The Objective: A Transputer Based General Purpose Parallel Computer

C.R. Jesshope, I. McNally

Introduction

Programs written for parallel machines are seldom portable between machines of different architectures or even similar machines which differ in the number or configuration of their processors. In particular, programs created to run on a network of transputers will usually have to be manually reconfigured for running on any new network[1].

We wish to create a more general purpose parallel machine for which programs can be written which are independent of network topology. To achieve this the interprocess communications must be re-structured, such that the locations of communicating processes are invisible to the user. Thus the program does not change whether two communicating processes are on the same processor, adjacent processors or even widely spaced processors.

A number of software based systems have been developed to route messages over networks of transputers, giving the virtual *any to any* connectivity required for a general purpose machine. Unfortunately as these systems are scaled up, more and more processing time at each node is taken up in the routing of messages which are *en route* for other nodes. This increase in 'through routing' results in less processor time available for computation.

We propose a system which uses considerably less processor time by employing a communication co-processor for every node in a two dimensional grid of transputers. The through routing of messages is then restricted to the co-processors in the communications network. This is the machine we call the 'Objective'.

The Objective

In the 'Objective' both the processing nodes and the communication nodes are transputers:

- The communication transputers are T414s, providing integer processing, 2k bytes of on-chip RAM and four bi-directional communication links. These transputers are not connected to any external memory.

- The processing transputers are T800s which are similar to the T414s with an additional 2k bytes of RAM and a floating point unit on-chip, together with a large amount of external memory for programs and data.

The routing strategy employed by the Objective divides the messages amongst four 'virtual networks' according to the direction of message travel, one network each for (+X,+Y),(-X,-Y),(-X,+Y) and (+X,-Y). This strategy can be proven deadlock free and also permits adaptive routing around hot spots [1]. Each transputer link carries two independent channels, one in either direction, allowing two virtual networks to be mapped onto a single grid of T414s without multiplexing channels. We use two planes of T414s, the first carries the (+X,+Y) and (-X,-Y) networks and the other carries the (-X,+Y) and (+X,-Y) networks. This gives us added concurrency and the virtual network independence we require, without the overheads of complex channel multiplexing.

All four bi-directional links of each T414 are used in the routing of messages in the routing plane. In order to get messages in and out of the routing plane, a fifth link must be grafted onto each T414 for communication with the corresponding T800 in the processing plane. This fifth link is a memory mapped C012 link adaptor, normally used to make connections between transputer networks and alien processors. The fifth link is not supported by Direct Memory Access, as a result it is slower than the other four links and takes up more of the T414's processing time. The effect of this will be less significant in larger networks where the proportion of through routed messages to arriving or sourced messages is greater.

Figure 1 shows the connections between the processing and routing transputer for a section of the array. Note that only two links from each T800 are required for connection to the T414s in the routing planes, leaving two for connection to peripherals or for other usage.

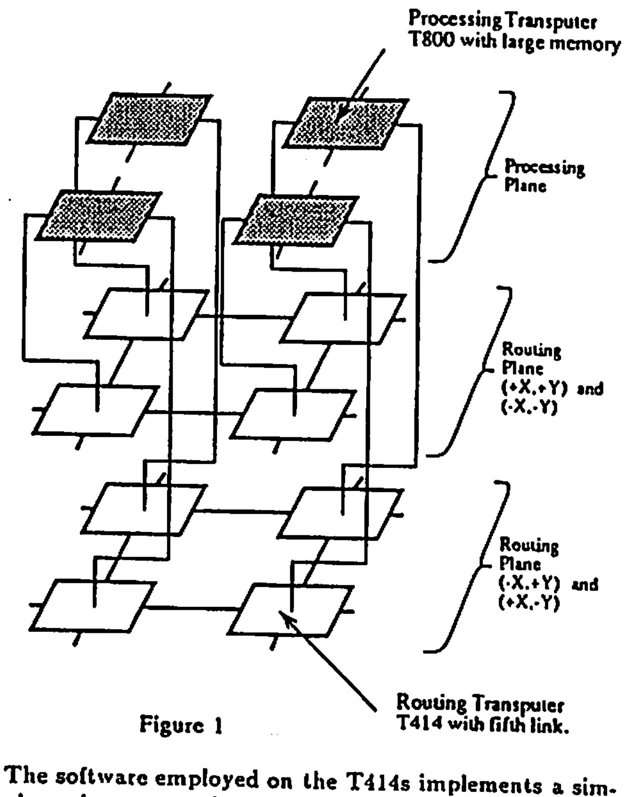

Processing Transputer T800 with large memory
Processing Plane
Routing Plane (+X,+Y) and (-X,-Y)
Routing Plane (-X,+Y) and (+X,-Y)
Routing Transputer T414 with fifth link.

Figure 1

The software employed on the T414s implements a simple packet store and forward algorithm with the direction of routing being determined by the packet address and the availability of a free output channel, giving adaptive routing. Within each T414 the packet buffers for the

[1]One exception to this is a program running under a process farm paradigm, requiring no inter-slave communications.

two virtual networks are kept seperate. Thus the independence of the two virtual networks, required to avoid deadlock, is maintained. Both the packet size and the algorithm are limited by the 2k bytes of storage available on the T414.

Implementation

The system is being implemented using Dowty's *Chiprack* [2] stackable chip carrier technology. The system is built as a number of chiprack stacks on a pcb mother-board. Each stack contains all the components for the section of the array shown in figure 1. A stack consists of a number of Chiprack Leadless Chip Carriers (CLCCs) interleaved with Chiprack Connectors. Figure 2 shows the chiprack stack and its components.

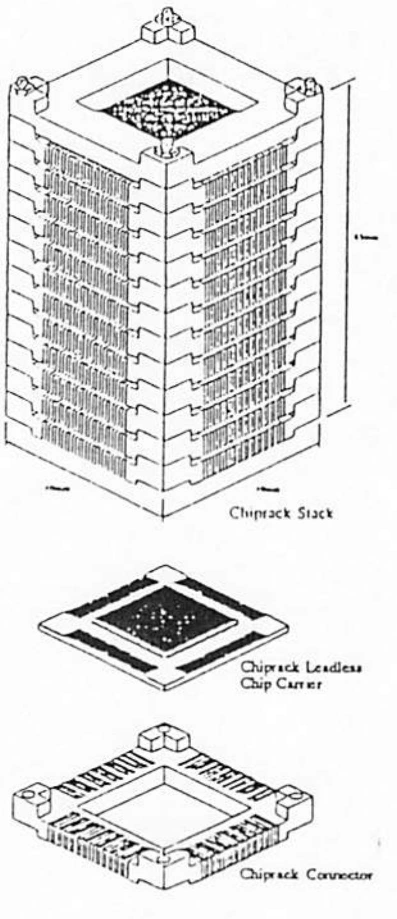


Figure 2: Chiprack stack and components.

The carriers have 208 contacts, 104 on each side. Each carrier will typically carry between one and four integrated circuits. All chips are unpackaged and bonded directly onto the carrier. Each carrier will also have a number of surface mount decoupling capacitors and other discrete components as required. The connectors provide the structure of the stack as well as connecting the contacts on the top of one carrier to those on the bottom of the one above.

There are three different carriers in our system:

- The processing carrier supports a T800 transputer.
- The memory carrier supports the external memory for one T800.
- The routing carrier supports the two T414s which make up the communications co-processor.

Due to the extra dimension introduced into the circuit by the use of a stack, the wiring of the system is much simpler. Power, clock and other system signals are distributed by making connections between corresponding pads on the top and bottom of each carrier to produce a common bus. Processor to memory connections are made very simple by placing a memory carrier directly above each processing carrier. The link connections from all four routing carriers (a total of 32 bi-directional links) are presented at the bottom of the stack for routing on the mother-board. These connections must be side stepped at each routing carrier in order to overcome the restriction that all the routing carriers present their links at the same pads. Similar side stepping is used to connect the T800s to the T414s, and also to get the spare T800 links to the top of the stack.

Progress

The first version of the routing carrier has been designed, constructed and tested with the routing software. Although found functionally correct an overheating problem was discovered. The overheating was caused by inadequate thermal conductivity between the two T414s and the metal tracks on the two layer carrier. The carrier has been redesigned onto four layers with large conductive Vcc pads under the T414s which should resolve this problem. The other two carriers are being produced in collboration with Dowty Interconnect plc. As the designs of theses carriers are minor variations on existing carriers, no overheating or other serious problem is anticipated.

Future Intentions

The T800 transputer has an overlapping link protocol such that a byte can be acknowledged before it is fully received. As a result the T800 is capable of greater effective link speed than the T414. This facility is now available on the T425 which replaces the T414. As the T425 is almost the same size as the T414 with a very similar pad layout, there should be no problem in using the T425 on a chip carrier designed for T414s, thus making the extra link speed available. The T425 also has an additional 2k bytes of internal memory, this should permit larger packet sizes and a more complex routing algorithm. It may be possible to implement *virtual cut through* rather than *store and forward* routing, giving a significant decrease in message latency.

References

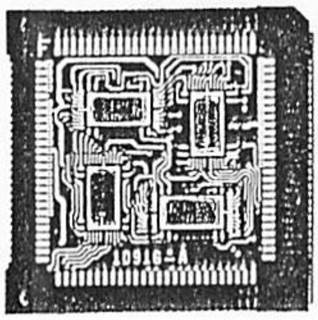
1 J.T. Yantchev, C.R. Jesshope, "Deadlock Free Packet Routing with Bounded Buffering for Asynchronous Regular Arrays of Processors", IEE Proc E, 1988.

2 Mike Anstey, "3D Interconnection - From Theory To Prototype", New Electronics, pp23-26, 9 Dec 1986.

3 P.R. Miller, C.R. Jesshope, "A Low Latency, Deadlock-Free Communications Network", 1989 Research Journal, pp75-77.

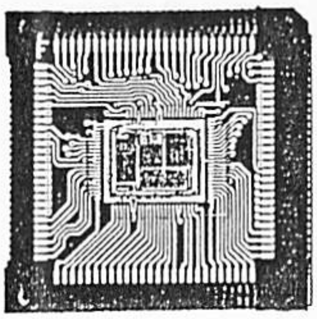
System Components

Memory Carrier



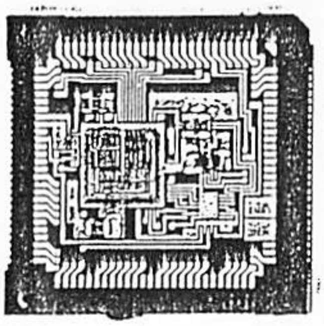
8k x 32 DRAM
(4x IMS 2630)

Processing Carrier

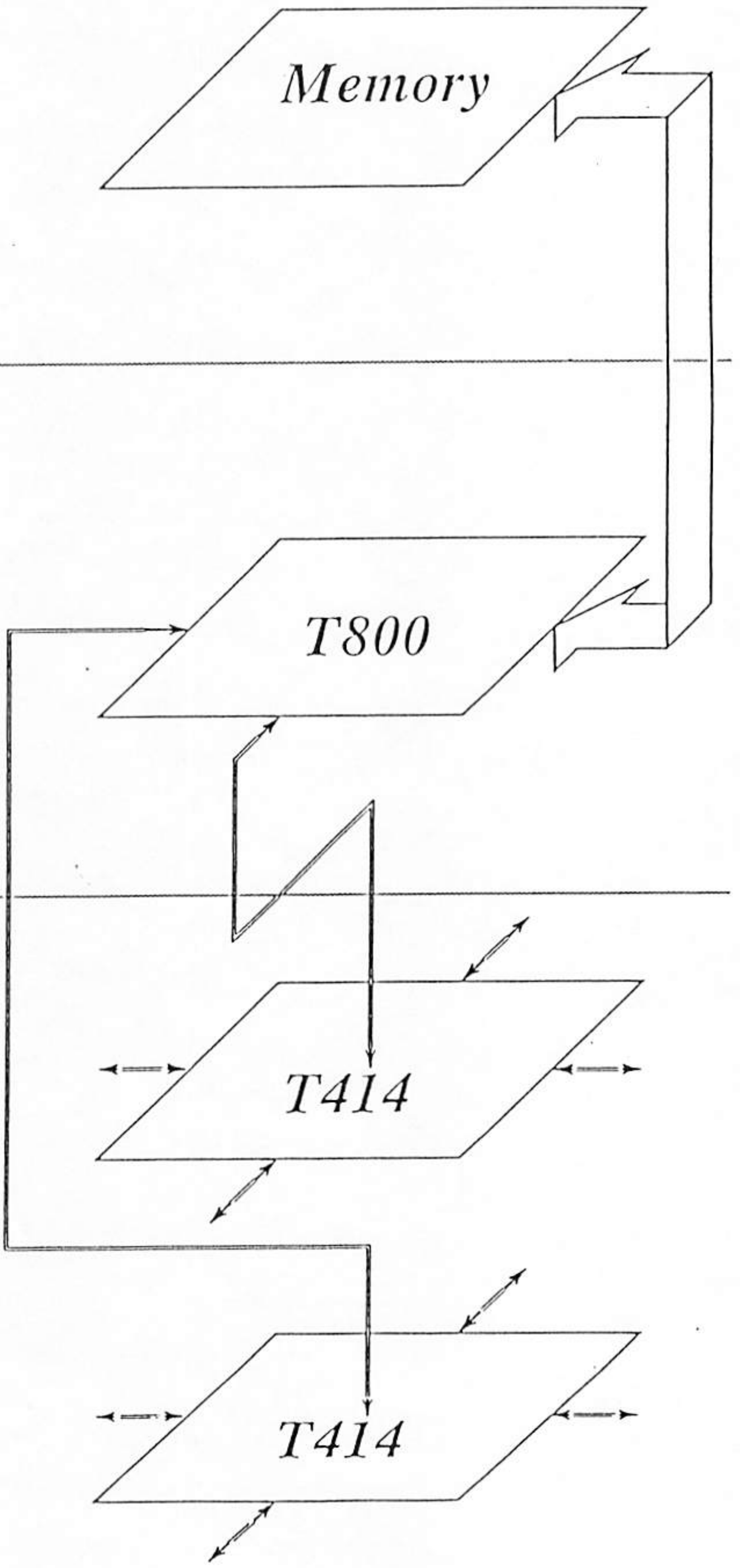


T800 32 bit transputer
with floating point unit.

Routing Carrier



2 x T414 transputers each
with 5 communication links.


Memory
T800
T414
T414