The Hardware Architecture of the

ICL GOLDRUSH MegaSERVER

Paul Watson & Ted Robinson

ICL, High Performance Technology, West Gorton, Manchester M12 5DR, UK

Abstract. This paper describes the hardware architecture of the ICL GOLDRUSH MegaSERVER, a distributed store parallel machine designed to operate as an open database server. The system is sold by ICL to commercial customers who require a high performance, high availability platform to support their critical business applications.

GOLDRUSH consists of up to 64 Elements, each of which can co-operate in database processing, exploiting both the parallelism found within complex queries and that found between queries in On-Line Transaction Processing workloads. The Elements are connected by a very high performance (1.2 GBytes/s) internal network. Each Element has local disks to store the database. Some Elements have external comms connections allowing clients to connect to the system and so send queries into it for processing. Large tape libraries can be connected to multiple Elements, so allowing databases to be archived (and restored) in parallel at high speeds.

This paper discusses the architectural options for the design of high performance computers, explaining why the Distributed Store Architecture was chosen for Goldrush. It then describes the design of the key aspects of the system including the parallel Elements, the internal network and the cabinets.

1. Introduction

The ICL Goldrush MegaSERVER [Watson & Catlow, 95] is an open, parallel database server designed for commercial users requiring high performance and availability. It has a distributed store, parallel architecture with up to 64 Elements, all of which can work together to exploit the parallelism found in database workloads. This includes both the parallelism found within complex queries and that found between queries in On-Line Transaction Processing workloads. Each Element has local disks on which the database data is held. The Elements are interconnected by a high performance Delta Network, while some Elements also have external comms connections allowing connection, via LANs and WANs, to clients which send queries into the parallel machine for processing. Large tape libraries can be connected to multiple Elements, so allowing databases to be archived (and restored) in parallel at high speeds.

The GOLDRUSH system evolved from earlier work carried out in collaborative projects with Universities and other Companies. The two most important of these projects were: the Alvey Flagship project [Watson, Woods, Watson, Banach, Greenberg & Sargeant, 1988] which produced a distributed store parallel system running both Declarative Languages and Databases; and the ESPRIT EDS project [Watson & Townsend, 1991] which developed a parallel system for Parallel Databases, Language Translation and Declarative Languages.

This paper discusses the architectural options for the design of high performance computers, explaining why the Distributed Store Architecture was chosen for Goldrush. It then describes the design of the key aspects of the system including the parallel Elements, the internal network and the cabinets.

2. Why choose a Parallel Architecture for Commercial Computing ?

The main reason for considering parallel machine architectures to support database workloads is their ability to provide very high performance. Whilst a single Element of a parallel machine may not exceed the performance of a high end Uniprocessor, the ability to harness sets of Elements to work together on the same problem makes it possible to construct systems whose performance is considerably greater, by a factor of 10 or more than that of a Uniprocessor. A parallel machine also offers the attractive property of performance scalability. Over the lifetime of a customer's use of an application, the performance requirements usually increase. With a Uniprocessor, once the limit of CPU power has been reached, it is necessary to buy a new, more powerful machine (if one exists) to provide the extra power needed. In contrast, the power of a parallel architecture can be increased by adding Elements. This removes the need to buy a new machine, so reducing cost and minimizing the disruption to users.

Another advantage of parallel architectures is that they are very cost effective. Figure 1 shows a graph of cost vs. performance for Uniprocessors. Initially the cost rises approximately linearly with the performance, but then the cost curve rises much more steeply and increases in performance become very expensive. This is the region where standard, relatively inexpensive commodity components are no longer fast enough to meet the required performance, and computer designers are forced to adopt more expensive solutions, including large static RAM caches with very low access times, interleaved stores, and leading edge, high speed semiconductor technology such as ECL.

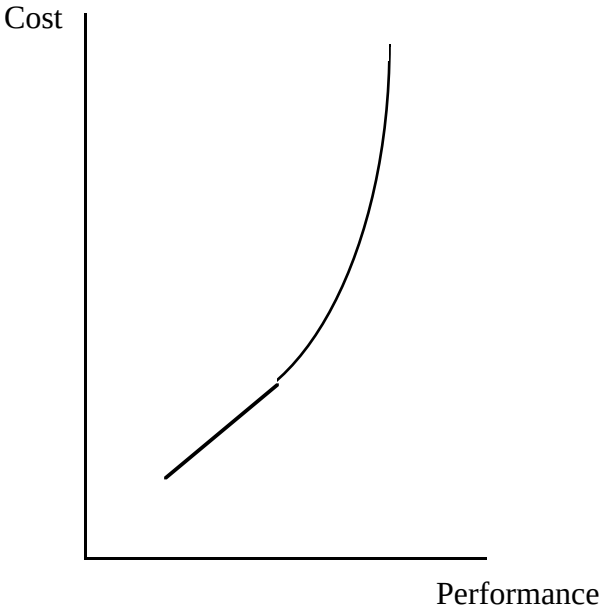

Figure 1. Cost vs. Performance for a Uniprocessor

Contrast this with the Cost vs. Performance Graph for a Parallel Machine (Fig. 2). Because performance can be increased by adding Elements, rather than by increasing the power of the Elements, the cost vs. performance graph rises in a much more linear fashion. The steps in the graph reflect the fact that due to the need to package Elements into cabinets, there will be points where adding an Element also requires the addition of a new cabinet.

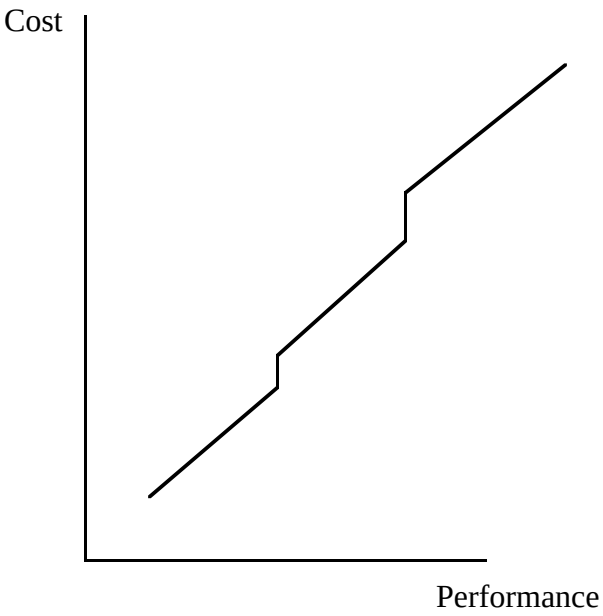

Figure 2. Cost vs. Performance for a Parallel Machine

One of the key issues in parallel machine design is choosing the right performance for the Elements. Given the cost vs. performance graph for a Uniprocessor shown in Figure 1, it is sensible to aim for a Element performance on the knee of that curve. Choosing a performance below the knee reduces Element cost, but more Elements are needed to achieve a particular performance, and this increases the cabinetry & network costs associated with those Elements. It also increases the amount of parallelism needed in the software to achieve a particular level of performance. Above the knee of the curve, increases in Element performance require relatively large increases in cost, and it is cheaper to achieve these performance gains by adding more Elements to the machine.

Another advantage of choosing an Element performance on the knee of the curve makes it possible to use commodity products in the Element design. Not only does this reduce cost but it also reduces development time and so decreases time to market when compared with the alternative of building a very high performance Element from low-volume or specially designed components. It is also an advantage that using commodity products provides a straightforward, low cost path for future performance upgrades. For example, the producers of the major microprocessor families are continually releasing higher performance but compatible versions of their products. This provides the opportunity to increase the Element performance by adopting the faster version of these components as they become available. It is important that the Element is designed to allow this, for example by ensuring that other parts of the design such as the bus and internal network connections do not become bottlenecks if the performance of the microprocessor increases.

Of course, the potential for high performance provided by the raw hardware of parallel machines is only realisable if the System and Application software can exploit it. This has been shown to be true for database servers where there is significant parallelism between transactions in On-Line Transaction Processing Workloads, and within Complex Queries [WATSON & CATLOW, 95].

3. Classes of Parallel Architecture

There are two basic types of parallel architecture to consider for commercial systems: Shared Store Multiprocessors and Distributed Store Processors (often known as MPP- Massively Parallel Processors). These are considered in this section along with the explanation for the choice of the architecture for Goldrush.

3.1 Shared Store Multiprocessors

Figure 3 shows the basic architecture of a Shared Store Multiprocessor. This consists of a set of processors communicating with store and IO (Disks and Communications Devices) over a shared bus. This architecture is attractive because all processors have access to the same store, and this simplifies the process of writing parallel programs. However, there are two potential bottlenecks which limit the scalability: the bus and the store. All processors share one store, and so it must be fast enough to service all their requests. This tends to become a problem when more than a small number of processors are sharing store. There are (expensive) solutions such as store interleaving [STONE, 93] which alleviate this problem, but this still leaves us with the second bottleneck: the bus. Each unit in a Shared Store Multiprocessor communicates with the other units over the bus. Therefore as processors are added to increase performance, the bus becomes more heavily used. However, because it has a fixed bandwidth, the share of the bus available to each processor decreases and this causes a bottleneck which limits scalability. This problem has been getting worse over time, as the processors themselves become faster, and so require greater bus bandwidth. However, it is not possible to produce similar performance increases in the buses themselves because they are basically just passive tracks of copper on a printed circuit board, with significant reactive load, whose speed is limited by the time it takes for changes in signals to propagate down the tracks. For this reason, current busses are limited to around 70MHz.

The problems of bus and store bottlenecks has limited the number of processors to typically 16 in Shared Store Multiprocessors. However due to increases in processor performance it is expected that the next generation of these machines will have more limited scalability: probably to 8 or less. This therefore reduces their performance gains over Uniprocessors.

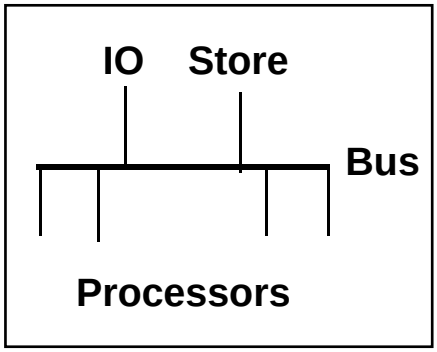

Figure 3. The Shared Store Multiprocessor Architecture

3.2 Distributed Store Multiprocessors

Figure 4 shows the Distributed Store Multiprocessor architecture. This consists of a set of Shared Store Multiprocessor Elements connected by a high performance network. The key constraint of this type of architecture is that each processor can only access its local store, and so all communication between Elements is via explicit messages sent over the internal network. There is considerable research work being undertaken in both academia and industry to develop mechanisms to provide the "illusion" that all store is equally accessible from all processors, but the main problem with this is the large difference in latency between a local store access and a remote access: the local cache hit rate must be very high if remote accesses are not to significantly reduce performance.

The key advantage that the Distributed Store architecture has over both Uniprocessors and Shared Store Multiprocessors is its scalability. By adding Elements, all the key performance attributes are scaled including: Processing, Bus, Store, IO bandwidth and connectivity. This scalability is only realizable if the Network interconnecting the Elements does not become a bottleneck. However it is possible to achieve this using today's technology (see Section 7 for a description of the Goldrush Internal Network). In fact, there are no architectural limits to the scalability of the Distributed Store hardware, however practical limits may be imposed by limitations in the scalability of the software or the cost of very large machines. For these reasons, most Distributed Store systems are limited to tens rather than hundreds of Elements. However, this still gives them a significant performance advantage over Shared Store Multiprocessor machines, and an enormous advantage over Uniprocessors.

To the system programmer writing a parallel application, the Distributed Store architecture is more difficult than the Shared Store Multiprocessor and Uniprocessor as it is necessary to sub-divide the program into units which run on different PEs and communicate via messages sent over the network. How this is achieved for the Goldrush system is described in [Watson & Catlow, 95].

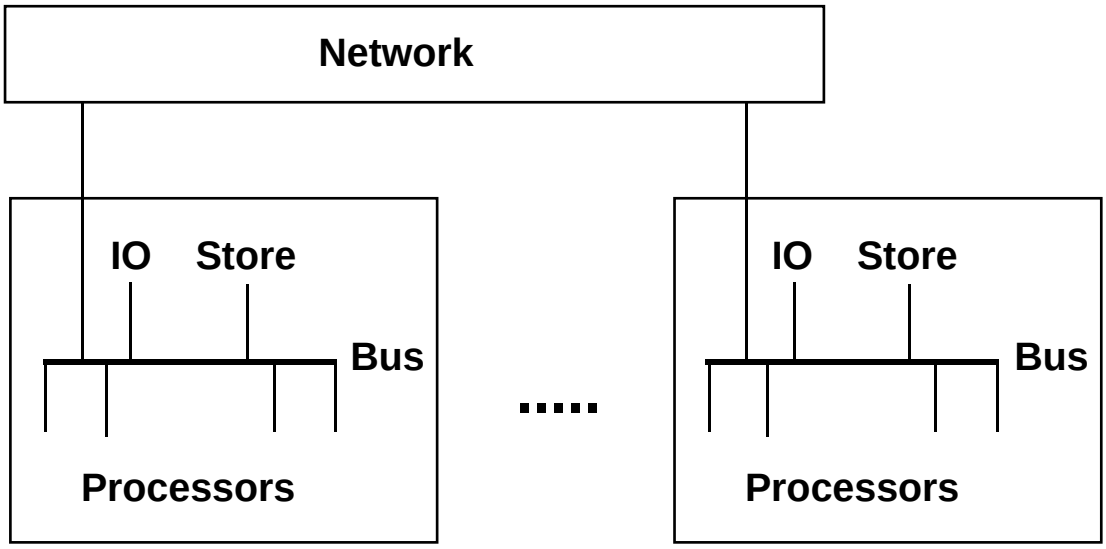

Figure 4. The Distributed Store Multiprocessor Architecture

Because of the scalability advantages of this architecture, producing a higher overall performance, this is the architecture adopted for Goldrush.

4. Hardware Architecture Overview

The hardware architecture of a GOLDRUSH system is shown in Figure 5. It consists of a set of up to 64 Processing Elements (PEs), Communications Elements (CEs) and Management Element (ME) connected together by a high performance network (DeltaNet).

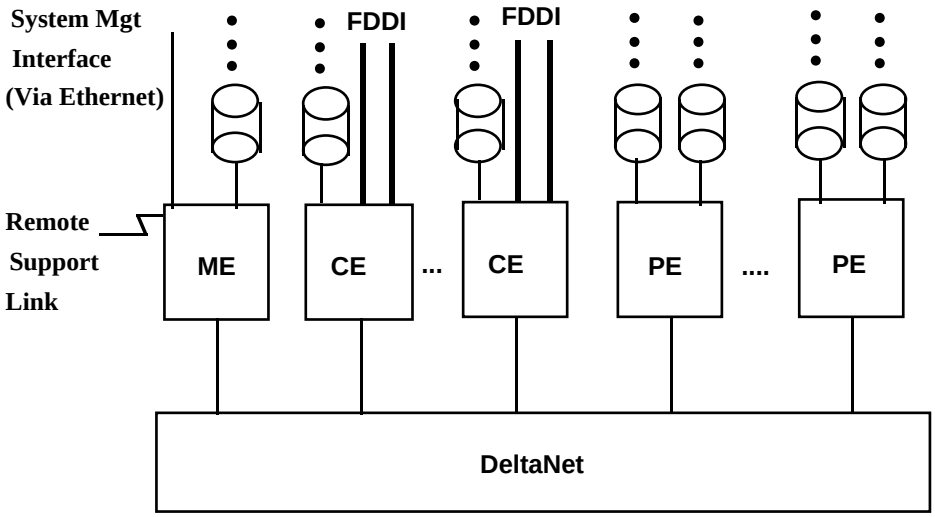

Fig 5. The Internal GOLDRUSH Architecture

The PE is the most common Element and is designed to run a Database Back-end. It has a connection to the internal DeltaNet, and up to 12 disks can be locally connected.

The Communications Element is identical to the Processing Element except that it also has two FDDI couplers for Client connection. Multiple CEs can be configured in a system to improve both performance and resilience.

The Management Element is a conventional mid-range UNIX processor (currently an ICL DRS6000) which runs the Management Software. Its responsibilities include controlling the establishment of the system, and problem diagnosis. It contains a "Teleservice" modem connection which allows problem reports to be sent to a service desk, and remote problem diagnosis.

For high performance archiving, tape libraries can be attached to the Elements via their SCSI connectors. Connections from multiple Elements to the tape library provide parallel archiving (and restoration) by allowing multiple data streams to be archived (and restored) simultaneously to multiple tape drives.

The following sections describe the design of the main components of the Goldrush Hardware: the Processing Elements (Section 5), the Communications Elements (Section 6), the DeltaNet (Section 7), the Cabinets (Section 8).

5. Processing Elements

The Processing Element (PE) is the basic building block of the Goldrush machine. Its job is to efficiently run a Unix operating system and Database Server. The main requirements it has to meet are therefore:

- High processing power
- High performance connection to the DeltaNet
- Large Main store to hold the database cache
- High disk connectivity so that large databases can be stored

It was decided to base the PE around SPARC processors because of their high performance and the availability of support chips. A standard SPARC Mbus bus is also used to connect the components within a PE. Figure 6 shows a schematic diagram of the PE hardware.

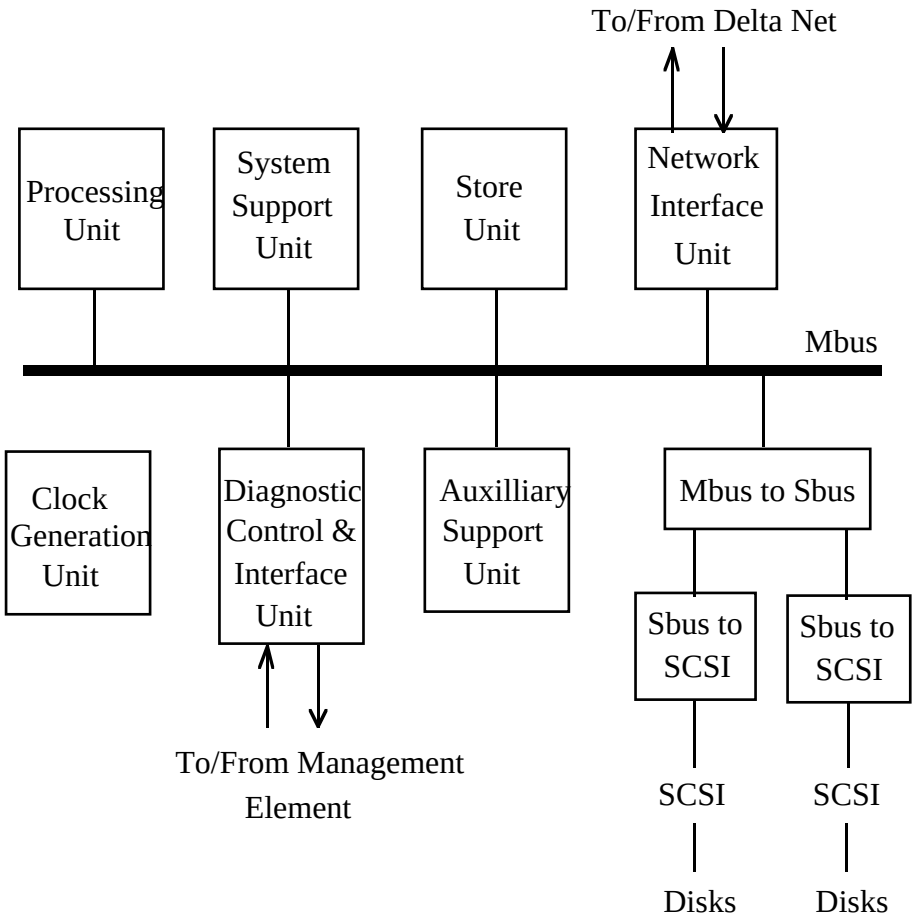

Figure 6. The Processing Element Design

The main units in the PE are:

5.1 Mbus: the standard SPARC processor bus, which operates at 40MHz and is 64 data bits wide. The Level 2 industry standard protocol is supported.

5.2 Processing Unit: a SPARC processor which runs the operating system and application software. Currently the SPARCs used for the Processing Unit (and the System Support Unit described below) are 90MHz parts but as new, faster SPARCs are produced by chip manufacturers, these can be utilised.

5.3 System Support Unit: another SPARC processor which is dedicated to the task of sending and receiving messages through the DeltaNet, so reducing the load on the Processing Unit and improving system performance. When a process on the Processing Unit wishes to send a message to another PE, it calls a system interface which passes the start address and length of the message to the System Support Unit. This is then responsible for fragmenting the message into packets which can be sent over the DeltaNet, and ensuring that they arrive without error. Similarly, when a message is received by a PE, it is the System Support Unit which de-fragments it (having waited until all the DeltaNet packets used to transmit the message have arrived), checks that there has been no transmission error, and places the message in store where it can be read by the receiving process. Only then is the Processing Unit informed of the message's arrival.

5.4 Store Unit: a large (currently 256MByte) main store built from (currently 16MBit) Dynamic RAMs. The data width is 64 bits, with a further 8 bits for Hamming error correction and detection. For high performance, the store is 2-way interleaved.

5.5 Network Interface Unit: this serves as a DMA unit between the Store and the DeltaNet. When a packet is to be sent, the System Support Unit provides the Network Interface Unit with the physical store address and length of the data. The interface to the DeltaNet operates asynchronously with the Mbus at a frequency of 20MHz. Two CRC (Cyclic Redundancy Code) bytes are appended to the message on transmission. On receipt of a message these bytes are checked. Additionally, the length of the packet is checked against that specified in the packet header. Also checked is the PE destination address in the header to ensure that the packet has not been misrouted.

5.6 Clock Generation Unit: generates 40MHz Mbus clocks to all Units.

5.7 Auxiliary Support Unit: this provides a collection of miscellaneous functions including timer, interrupt generator and bootstrap EPROM.

5.8 Diagnostic Control and Interface Unit: The Goldrush Management Element (ME) has a diagnostic, RS-232 connection to each Element, and the DeltaNet. This is used to co-ordinate the establishment of the machine by sending boot commands to each Element, but it is also used to allow the ME to access diagnostic information if an Element cannot be reached over the DeltaNet because it has failed in such a way that it can no longer send or receive messages. The Diagnostic Control and Interface Unit provides this connection from the PE to the ME.

5.9 Connection to Local Disk is provided by two fast and wide SCSI-2 connections, each of which can connect to up to 6 Disks. They have a peak transfer rate of 20MBytes per second. The SCSIs are connected to the Mbus via Mbus-Sbus and Sbus-SCSI bridges. The Disks currently used are standard 3.5" diameter devices with a capacity of 4GBytes. Because the disks are of standard size and interface, this specification will be upgraded as new, higher capacity devices become available from manufacturers.

The physical dimensions of the PE are 364mm x 233mm x 41mm.

6. Communication Elements

The Communication Element (CE) is exactly the same as the Processing Element except that one of the Sbus-SCSI bridges is replaced by two Sbus-FDDI bridges. This provides each CE with two optical FDDI couplers. FDDI was chosen as the external comms interconnect because of its high performance (100 Mbits per second) and because bridges are available from it to virtually all other types of LAN and WAN, including Ethernet.

In the Goldrush system, the CEs do not run database servers, but instead are dedicated to transparently relaying messages between the PEs and the FDDI couplers. This is done by software in such a way that each Processing Element behaves as if it had local FDDI connections.

The number of CEs per Goldrush machine is variable, but 1-2 per 16 Elements is typical. The exact number is determined by the need to provide sufficient external comms bandwidth for database queries, and also resilience to CE failure: if a CE fails then comms messages will be re-routed through another CE provided that there is a route from it to the external Client.

7. DeltaNet

The internal network is a key component of any Distributed Store Parallel Machine because if it is a bottleneck, then it will prevent the system's performance scaling as Elements are added. Also, if its latency is too high, then this can reduce the response time for database queries to unacceptable levels. The DeltaNet is therefore designed to provide both high throughput and low latency. The design was pioneered in the Flagship machine and then enhanced in the EDS project.

The DeltaNet is a multi-staged network built from the basic building block of an 8 x 8 crossbar switch (called a Network Switching Element: NSE) which dynamically provides unidirectional channels from each of its 8 inputs to any of its 8 output ports. These NSEs are connected in stages: two stages allowing up to 64 connections. Figure 7 shows how a 32 Element network can be created from these switches. Note that the unconnected NSE inputs and outputs would be used in a 64 Element machine. Alternatively, they can be used as alternative paths in systems of up to 32 elements.

Each packet transmitted through the network consists of 128 Bytes of data, 16 Bytes of header information and a 2 Byte Cyclic Redundancy Check. The header includes the destination Element, the source Element, the length, and a sequence number which is used if a message of greater than 128 Bytes has to be fragmented (by the System Support Unit) and sent in a set of 128 Byte packets. The 2 Byte Cyclic Redundancy Check is generated by the sending Element, and checked by each NSE the packet passes through- so ensuring the integrity of the transmission. Connections from the Elements to the NSEs are 11 bits wide: 8 data plus 3 control.

When a packet is sent by an Element it reaches the first NSE where the Destination address is inspected and the packet is immediately routed out along the appropriate path towards its destination. When it reaches the second stage of the network, this process is repeated and this causes the packet to be routed to the destination Element. Therefore, if there is no contention, a packet will pass through both Network stages without any buffering, and this reduces the latency of the network. However, if two or more packets arrive at the inputs of an NSE simultaneously, and each needs to be transmitted through the same output, then this results in contention as only one packet can be transmitted through an output at any one time. The Network deals with this as follows. Each NSE has 4 buffers on each input. If a packet arrives at an input, and its output is blocked then it is held temporarily in one of these buffers until the output is free. In the worst case, if all four buffers at an input are full, then no more packets are accepted on that input until a buffer becomes free.

The Delta Network runs asynchronously with respect to the Elements, currently at 20MHz. In combination with the System Support Unit, it allows each Element to send and receive data at up to 20MBytes/s simultaneously. For a 64 Element system, this provides a total bandwidth of 1.2 GBytes/s.

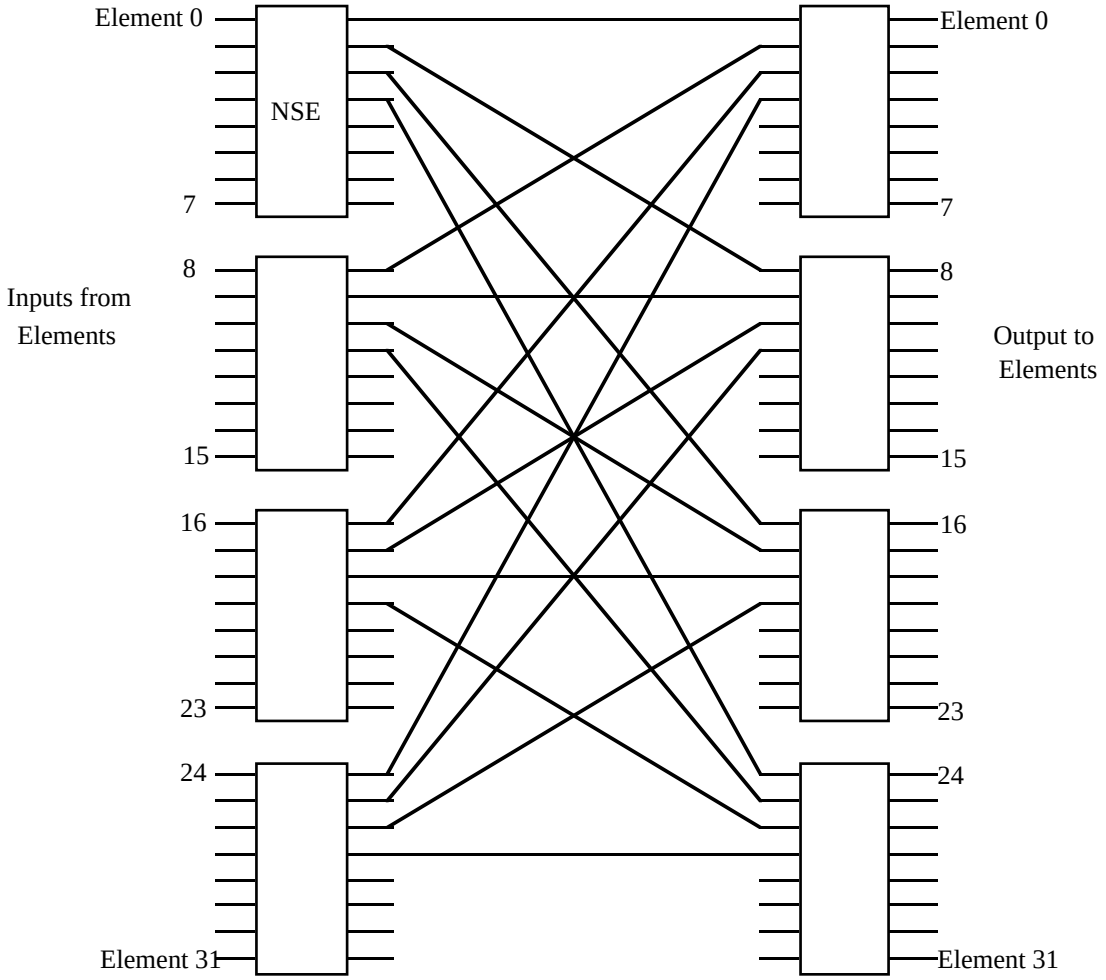

Figure 7. A 32 Element DeltaNet

8. Cabinets

The design of the cabinets for a parallel machine is important because of the need to support a wide variety of configurations - Goldrush machines can have from 4 to 64 Elements - and the importance of limiting the space occupied by the larger configurations so they do not require a very large machine room.

There are two types of cabinet used in Goldrush: the Processor Cabinet and the Disk Cabinet. These are described below.

8.1 Processor Cabinet

This contains the DeltaNet, up to 16 Elements and up to 64 Disk Drives. Assuming that 4 GByte disks are used, this gives a total of 256 GBytes per Processor Cabinet. All disks are housed so as to allow them to be replaced without powering down the system using hot pull and push. The cabinet is shown in Figure 8. There are between 1 and 4 of these Cabinets per system, with dimensions of 1450mm high x 705mm wide x 860mm deep, giving a high density of processing power and disk storage. In order to provide high availability, the cabinet has five fans, but can still operate even if 2 fail; Similarly, there are two power supplies, but the system can continue if one fails.

[Photograph]

Figure 8. The Goldrush Processor Cabinet

8.2 Disk Cabinet

Each Processor Cabinet may be connected to one Disk Cabinet, which may contain up to 120 disks, so allowing the filestore of the Processors to be considerably increased by a total of 480 GBytes. This gives a total of up to 50 GBytes per Processing Element.

9 Conclusions

Goldrush is one of the first commercial embodiments of a type of computer architecture expected to be pervasive by the end of the century. It has demonstrated how a parallel machine can offer levels of performance not possible with any other type of architecture, but with low cost-performance. This is achieved by exploiting the scalability of the Distributed Store Architecture and by utilizing high performance, but commodity, components where possible.

Acknowledgements

The GOLDRUSH MegaSERVER is the result of the work of a large number of people at ICL, High Performance Technology, West Gorton, Manchester. We would particularly like to recognise the contribution of all past and present members of the Goldrush Hardware Team. A debt is also owed to our partners in the Alvey Flagship and ESPRIT EDS projects.

References

ALAMASI, G.S. and GOTLIEB, A.J., *Highly Parallel Computing*, Benjamin/Cummings, ISBN 0-80530-443-6, 1989.

STONE, H.S., *Highly-Parallel Computer Architecture*, Addison Wesley, ISBN 0-20152-688-3, 1993.

WATSON, P. and CATLOW, G.W., *The Architecture of the ICL Goldrush MegaSERVER*, In this Issue.

WATSON, I., WOODS, J.V., WATSON, P., BANACH, R., GREENBERG, M. & SARGEANT, J., *Flagship: A Parallel Architecture for Declarative Programming,*. in Proceedings of the 15th Annual International Symposium on Computer Architecture, Honolulu, Hawaii, 1988.

WATSON, P., and TOWNSEND, P., *The EDS Parallel Relational Database System*, in Parallel Database Systems, ed. P. America, Lecture Notes in Computer Science 503, Springer-Verlag, 1991.

Biographies

Paul Watson

Paul Watson received a BSc in Computer Engineering from the University of Manchester in 1983, and gained a PhD from the same institution in 1986. From 1986 to 1989 he was a Lecturer in Computer Science at the University of Manchester but in 1990 he joined ICL, leaving in 1995 to take up a post in the Computing Science Department at the University of Newcastle.

During his time in Academia and Industry he has worked on the design of 3 generations of distributed store parallel systems. The first, funded by the Alvey Flagship project, resulted in the production of a prototype machine executing declarative languages. The second, the ESPRIT funded EDS machine, ran parallel declarative, language translation and database systems. The third is the ICL Goldrush machine described in this paper.

Dr. Watson is a Member of the British Computer Society and a Chartered Engineer.

Ted Robinson

Ted Robinson obtained a BEng in Electrical Engineering from Liverpool University in 1958. After a period with AEI he joined ICL (then ICT) in 1967 as the designer of a large fixed disk controller for 1904E machines. Subsequently he has been involved in a variety of hardware projects- System 4 emulator in 2970; 2966 OCP design; clock and store design for distributed system 39 (DM1).

For the past 9 years he has had numerous responsibilities in parallel architecture collaborations- Alice, Flagship and EDS, the latter evolving into the Goldrush MegaSERVER. Ted is currently investigating options for Goldrush enhancements and replacements.