systems, Heat Engineering Laboratory report 88-3, Department of Chemical Engineering, Åbo Akademi, 1988.

Rapporter från **Åbo Akademi** – Informationsbehandling & Matematik

Serie A

1988

57. *R.J.R. Back*, Refining Atomicity in Parallel Algorithms.
58. *R.J.R. Back, R. Kurki-Suonio*, Decentralization of Process Nets with Centralized Control.
59. *Aimo A. Törn*, Models of software accumulation.
60. *Aimo A. Törn*, PICA - A graphical program development tool.
61. *Aimo A. Törn*, Parallel Monte Carlo with application to global optimization.
62. *Göran Högnäs*, Invariant Measures and Random Walks on the Semigroup of Matrices.
63. *Marina Walldén, Kaisa Sere*, Design and Implementation of Full-Text on Transputer Networks.
64. *R.J.R. Back, Kaisa Sere*, Stepwise Refinement of Parallel Algorithms.
65. *R.J.R. Back, Kaisa Sere*, An Exercise in Deriving Parallel Algorithms: Gaussian Elimination.
66. *Aarne Rantala, Antti Raunio, Dan-Johan Still*, A Multiprocessor System for Fast Geometric Ima Transformation.
67. *Kaisa Sere*, Transforming Communication Topology in Distributed Algorithms.
68. *R.J.R. Back*, Data Refinement in the Refinement Calculus.
69. *Kaisa Sere*, Deriving Action Systems for Processor Farms.
70. *Hong Shen*, A Fast Parallel Algorithm for Integer Sorting.
71. *Göran Högnäs*, A Note on the Semigroup of Analytic Mappings with a Common Fixed Point.
72. *Hans-Jürgen Sebastian, Tilo Brock, Stephan Bönewitz*, Modelling and a First Realization of a T for Configuration Expert Systems.

1989

73. *Aimo Törn*, An Efficient Procedure for Determining the Enabled Set.
74. *Göran Högnäs*, Nonlinear Autoregressive Processes.
75. *Mats Aspnäs, R.J.R. Back, Reino Kurki-Suonio*, Efficient Implementation of Multi-process Hands king on Broadcasting Networks.
76. *Lena Ståhl, R.J.R. Back*, An Implementation of Multiprocess Handshaking on Transputer Netwo
77. *R.J.R. Back, J. von Wright*, Duality in Specification Languages: a Lattice-theoretical Approach.
78. *R.J.R. Back, Kaisa Sere*, Stepwise Refinement of Action Systems.
79. *Hong Shen*, Mapping Parallel Programs onto Transputer Networks.
80. *Mats Aspnäs, R.J.R. Back, Tor-Erik Malén*, The Hathi-2 Multiprocessor System.
81. *Jan Komorowski*, Synthesis of Programs in the Framework of Partial Deduction.

Serie B

1987

5. *Marina Walldén*, A Case Study: Performance of a Distributed Algorithm.

1989

6. *Mats Aspnäs, Tor-Erik Malén*, Hathi-2 Users Guide, version 1.0.