Interconnection Network and Distributed Shared Memory of a Massively Parallel Machine JUMP-1

Hideharu Amano† (hunga@aa.cs.keio.ac.jp)
Katsunobu Nishimura† (nisimura@aa.cs.keio.ac.jp)
Tomohiro Kudoh‡ (kudoh@cc.teu.ac.jp)
Hiroaki Nishi† (west@aa.cs.keio.ac.jp)
Ken'ichiro Anjo† (anjo@aa.cs.keio.ac.jp)

†Dept. of Computer Science, Keio University
3-14-1, Hiyoshi Yokohama 223 JAPAN
Phone: +81-45-560-1063 Fax: +81-45-560-1064
‡Dept. of Information Technology, Tokyo Engineering University
1404-1 Katakura Hachioji Tokyo 192 JAPAN
Phone: +81-426-37-2111 (ext.2215) Fax: +81-426-37-2118

Abstract

For cache coherent distributed shared memory on a large scale parallel machine, each node processor of JUMP-1 shares a global virtual address space with two-stage TLB implementation. The directory is attached not to every cache line but to every page, while the data is transferred by the unit of a cache line. Reduced Hierarchical Bit-map Directory schemes (RHBDs) are introduced to manage coherent messages to a large number of nodes which share a page. Using a hierarchical structure of a network RDT(Recursive Diagonal Torus), coherent messages for RHBD are multicast quickly to local area of the source node.

From the results of the simulation, all schemes of the RHBD support almost a half latency of the traditional directory scheme based on 1 to 1 message transfer.

1 Introduction

Cache Coherent Non-Uniform Memory Access machine (CC-NUMA) is one of hopeful candidates for massively parallel computers. It provides distributed main memory, scalable interconnection network, and directory-based coherent cache. JUMP-1 is a massively parallel processor prototype with cache coherent distributed shared memory developed by collaboration between 7 Japanese universities[6]. The major goal of this project is to establish techniques for building an efficient distributed shared memory on a massively parallel processor. As shown in Figure 1[3][6], JUMP-1 consists of clusters connected with an interconnection network Recursive Diagonal Torus (RDT)[16] which provides both torus and hierarchical structure. A cluster also provides a high speed point to point I/O network connected with disks and high-definition video devices.

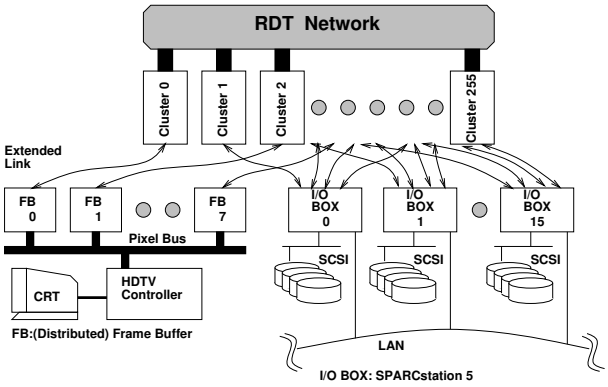

Figure 1: Structure of JUMP-1

As shown in Figure 2, each cluster is a bus-connected multiprocessor including 4 coarse-grained processors(CPU), a fine-grained processor (Memory Based Processor or MBP) which is directly connected to a main memory and the RDT router chip. A CPU is an off the shelf RISC processor (SUN Super-Sparc+) which performs the main calculation of the program.

In most existing CC-NUMA (Stanford DASH[2]/FLASH[7], MIT Alewife[8] or STiNG[10]), a directory entry is associated with every cache line, and the directory without hierarchical structure is utilized. However, this approach requires a large amount of memory for a massively parallel processor which provides both a large number of processors and a large amount of memory space. However in JUMP-1, node processors share a global virtual address space with two-stage TLB implementation, and the directory is attached not to every cache line but to every page, while the data transfer is performed by

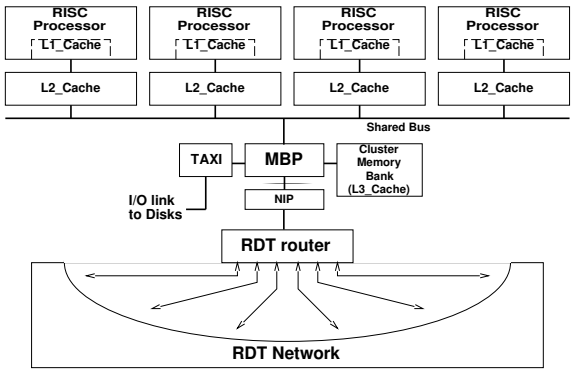

Figure 2: Structure of a JUMP-1 cluster

a cache line. Unlike other CC-NUMAs, update type cache coherence protocols can be utilized in JUMP-1 for applications which require frequent data exchange.

For supporting this approach, Reduced Hierarchical Bit-map Directory schemes (RHBDs) were introduced[14][9]. In the RHBD, the bit map directory is reduced and carried in the packet header for quick multicasting without accessing directory in each hierarchy. The hierarchical structure of the RDT(Recursive Diagonal Torus) is suitable for efficient implementation of the RHBD. In this paper, the interconnection network and distributed shared memory structure are introduced with initial evaluation results, and the current status of the project is reported.

2 Reduced hierarchical bit-map directory scheme

2.1 Hierarchical bit-map directory scheme

In JUMP-1, directory entries are associated with pages while the data are transferred by a cache line[15] in order to reduce the required amount of memory for the directory. Using this strategy, both the expansion of the directory memory and the congestion of the network caused by the large size message transfers can be avoided.

However, in this case, number of destinations of the coherence maintenance messages increases especially when an update type protocol is used. If there are considerable number of destinations, it will take a long time to send a message one after another (sequentially). The hierarchical bit-map directory scheme (HBD) used in COMA[4] (Cache Only Memory Architecture) permits a message to be transferred to different destinations simultaneously (i.e. multicast) through a tree structured multicasting paths (multicasting tree).

Figure 3 illustrates the concept of the hierarchical bit-map directory scheme. Leaves of the tree correspond to processors. A packet is first sent to the root of the tree, and then multicast through the tree structure. Each node multicasts a packet

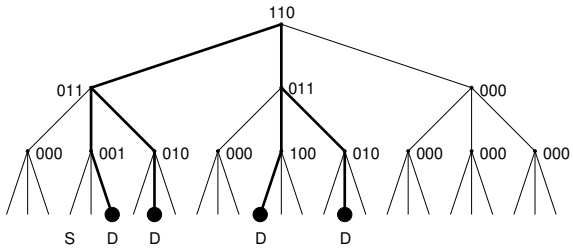

Figure 3: Hierarchical bit-map directory scheme

to its multiple branches. To specify the destination leaves precisely, an n-bit bit-map is required at each node when an n-ary tree is used. Thus, for m height n-ary tree, total of $\sum_{k=1}^{m} n^k$ bits are required for each entry. Although the required amount of memory is slightly larger than that of the full-map directory scheme in which n^m bits are required, a packet can be multicast simultaneously.

For cache consistency, acknowledge packets are usually required. These packets are transferred from the destination nodes (leaves) to the source node (root), and informs the finish of invalidation (or data update). In the hierarchical directory method, acknowledge packets can be collected in each hierarchy. It reduces the network traffic especially when the number of destination nodes is large.

2.2 The RHBD

Since the HBD scheme requires look-up to the directory at each level of the interconnection, memory access latency with the HBD becomes inhibitive on a massively parallel processor. In order to cope with this problem, reduced hierarchical bit-map directory scheme (RHBD) was proposed[14][9]. In this scheme, the bit-map is reduced using the following techniques:

- One bit-map is provided for a level of the hierarchy. The same bit-map is used at nodes of the level.
- A packet is sent to all children of a node (thus, broadcast) when some condition is met.

Note that the reduced directory is not maintained at each hierarchy, but stored only at the root node and sent in the header of a packet. Thus, no directory access is needed at the intermediate nodes, and packets can be multicast quickly.

By the combination of the above two techniques, three schemes, the LPRA, SM, and LARP are delivered:

LPRA (Local Precise Remote Approximate) scheme: A packet is first goes from a

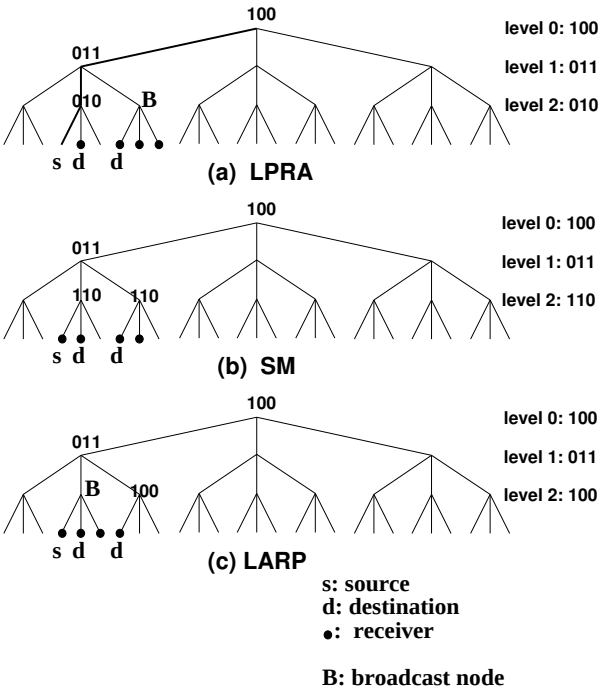

Figure 4: Hierarchical bit-map directory schemes

leaf (i.e. source processor) to the root node (upstream), and then multicast from the root node. When multicasting, (i.e. packet goes downstream), the bit-map is used only at the nodes on the upstream path. For other nodes, the packet is broadcast to all their children. Figure 4(a) shows an example of this scheme for the 3-ary tree. In this figure, 's', 'd' and • denote the source leaf, the destination to which packet is sent and leaves which receive data respectively. Using this scheme, the bit-map is used for local nodes of the source node, while the message is broadcast in the remote subtree marked **B**. This scheme is advantageous when a node sends a message to a remote node group.

SM (Single Map) scheme: In the SM scheme, all nodes at a level use a unique bit-map as shown in Figure 4(b), and thus, no broadcast is made (unless a bit-map is all-1). This scheme is advantageous when the number of destination is not so large.

LARP (Local Approximate Remote Precise) scheme: The LARP is a complimentary scheme of the LPRA. In this scheme, a node on the upstream path broadcasts if its parent sends the packet to multiple nodes (i.e. a sibling also receives the packet). Once a node broadcasts, all its descendants on the upstream path also broadcast.

In the example shown in Figure 4(c), since the multicast starts at the level 1 (the top node (level 0) only sends a message to a child), the broadcast (marked **B**) starts at the level 2 to the subtree which includes the source node. In this case, the broadcast is done for local nodes of the source node, while the bit-map is used for remote nodes. This scheme is advantageous when the mapping can make the best use of the locality of communication.

2.3 The RHBD on the RDT

2.3.1 Interconnection network RDT

The RDT is a network consists of recursively formed two-dimensional square diagonal tori. Bypass links for the diagonal direction is used for reducing the diameter of the torus network. Assume that four links are added between a node (x, y) and nodes $(x \pm n, y \pm n)$. Here, n is called the *cardinal number*. Then, the additional links form a new torus network (rank-1 torus) at an angle of 45 degrees to the original torus, and the grid size is $\sqrt{2}$n times of the original torus. Similarly, on the rank-i torus, we can make another torus network (rank-$i+1$ torus) by providing four links in the same manner. The maximum rank level is $\lfloor \log_n \sqrt{N} \rfloor$, where N denotes the number of processor. Note that, each node can select different rank of upper tori from others.

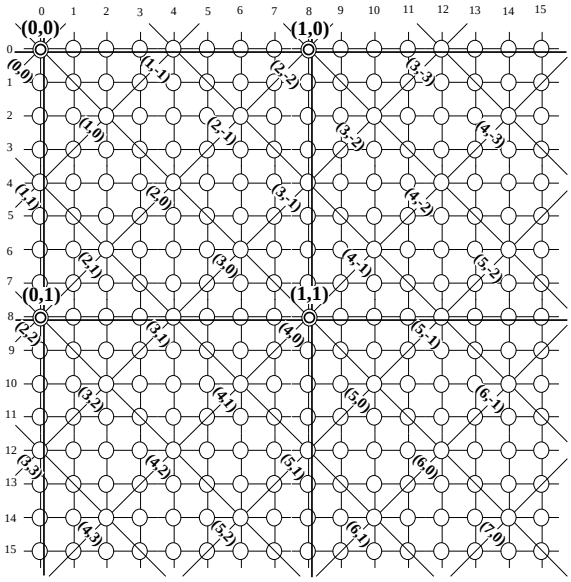

Figure 5: Upper rank toruses

The RDT in which every node has links to form all possible upper tori is called the perfect RDT (PRDT(n,R)) where n is the cardinal number (usually, 2) and R is the maximum rank. Although PRDT is unrealistic because of its large degree (4(R+1)), it is important as a basis for establishing routing algorithm, broadcasting/multicasting, and other message transfer algorithms The recursive diagonal torus RDT(n,R,m) is generally defined as a class of networks in which each node has links to form base (rank-0) torus and m upper tori (the maximum rank is R) with the cardinal

number n. Note that, each node can select different rank of upper tori from others.

The JUMP-1 must be scalable to the system with ten thousand nodes. In this case, m is set to be 1 (degree = 8). For this number of nodes, the maximum rank of upper tori is 4. Thus, the RDT(2,4,1) is treated here.

In the RDT, each node can select different rank tori from others. Thus, the structure of the RDT(2,4,1) also varies with the rank of tori which are assigned to each node. This assignment is called the *torus assignment*. Various torus assignment strategies can be selected considering the traffic of the network. If the local traffic is large, the number of nodes which have low should be increased. However, complicated torus assignment introduces difficulty to the message routing algorithm and implementation. For the JUMP-1, we selected a relatively simple torus assignment shown in Figure 6.

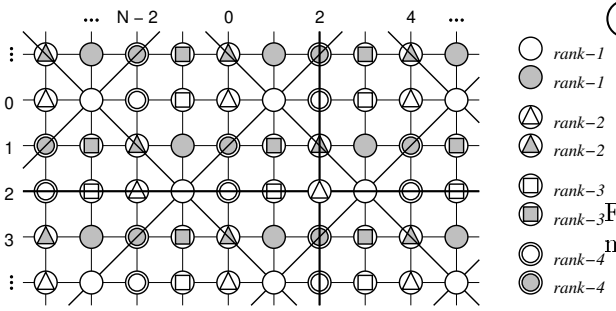

Figure 6: Torus assignment used in the JUMP-1

In this assignment, a node has eight links, four for the base (rank-0) torus and four for rank (1-4) torus (Most of links for upper rank tori are omitted in Figure 6). Note that a node with a rank torus has neighboring nodes with tori of other three ranks. Therefore, any rank torus can be used with a single message transfer between neighboring nodes. This property reduces the diameter and average distance between nodes.

The RDT provides various advantages as an interconnection network of the JUMP-1. It includes two dimensional mesh, and simple near-optimal routing algorithm enables smaller diameter than that of the hypercube (11 for 2^{16} nodes) with smaller degree (8 links per node). Moreover, it includes the fat-tree of mesh structure. Using this structure, hierarchical multicast can be efficiently supported. This feature is important to implement the hierarchical bit-map directory scheme treated here.

2.3.2 Implementation of the RHBD

In the RHBD, messages are transferred on a tree structured communication paths. However, if a simple n-ary tree is used, following problems will bottleneck the performance.

- If multiple nodes multicast simultaneously, congestion may be caused around the root node of the tree.
- Depending on the location in the tree, some messages should have transferred through the very root of the tree just to move to neighboring node.
- It takes a large latency if a sender node at a leaf sends a message upstream to the root of tree.

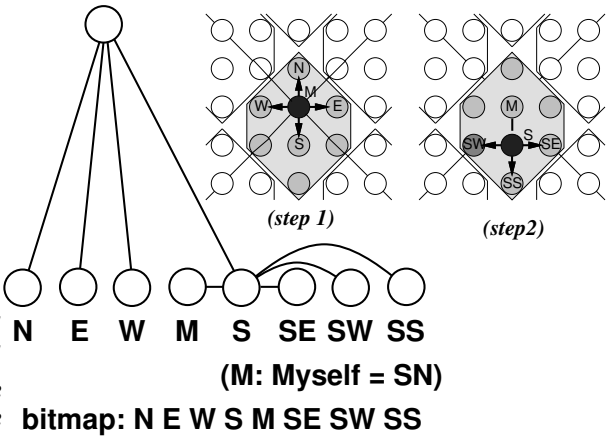

Figure 7: The 8-ary tree and bit-map pattern for multicast on the RDT

However, since the RDT involves a fat tree of tori with multiple root nodes, these problems can be avoided. The pattern of message transfers for emulating a fat tree is shown in Figure 7. Two steps are required: (1) each node transfers a message to four neighbors, (2) a neighbor transfers the message to three neighbors. (South direction in this figure). As shown in Figure 7(b), the eight nodes which received the data are not duplicated with node which received other nodes on the same rank-i torus. Thus, if all nodes with rank-i tori execute this pattern, the message is transferred to all nodes with rank-(i-1) tori. By the iteration of this data transfer from the maximum rank to the rank-0, 8-ary tree in which is formed on the RDT. In this case, a rank in the RDT directly corresponds to the level of the tree. Moreover, in the RDT, the upper rank torus can be used within a step of message routing. Thus, the message can be directly transferred from the sender node to the root node without using the tree structure.

Figure 7 shows the 8-ary tree involved in the RDT, and bit-map pattern for the hierarchical bit-map directory scheme. When schemes described in the previous section are applied on this 8-ary tree on the RDT, followings are advantageous:

- On the RDT, there are a number of nodes with the maximum rank torus, and all of them are used as a root node. Thus, the message multicast is almost directly started from the root node without going upstream on the tree.

- Since there are many upper rank tori in the RDT, the tree is a kind of "fat-tree" which provides many root nodes in each rank. Therefore, the congestion of root nodes is relaxed even if many source nodes multicast their data simultaneously and independently.
- In the RDT, nodes which receive the message through the tree whose root rank is 'i' are located around the source node. For larger 'i', the number of such nodes becomes large, thus the area which a message is multicast becomes wide. We call such an area "territory" of a multicast. Figure 8 shows territories of a multicast from rank-0 and rank-1. Since the territory of is always formed surrounding a source node, message multicast to local nodes are performed from a lower rank (thus, with only a small territory).

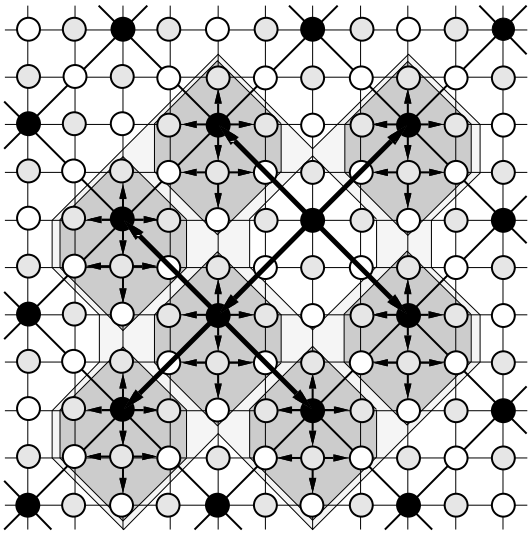

Figure 8: Territory of a multicast

3 Latency analysis considering congestion

Here, the performance of the distributed shared memory on JUMP-1 is compared with conventional directory scheme based on 1-to-1 message transfer.

3.1 Simulation with probabilistic model

First, the results of simulation with probabilistic model is shown. In this model, the traffic is generated with probabilistic model. Figure 9 shows multicast latency versus message transfer interval of the RHBD(LPRA, SM and LARP) and traditional directory methods (1 to 1). Here, multicast latency means the time required for transferring all multicast packets to destination nodes. All packet length is fixed as 8 flits, and the size of the system is 256 nodes. From the analysis of some parallel programs from SPLASH benchmark suits, the average number of destinations is 4-6 when the distributed memory is managed by 4K byte page[9]. Here, the number of destinations is set to be 6.

In the RHBD, if destination nodes can be mapped inside the territory, the number of unnecessary messages can be drastically reduced. However, it depends not only on the mapping strategy but also on the characteristics of application programs. Here, the destinations are determined by a random numbers which follow a given distribution. In the following graphs, average value calculated from 10,000 trials based on the random number generation library of Sun OS4.1.3 are shown. Here, the standard deviation (SD) is set to be 5. In this situation, since the locality of the communication is not so strong, a lot of packets are sent beyond the territory.

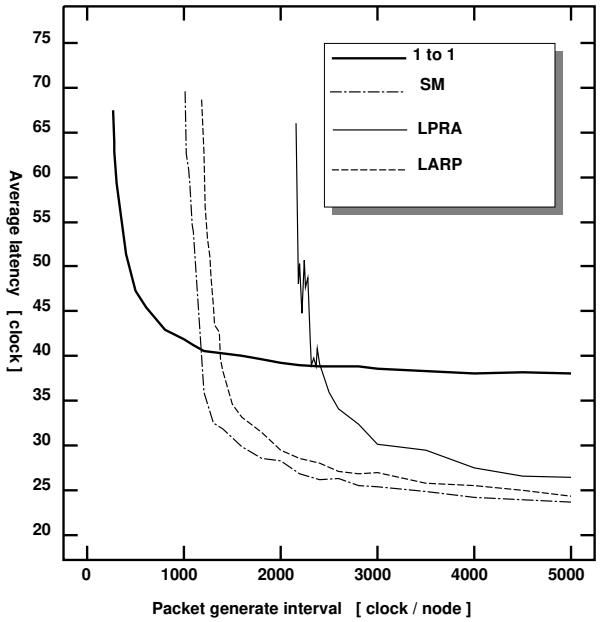

Figure 9: Multicast latency vs. message generation interval

The multicast latency is almost a half of traditional methods when the messages for cache coherence are not frequent (Thus, the interval is long.). With traditional directory methods, packets must be transferred one after another unlike they are simultaneously multicast by the router chip in the RHBDs. Because of the sequential packet transfer, links between the router chip and the MBP bottleneck the network with traditional methods.

When message generation interval becomes short, the multicast latency is rapidly increased because of the network congestion. The inverse of the interval which causes rapidly increasing of the latency gives available bandwidth of the network.

The available bandwidth of the RHBD is worse than that of traditional 1 to 1 message transfer because of the congestion caused by unnecessary packets.

3.2 Trace driven simulation

Figure 10 shows the results of trace driven simulation. The trace data is generated by executing Choleskey (solving a simultaneous linear equation with a large sparse coefficient matrix) from SPLASH benchmark[12] on the execution driven simulator MILL[13]. By limitation of the simulation time, the number of processors is only 64 (8×8). The simplest modular mapping is adopted.

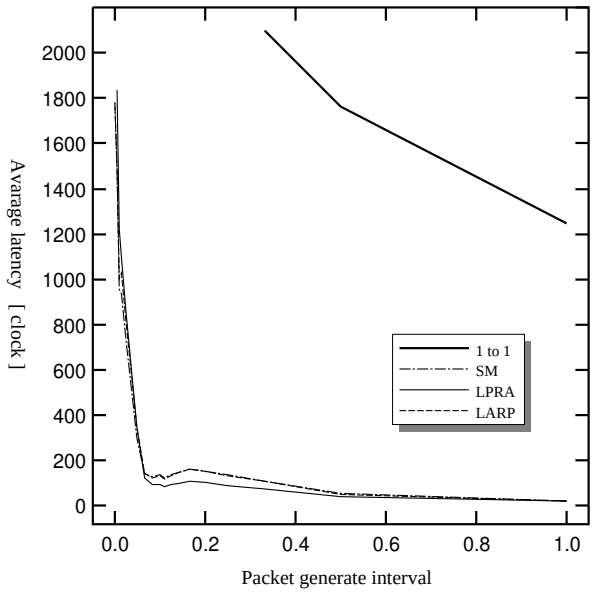

Figure 10: Trace driven simulation

As shown in Figure 10, the latency of three schemes of the RHBD is almost same and much better than that of traditional 1 to 1 message transfer. This comes from the fact that a large part of communication in Choleskey becomes the broadcast since the number of processors are not so large. In this case, the bit-map used in three RHBD methods is almost the same. The frequent broadcast causes severe bottleneck on the links between the MBP and router with 1 to 1 message transfer.

This result demonstrates that the performance of the distributed shared memory management mechanism on JUMP-1 under the practical situation.

4 Conclusion and Current Status of the Project

Here, the distributed memory management scheme on the interconnection network RDT is described. Simulation results demonstrate the efficiency of this method under the practical situation.

Now, the JUMP-1 project is in the following stage:

Router chip: The RDT router chip (Figure 11) which supports all RHBD schemes shown here is now operational.

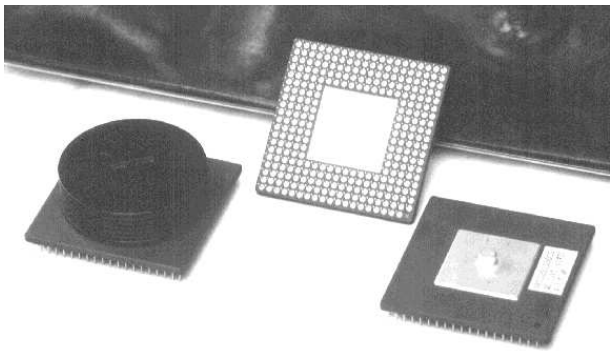

Figure 11: The RDT router chip

The core of the chip is a 10×11 crossbar which exchanges packets from/to 10 18-bits-width links as shown in Figure 12, that is, four for the rank-0 torus, four for the upper rank torus, and two for the MBPs which manage the distributed shared memory of JUMP-1.

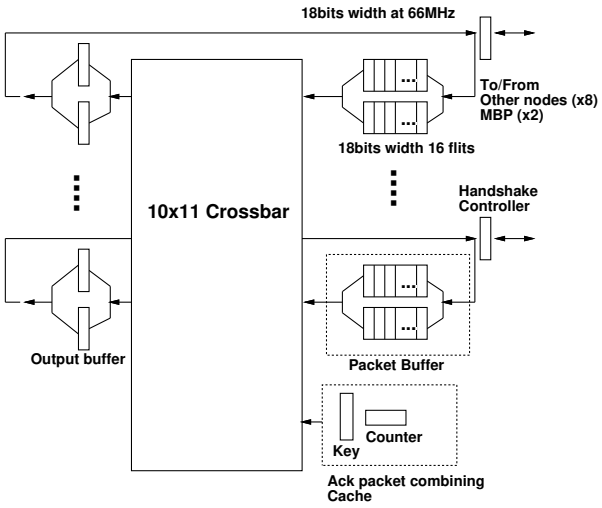

Figure 12: The structure of the RDT router

In JUMP-1, two RDT router chips are used in the bit-sliced mode to form 36 bits width for each link. All packets are transferred between router chips synchronized with a unique 60MHz clock. In order to maximize the utilization of a link, packets are bi-directionally transferred. Maximum packets

length is 16flits (36 bits-width 16flits-length) so as to carry a line of the cache. 3 flits header which carries the hierarchical bit-map are attached to all packets, but the length of the body is variable. If there is no conflict, the first flit of the packet can pass through this router with 5 clocks.

The virtual cut-through (asynchronous wormhole) routing is adopted to cope with frequent multicasting. Although a packet can be forwarded to the buffer in the next node while the packet is being received like a usual wormhole routing, a buffer which can hold the maximum size packet is provided for each virtual channel. When the target buffer is occupied, the whole packet can be stored in the buffer inside the chip.

The simplest way of multicast is to wait for all required opposite buffers and then multicast simultaneously. However in this way, the opportunities of multicast are badly reduced when the number of the multicast destinations is increased. In the RDT router chip, each buffer provides the bit-map indicating required destination of a packet. The packet is sent whenever the empty opposite buffers are found, and then the corresponding bits of the bit-map are reset. Just after the multicast which clears all bits of the bit-map is started, the buffer is allowed to receive the next packet.

The specification of the RDT router chip is shown in Table 1.

Table 1: The number of gates and data of the RDT router

Block name	Gates	Number of Blocks	Total gates
Crossbar	2,927	1	2,927
Arbiter	2,736	1	2,736
Multicast controller	1,558	10	15,580
Bit map generator	2,288	10	22,880
Acknowledge combining	2,009	1	2,009
RAM for buffer	2,021	20	40,420
Total			90,522

Total pins	299(Signal 260)
Power consumption	19.4W
Optimized number of gates	80,307
Rate of gate utilization	63%

MBP: The MBP consisting of Memory Module Controller, MBP core processor, and RDT Packet manager is now under developing by University of Tokyo and Keio University. It will be implemented in the CMOS SOG with 200,000 gates in the next March.

L2 Cache and bus chip: JUMP-1 L2 cache which is under development in Kyoto University is a sophisticated snoop cache which provides versatile facilities for the memory based FIFO[5], mixed protocol, cache injection, and cached I-structure. In order to avoid the electromagnetic side effects caused by the shared bus, the JUMP-1 cluster uses "bus-chip". This chip is a high speed switch which behaves as if it were a passive shared bus with the arbitration and snoop facility. L2 cache and bus chip will be available during this year.

I/O systems: A point-to-point I/O called the STAFF link[11] is now available for JUMP-1. STAFF link mechanism consists of fast serial communication LSI(AMD Taxi) chip sets, Send and Receiving FIFO, and controller of them. Parallel disk system based of the STAFF link is now being developed on the SUN workstation.

4.1 Prototype systems of JUMP-1

In JUMP-1, the page table for the distributed shared memory is managed by the operating system, and the coherent messages for the cache is managed by the MBP core. Thus, the development of such programs is essential for JUMP-1 project.

For environment developing such a software, two prototype systems are used:

JUMP-1/3: JUMP-1/3 is a workstation cluster system with a distributed shared memory. A shared memory board which provides a microprocessor for protocol management is attached to the SBUS of the SS5. The shared memory board is connected with boards attached to other SS5 workstations through the RDT boards each of which provides two RDT router chips. This system with four nodes (Figure 13 is available now, and the software for providing distributed shared memory is now under development[1].

JUMP-1/2: JUMP-1/2 is a prototype multiprocessor which provides a commercial microprocessor (TMS320C40) instead of the MBP. This system is used not only for software development but also the test of the hardware core of the JUMP-1: L2 cache and bus chip.

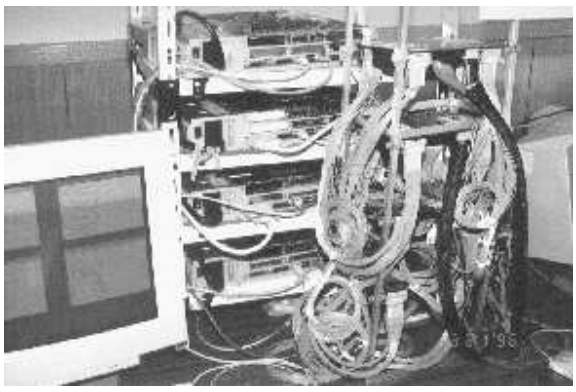

Figure 13: The JUMP-1/3 workstation cluster

Based on the experiences of these prototypes, the JUMP-1 will be scheduled to start to be built up from the next March.

Acknowledgment

The authors would like to express their sincere gratitude to the members of the Joint-University project for their valuable advice and discussion, specially, Prof. K.Hiraki, Dr. M.Sato, Prof. T.Matsumoto of University of Tokyo, Prof. H.Nakajo of Kobe University, Prof. S. Tomita, Prof. H.Nakashima and Prof. M.Goshima of Kyoto University for generous discussion. The authors also express their thanks to Mentor Graphics Japan for supporting design tools.

A part of this research was supported by the Grant-in-Aid for Scientific Research on Priority Areas, #04235130, from the Ministry of Education, Science and Culture.

References

[1] K. Anjo, H. Nishi, X. Dong, A. Yoshiyama, T. Kudoh, H. Nakajo, and H. Amano. JUMP-1/3: A workstation cluster with cache coherent distributed shared memory. In *IEICE Japan SIG Reports, CPSY 96 (in Japanese)*, 1996.

[2] D.Lenoski, J.Laudon, K.Gharachorloo, W.-D.Weber, A.Gupta, J.Hennessy, M.Horowitz, and M.S.Lam. The Stanford DASH multiprocessor. *IEEE Computer*, 1992.

[3] T. Tanaka (Editor). The massively parallel processing system JUMP-1. In *Ohmsha, Ltd/IOS Press*, 1996.

[4] E.Hagersten, A.Landin, and S.Haridi. DDM - a cache-only memory architecture. *IEEE Computer*, 1992.

[5] M. Goshima, S. Matsumoto, H. Nakashima, and S. Tomita. Virtual Queue: A message communication mechanism for massvely parallel computers. In *Proc. of JSPP95 (in Japanese)*, 1995.

[6] K. Hiraki, Hideharu Amano, Morihiro Kuga, Toshinori Sueyoshi, Tomohiro Kudoh, Hiroshi Nakashima, Hironori Nakajo, Hideo Matsuda, Takashi Matsumoto, and Shin ichiro Mori. Overview of the JUMP-1, an MPP prototype for general-purpose parallel computations. In *Proc. of the International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN'94)*, 1994.

[7] J.Kuskin, D.Ofelt, M.Heinrich, J.Heinlein, R.Simoni, K.Gharachorloo, J.Chapin, D.Nakahira, J.Baxter, M.Horowitz, A.Gupta, M.Rosenblum, and J.Hennessy. The stanford FLASH Multiprocessor. In *Proc. of the 21st Annual International Symposium on Computer Architecture*, 1994.

[8] K.Kurihara, D.Chaiken, and A.Agarwal. Latency tolerance through multithreading in large-scale multiprocessors. In *Proc. of International Symposium on Shared Memory Multiprocessing (ISSMM)*, 1991.

[9] T. Kudoh, H. Amano, T. Matsumoto, K. Hiraki, Y. Yang, K. Nishimura, K. Yoshimura, and Y. Fukushima. Hierarchical bit-map directory schemes on the RDT interconnection network for a massively parallel processor JUMP-1. In *Proc. of ICPP95*, 1995.

[10] Tom Levett and Russell Clapp. STiNG: A cc-numa computer system for the commercial marketplace. In *23rd ISCA*, 1996.

[11] H. Nakajo, T. Okada, T. Matsumoto, M. Kohata, H. Matsuda, K. Hiraki, and Y. Kaneda. I/O subsystem for the massively parallel computer JUMP-1. In *Proc. of JSPP95 (in Japanese)*, 1995.

[12] J.P. Singh, W. Weber, and A. Gupta. SPLASH: Stanford parallel applications for shared-memory. In *Tech. Report, Computer System Laboratory, Stanford University*, 1992.

[13] T. Terasawa and H. Amano. Performance evaluation of the mixed-protocol caches with instruction level multiprocessor simulator. In *Proc. of IASTED International Conference of MODELLING AND SIMULATION*, 1994.

[14] T.Matsumoto and K.Hiraki. A shared-memory architecture for massively parallel computer systems. In *IEICE Japan SIG Reports, Vol. 92, No. 173, CPSY 92-26 (in Japanese)*, 1992.

[15] T.Matsumoto and K.Hiraki. Distributed shared-memory architecture using Memory-Based Processors. In *Proc. of Joint Symp. on Parallel Processing'93 (in Japanese)*, 1993.

[16] Y. Yang, H. Amano, H. Shibamura, and T.Sueyoshi. Recursive diagonal torus: An interconnection network for massively parallel computers. In *Proc. of 1993 IEEE Symposium on Parallel and Distributed Processing*, 1993.