[12] D. H. Lawrie. Access and alignment of data in an array processor. *IEEE Transactions on Computers*, 24(12):1145–1155, December 1975.

[13] K. Y. Lee and W. Hegazy. The extra stage gamma network. In *Proceedings of the 13th Annual International Symposium on Computer Architecture*, pages 175–182, Tokyo, Japan, June 1986.

[14] R. J. McMillen and H. J. Siegel. Performance and fault tolerance improvements in the inverse augmented data manipulator network. In *Proceedings of the 9th Annual Symposium on Computer Architecture*, pages 63–72, Austin, Texas, April 1982.

[15] D. S. Parker, Jr. Notes on shuffle/exchange-type switching networks. *IEEE Transactions on Computers*, 29(3):213–222, March 1980.

[16] J. H. Patel. Processor-memory interconnections for multiprocessors. In *Proceedings of the 6th Annual Symposium on Computer Architecture*, pages 168–177, Philadelphia, Pennsylvania, April 1979.

[17] J. H. Patel. Performance of processor-memory interconnections for multiprocessors. *IEEE Transactions on Computers*, 30(10):771–780, October 1981.

[18] M. C. Pease. The indirect binary n-cube microprocessor array. *IEEE Transactions on Computers*, 26(5):548–573, May 1977.

[19] C. S. Raghavendra and A. Varma. INDRA: a class of interconnection networks with redundant paths. In *Proceedings of the 1984 Real-Time Systems Symposium*, pages 153–164, 1984.

[20] D. A. Rennels. Distributed fault-tolerant computer systems. *IEEE Computer*, 13(3):55–65, March 1980. Special issue on Fault-Tolerant Computing.

[21] Kevin B. Theobald. Adding fault-tolerance to a static data flow supercomputer. Master's thesis, Massachusetts Institute of Technology, December 1990. Also published as Technical Report MIT/LCS/TR-499, MIT Laboratory for Computer Science.

[22] Thorsten von Eicken, David E. Culler, Sech Copen Goldstein, and Klaus Eric Schauser. Active messages: a mechanism for integrated communication and computation. In *Proceedings of the 19th Annual International Symposium on Computer Architecture*, pages 256–266, Gold Coast, Australia, May 1992.

[23] C.-L. Wu and T.-Y. Feng. On a class of multistage interconnection networks. *IEEE Transactions on Computers*, 29(8):694–702, August 1980.

Finally, it was shown how to extend the W-Network so that spare processors can substitute for faulty processors. Thus, the W-Network will be a useful part of a complete fault-tolerant multiprocessor.

References

[1] G. B. Adams III and H. J. Siegel. The extra stage cube: A fault-tolerant interconnection network for supersystems. *IEEE Transactions on Computers*, 31(5):443–454, May 1982.

[2] D. P. Agrawal. Graph theoretical analysis and design of multistage interconnection networks. *IEEE Transactions on Computers*, 32(7):637–648, July 1983.

[3] K. E. Batcher. The Flip Network in STARAN. In *Proceedings of the 1976 International Conference on Parallel Processing*, pages 65–71, Detroit, Michigan, August 1976.

[4] L. Ciminiera and A. Serra. A connecting network with fault tolerance capabilities. *IEEE Transactions on Computers*, 35(6):578–580, June 1986.

[5] William J. Dally and Charles L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on Computers*, 36(5):547–553, May 1987.

[6] D. M. Dias and J. R. Jump. Packet switching interconnection networks for modular systems. *IEEE Computer*, 14(12), December 1981.

[7] D. M. Dias and J. R. Jump. Augmented and pruned n log n multistage networks: Topology and performance. In *Proceedings of the 1982 International Conference on Parallel Processing*, pages 10–11, Bellaire, Michigan, August 1982.

[8] L. R. Goke and G. J. Lipovski. Banyan networks for partitioning multiprocessor systems. In *Proceedings of the First Annual Computer Architecture Conference*, pages 21–28, December 1973.

[9] Dana S. Henry and Christopher P. Joerg. A tightly-coupled processor-network interface. In *Proceedings of the Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 111–122, Boston, Massachusetts, October 1992.

[10] M. Jeng and H. J. Siegel. A fault-tolerant multistage interconnection network for multiprocessor systems using dynamic redundancy. In *Proceedings of the 6th Annual International Conference on Distributed Computing Systems*, pages 70–77, 1986.

[11] V.P. Kumar and S.M. Reddy. Augmented shuffle-exchange multistage interconnection networks. *IEEE Computer*, pages 30–40, June 1987.

The W-Network can tolerate a single fault anywhere in the network. The W-Network can also be extended so that processor faults can be tolerated as well, through the use of spare processors. The ordinary switches in stages 0 and $n-1$ can be replaced with W-switch chips. The extra ports added can be connected to spares. This would increase the redundancy costs of the W-Network to 31% for the 64×64 network and 20% for the 256×256 network, but these values are still considerably lower than for other networks, even those which allow spare processors to be included.

The W-Network can be made to tolerate k arbitrary faults by adding k subnetworks to each stage of the MIN instead of 1, and by making the W-switch in each chip a $b \times (b+k)$ switch. A total of k ports must overlap between adjacent W-switch chips in the same subnetwork. Also, the number of connections between each processor and the network must increase to $k+1$. The fault tolerance will increase to k, but the redundancy cost increases rapidly with k. Still, it may be worthwhile if fault tolerance requirements are more severe.

5 Summary

The design of a low-cost fault-tolerant communications network is an important step in the development of a high-performance, large-scale multiprocessor. This paper looked at the current trends in parallel processing in order to derive a set of performance and fault tolerance goals for a network. It was decided that the network should support low-latency transfer of short messages, and that it should be able to continue operation after the occurrence of a single hardware fault without a significant drop in performance. The class of Multistage Interconnection Networks called delta networks was chosen as the basis for a network design meeting the performance and fault tolerance criteria.

Previous fault-tolerant networks were reviewed and judged according to these criteria. These networks were classified according to the main strategies for adding fault tolerance: network duplication, adding extra stages, adding extra links, or adding both extra switches and links. They either suffer significant performance drops when failures occur (compared to an optimally-loaded delta network), require a lot of redundant hardware, or both.

The W-Network was shown to provide the same degree of fault tolerance at a lower cost. The W-Network is a network in which it is possible to embed a regular delta network consisting of a subset of the switches and links. These switches and links are selected using special switches called W-switches that are used to isolate failed components. The W-switches allow a spare component to substitute for many components in the network, which means that fewer spare switches and links are needed in the W-Network than in other fault-tolerant networks.

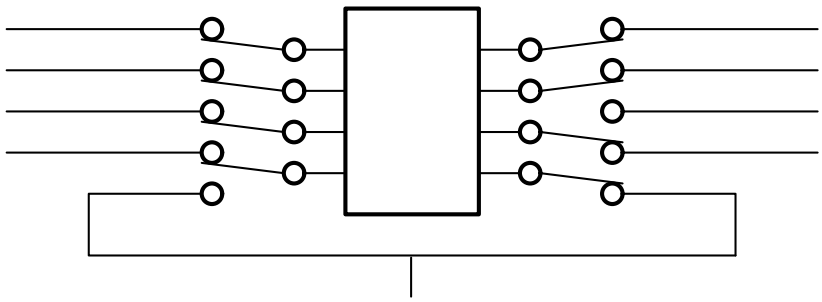

Figure 8: Universal chip design for W-switch crossbar

The W-Network is capable of tolerating any single fault in any switch or link. It can also tolerate two faults if one is located in the first stage and the other in the second stage. There are two main sources of redundancy in the W-Network:

1. Two extra subnetworks must be added to the $2\sqrt{N}$ already in the non-fault-tolerant network (cost = $1 + \frac{1}{\sqrt{N}}$).

2. An extra port must be added to each chip with a W-switch. If all other chips use the original, non-fault-tolerant implementation, then the extra port need only be added to each chip in 2 out of the $\log_b N$ stages. If the linear cost-to-pin count relation is assumed, then the extra cost is

$$1 + \frac{2}{\log_b n}(\frac{1}{2b})$$

The total redundancy cost is the product of the two terms. For instance, the cost is 22% for a 64×64 network made of 2×2 switches, and 13% for a 256×256 network made of 4×4 switches.

There are a few slight additional costs. Output drivers must be made larger, since most will have to drive a line with two other chips attached, instead of one. The greater capacitance could slow down the signals a bit, but the smaller package and chip count may lead to a more compact network, which would partially compensate for this. The W-switch will also add a slight latency compared to the non-fault-tolerant network, but it will be negligible because it is internal and can be integrated with the I/O pins. The delay should be less than the extra delay caused by increasing the size of an internal crossbar, and certainly less than the latency added by an extra stage of switches.

4.3 Extensions

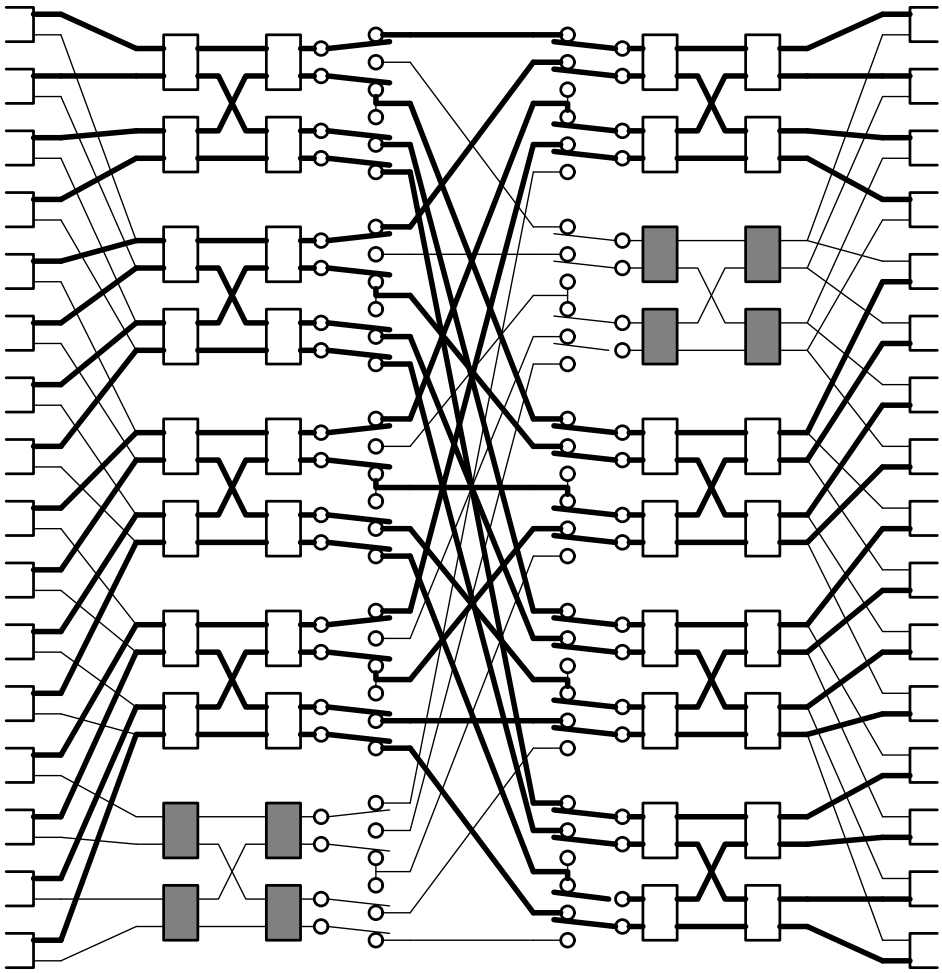

Figure 7: A complete network

single design in which both sides of the internal crossbar are connected to W-switches, and the extra port can be either an input or an output. The design for $b = 4$ is illustrated in Figure 8, in which the extra port is shown being used as an output.

Since the W-switch settings only need to be changed in response to a network fault, transferring the settings from the control processor to each switch doesn't need to be very fast. Since the network would not be sending ordinary data during the diagnosis and reconfiguration phase, the control processor could send special control packets for setting the W-switches through the same network. Thus, no additional pins would be required on the switch chips for this function.

4.2 Redundancy Costs

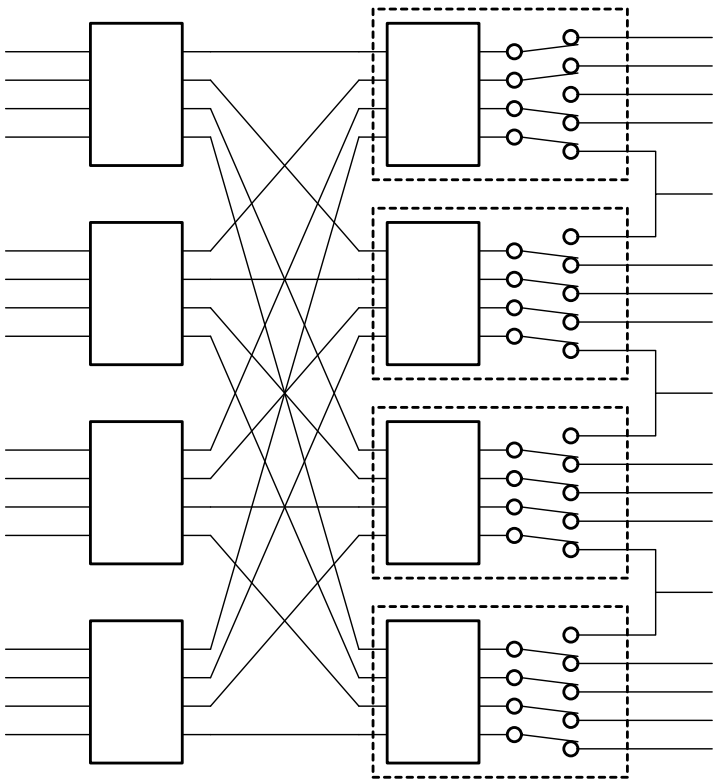

Figure 6: A complete subnetwork

network topology. However, extra hardware is needed for the W-switches.

It is not necessary to make a separate external W-switch. Instead, each switch chip in the last stage of each first-stage subnetwork, and each switch chip in the first stage of each second-stage subnetwork, can have its own $b \times (b+1)$ W-switch. The W-switches can be combined into a larger W-switch by connecting external port b of one switch with external port 0 of the next stage. (Naturally, this requires that the unused external port be left undriven.) Figure 6 shows how to make a 16×17 subnetwork out of 4×4 switches by combining 4 W-switches. The dashed lines indicate the chip boundaries in the second stage of the subnetwork.

Figure 7 illustrates the W-Network by showing a 16×16 network made out of 2×2 switches. The W-switches are configured for bypassing subnetwork 4 of the first stage and subnetwork 1 of the second stage. The bold lines show the active 16×16 delta network, while the thin lines show the inactive links and shaded blocks show the inactive switches. Thus, this is one possible configuration for tolerating any failures of those links or switches.

As presented, the design would require 3 basic chip types, because the W-switches in the first stage of subnetworks are driving output ports, while the W-switches in the second stage of subnetworks are connected to input ports. However, the two chip types can be combined into a

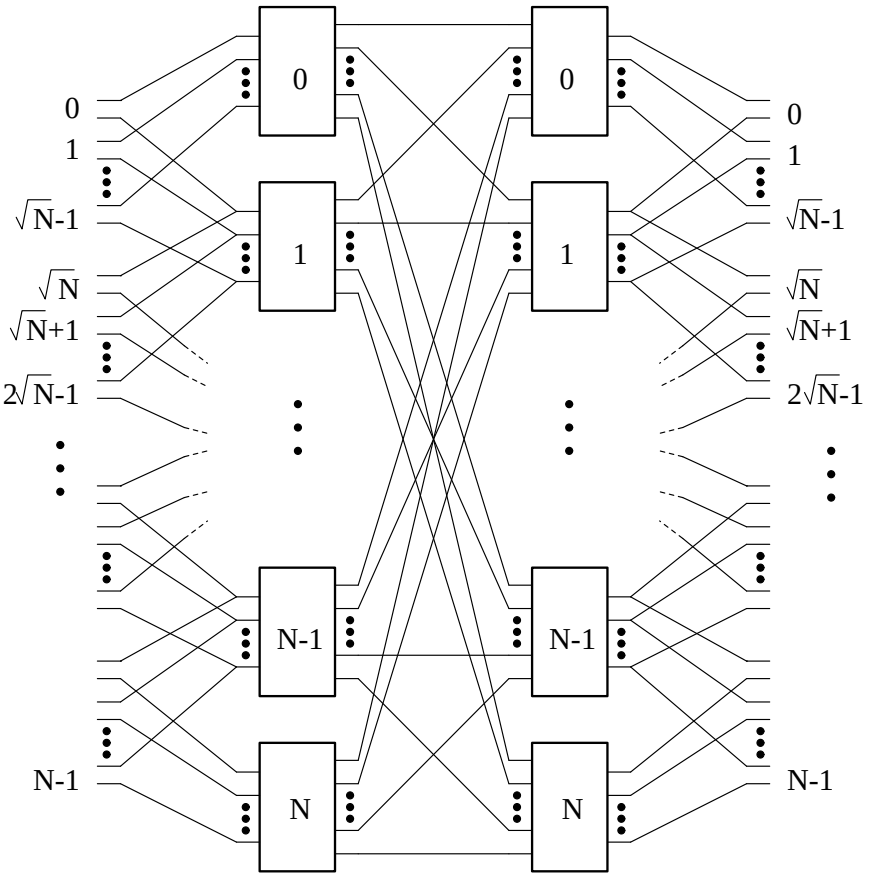

Figure 4: Top-level diagram showing W-Network and processors

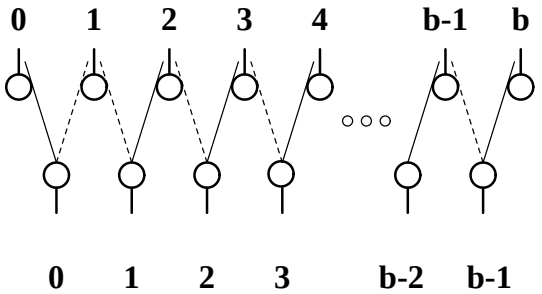

Figure 5: The W-switch

fault in a link between a processor and a subnetwork is treated like a fault in that subnetwork. A fault in a link between stages may be treated as a fault in either of the two subnetworks. If a stage of subnetworks has no fault, then any one of the subnetworks can be switched out.

The subnetwork could be built on a single chip, but this may still be impractical for large N. Therefore, the subnetwork can be built out of smaller switches using any conventional delta

4.1 Design

The first step in building a W-Network is to divide the $N \times N$ network into two stages, each containing $\sqrt{N}$ subnetworks of size $(\sqrt{N} \times \sqrt{N})$. Then an extra subnetwork is added to each stage. Each subnetwork in the first stage is augmented by one extra output port, and each subnetwork in the second stage is augmented by one extra input.

Each stage now has $(\sqrt{N} + 1)$ subnetworks. These subnetworks are then fully interconnected with $(\sqrt{N} + 1)^2$ links. Each processor has two input connections to the MIN and two output connections. Two are needed because if each processor were connected to a single switch, a fault affecting a single switch could isolate b processors. If the input ports on the $(\sqrt{N} + 1)$ subnetworks in the first stage are consecutively numbered from 0 to $\sqrt{N}(\sqrt{N} + 1)$, then processor i is connected to ports i and $i + \sqrt{N}$. The output is similarly connected.

A top-level view of the W-Network with the dual inputs and dual outputs is shown in Figure 4. Note that each input to the first stage is connected to the outputs of two processors. Only one of these two outputs can be used, so each processor only drives one of its outputs and leaves the other output undriven.

Each subnetwork contains $\sqrt{N}$ ports on one side and $\sqrt{N}+1$ ports on the other. However, only $\sqrt{N}$ ports are active on each side at any one time. In the first stage, the outputs of the internal crossbar can be selectively connected to the $(\sqrt{N}+1)$ subnetwork outputs, so that one of the subnetwork outputs can be left disconnected. Similarly, in the second stage, the inputs of the internal crossbar can be selectively connected to the $(\sqrt{N} + 1)$ subnetwork inputs.

The connection between the internal switch and the outside is made with a *W-switch*. Figure 5 shows a W-switch for a b-port network or chip. For each port i of the internal crossbar, a switch can connect it to one of two external ports, either i or $i + 1$.[2] The switches connect the internal ports to all of the external ports but one, maintaining the order of the internal ports. This switch configuration is a static setting which is only changed when the network must be reconfigured due to a fault. In Figure 5, external port 1 is isolated, while all others are connected to the internal switch.

If subnetwork $i(0 \leq i \leq \sqrt{N})$ in the first stage has a fault, then all processors between 0 and $\sqrt{N}i - 1$ (inclusive) activate their top output links, while all other processors activate their bottom links. All inputs in the second stage subnetworks are switched to ignore the i^{th} input. Similarly, if subnetwork i in the second stage is faulty, then all processors between 0 and $\sqrt{N}i - 1$ (inclusive) read their top input links, while all other processors use their bottom links. All outputs in the first stage subnetworks are switched to disconnect the i^{th} output. A

[2]The name "W-switch" comes from the shape formed by the superposition of all possible switch positions when $b = 2$.

Network	64×64 with 2×2 switches				256×256 with 4×4 switches			
	Extra Chips	Extra Links	Perf. Drop	Total Cost	Extra Chips	Extra Links	Perf. Drop	Total Cost
Multiple-copy (e.g., INDRA)	1	1	2	2	1	1	2	2
Extra-stage (e.g., ESC)	7/6	1	2	2.33	5/4	1	4/3	1.67
F network	1	2	2	4	(not designed for $b > 2$)			
Enhanced IADM	1	61/28	3/2	3.27	(not designed for $b > 2$)			
Augmented SE network	1	3/2	2	3	1	5/4	4/3	1.67
Extra Stage Gamma	7/6	3/2	3/2	2.63	(not designed for $b > 2$)			
Dynamic Redundancy	33/32	3/2	1	1.54	65/64	7/4	1	1.78
W-Network	9/8	13/12	1	1.22	17/16	17/16	1	1.13

Table 1: Comparison of fault-tolerant networks

these values will change if hardware costs are computed differently (e.g., link cost measured quadratically instead of linearly), such a change only makes the difference between the W-Network and the other networks even greater.

4 The W-Network

The FTMIN designs in the previous section had high redundancy costs because each spare link or switch could only replace a small number of faulty components. Ideally, in a homogeneous system with many identical components, a spare should be able to replace *any* failed component. While it would clearly be impractical to create, for instance, a single spare link that can replace *any* link, a FTMIN design should enable a spare link or switch to replace as many links or switches as possible.

The following presents the design of the W-Network. This FTMIN, like the Dynamic Redundancy network, is a superset of a basic MIN, At any given time, a subset of the total network is selected to form an ordinary delta network. Therefore, there is no performance drop due to load imbalances. However, the W-Network has low redundancy, mainly because each spare component can substitute for as many as $\sqrt{N}$ other components. The W-Network can be built using only a few chip types. The general design is illustrated by building a 16×16 W-Network out of 2×2 switches. The rest of the section measures the redundancy and suggests ways to extend the basic network.

1. A 64×64 network made of 2×2 switches, and

2. A 256×256 network made of 4×4 switches.

The values in the table represent the ratio between the hardware needed by the fault-tolerant network and the hardware needed by a corresponding optimally designed non-fault-tolerant delta network. The non-fault-tolerant network has the same performance characteristics as the fault-tolerant network *after* a fault has occurred. Measurements of the effect of faults on performance are made according to the assumptions made in Section 2 about how the network will be used by a multiprocessor. The results may not be valid given a different set of assumptions, e.g., localized or sparse network traffic.

It is extremely difficult to quantify redundancy costs exactly. While chip count is easy to measure, other costs, such as extra links, are harder to measure because of the multitude of tradeoffs available to the designer. For instance, how should the cost of extra links be measured? If data paths are kept constant, and extra ports are added to each chip, the area of the chip will increase quadratically with the increase in ports, as previously mentioned. However, as in Section 3.1, the data paths may be distributed among multiple chips. This would imply a linear increase in cost, but doesn't account for the redundant control information needed.

Since the goal of this measurement is to show the dramatic improvement achieved by the W-Network, we make simple assumptions about the costs which are more generous to the other networks. For all networks, hardware costs are linearly proportional to the increase in chip count. If extra ports are added to each switch, the hardware cost is linearly proportional to the increase. Costs due to performance losses indicate the increase in traffic over the most overloaded link or switch following a worst-case fault. The values listed assume that the redistribution of messages is as even as possible under this fault. The total cost is the product of the three cost components. Extra latency from lengthening a path (as in extra-stage networks) is not counted, because the use of wormhole routing may make it small compared to the total time needed to send a message through the network. Also, the cost of creating two links from each processor to the network (as shown in Figure 3) are not included as it is a cost which any fault-tolerant network will have to bear in order to be fully fault-tolerant.

The cost of the multiple-copy network is for a duplicated network. The extra-chip cost of the DR assumes one row of spares. The cost for extra links in the enhanced IADM is based on the average over all the various stages. the hardware cost component. The IADM, F-network and Extra Stage Gamma network are designed for $b = 2$. The extra-stage, Augmented Shuffle Exchange, and Dynamic Redundancy networks were originally only designed for $b = 2$, but there are straightforward extensions to higher values of b.

The W-Network, which is presented in the next section, is included in the table for comparison. The next section shows how the values of 22% and 13% were computed. Although

For instance, suppose a 2×2 switch is increased to 4×4 in order to add fault tolerance, and the data paths are halved in order to keep chip size the same. If this can be done without decreasing system performance, then the non-fault-tolerant delta network can be converted from 2×2 switches to 4×4 switches, cutting chip count by a factor of 4! The 4×4 switches would only have half the bandwidth of 4 2×2 switches, due to the reduced data paths, but the savings in hardware would still be a factor of 2.

Extra-link designs also have the same problem with load-balancing as the extra-stage designs in Section 3.2. Since they add extra links but *not* extra switches, a faulty switch will force the messages that normally go through that switch to be diverted to other switches. The topologies don't allow this traffic to be uniformly distributed over the network, so certain switches will be more heavily loaded than others.

3.4 Redundant-Switch Networks

Some networks address the load-balancing problem by adding extra *switches* as well as extra links. In many cases, the links and switches in the fault-tolerant network form a superset of the links and switches in a basic non-fault-tolerant MIN. At any given time, only a subset of the links and switches are active, and this active set forms a basic MIN.

The *Dynamic Redundancy* network [10] is one example of this. The DR network starts with a basic delta network with 2×2 switches, and adds a third link to and from each switch. An arbitrary number of extra switches can be added in each stage, provided the same number is added to each stage. Each switch is 3×3, but only 2 inputs and 2 outputs are active at any given time. It is possible to form the basic delta network MIN from any set of $N/2$ contiguous rows (the top and bottom rows are contiguous). The advantage of this system is that it includes processors; the mechanism allows spare processors to substitute for faulty processors.

Since this network always retains the connectivity of a delta network when properly configured, there is no problem with load imbalance. The biggest disadvantage with the DR, given our assumptions, is the high cost of the extra pins on the 3×3 switches.

3.5 Comparison

In this section, the networks surveyed above are compared in terms of their redundancy costs for two cases:

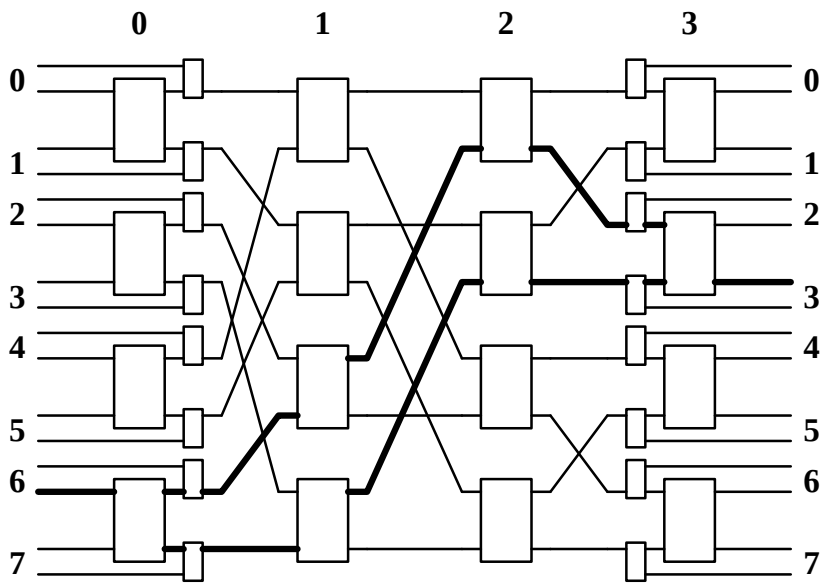

Figure 3: 8×8 Extra-Stage Cube Network

Another way to tolerate link and node failures is to create extra links that can be used to make new pathways that bypass failed components. This is usually done by augmenting each $b \times b$ switch chip with additional input and output ports. In some cases, the size of the internal crossbar is also increased, because all of the extra ports may be in use at the same time. In other designs, the internal crossbar is still $b \times b$, because only b inputs and outputs are active in any configuration, and this is changed only when the network is reconfigured after a fault.

There are many examples of this approach. The *F-network* [4] doubles the number of switches and changes each 2×2 switch into a 4×4 switch. Extra links between stages are added, which allows alternate paths to be created to bypass failures in all but the first and last stages. The *enhanced Inverse Augmented Data Manipulator* [14] uses a combination of 3×5, 4×5, and 5×5 switches. The *Augmented Shuffle-Exchange* network [11] keeps a basic delta network, but adds links between switches within the same stage in order to provide alternate paths. This is done with 3×3 switches. The *Extra Stage Gamma* network [13] combines extra links (with 3×3 switches) and an extra stage. None of these can handle faults in the first or last stage, so additional hardware would be needed for complete fault-tolerant.

A major disadvantage of these designs, given the requirements for wide data paths assumed in this paper, is the high costs associated with the extra links. Extra ports on a chip will require a bigger chip or smaller port widths. In all the designs above, the extra ports are also active, so the total bandwidth is not being wasted. But there is an opportunity cost associated with the increase in port count, because it precludes the possibility for further integration of the base network.

sends a message. An example of this is the *INDRA* network [19]. Such a network can tolerate at least $k-1$ network faults.

Unfortunately, the redundancy cost is high. If a single fault occurs, and the load is evenly redistributed, the load on each healthy subnetwork increases by $\frac{k}{k-1}$. As argued in the previous section, this represents a redundancy cost. For instance, if k is 2, then there are two parallel networks. If one switch is faulty, then all the affected traffic has to go through the other network. Therefore, both networks have to have the same performance as the original non-fault-tolerant delta network, which is a hardware cost of 2.

As k increases, the relative redundancy cost drops. However, for large k, the width of the data path in each subnetwork would be decreased so much that the latency might increase significantly, which would decrease total system performance. Alternately, a bit-sliced approach can be used, where messages can be partitioned and sent in parallel over a subset of the networks. However, this would either require duplicating the control information in a message for all partitions (expensive for short messages) or coordinating the parallel networks by exchanging control information, which will add to latency.

3.2 Extra-Stage Networks

Another way to make a MIN fault-tolerant is to add an extra switching stage. The purpose is to create more than one path between any input-output pair. This way, if one path is unavailable due to a fault, one of the other paths can be used instead. Special switches are needed to bypass the first or last stage, since these stages are connected directly to the processors. The principle is illustrated in Figure 3 with an 8×8 *Extra Stage Cube* network [1] made from 2×2 switches. The bold lines show the alternate paths from input 6 to output 3. The *augmented delta network* [7] is another example.

Extra-stage networks, regardless of their particular topologies, suffer from the same problem of imbalanced load-shifting. Suppose that a switch in stage 1 fails. Then all the switches in stage 0 must be used. Now consider a switch i in stage 0 which is connected to the failed switch. All messages passing through switch i must still go through that switch. However, since one of the b output links is no longer available, the messages must be redistributed over the remaining $b-1$ links. Even if the redistribution is perfectly even, the traffic on each viable link will increase by $\frac{b}{b-1}$. By our assumptions, this represents a redundancy cost in *addition* to the cost of the extra stage.

3.3 Redundant-Link Networks

These costs should be measured by the standards of current VLSI technology. Presently, a lot of switching logic and buffering can be packed into a single chip. Since wiring inside a chip is cheaper than external wiring, it is desirable to have a topology which minimizes the number of chip boundaries. However, chips are more limited in the number of I/O signals they can have.

If data paths are to be wide, then today's chip technology requires a fairly small b. Therefore, neither the internal $b \times b$ crossbar inside a switch chip nor the input/output buffers won't be the main factor limiting chip area. The main factor will be the total number of pins on the chip. If a chip is pin-bound, then its area is proportional to the square of the number of pins. This is because the connections can only be made at the perimeter of a chip. The same is true for most types of packaging.

When measuring the costs of fault tolerance, it is just as important to count any performance drops resulting from a fault. Suppose that a fault-tolerant network shifts some of the traffic from a faulty link to a healthy link, while the traffic already on the healthy link stays there. If the increased load on the link exceeds the bandwidth, then the latency of some messages will increase. In the worst case, this could slow down the entire computation if the affected messages were on critical paths. The bandwidth of each link can be increased to handle this condition (e.g., by widening the data paths or creating multiple copies of the network), but then this is an increase in hardware, which becomes part of the total redundancy cost. On the other hand, if the increased traffic never exceeds the bandwidth, then the network had excess capacity when there were no faults. This represents an "opportunity cost" because this excess capacity could be removed from the non-fault-tolerant network, hence cutting its cost. Thus, either way, the uneven distribution of traffic following a fault imposes a redundancy cost.

Many fault-tolerant network proposals look costly when seen from this perspective. (This does *not* mean that they are bad designs. Many have other good properties, such as higher levels of fault tolerance or better avoidance of congestion. They are simply too costly for the type of multiprocessor we are considering.) The following sections review some well-known fault tolerance approaches. The end of this section lists the redundancy costs of each for two representative MIN sizes.

3.1 Multiple-Copy Networks

A straightforward way to add fault tolerance to a MIN is to replicate the network k times, making the network able to tolerate $k-1$ faults. Additional switching is required, either by adding reliable $1 \times k$ and $k \times 1$ switches before the first stage and after the last stage, respectively, or by providing k ports per processor and letting the processor pick a subnetwork every time it

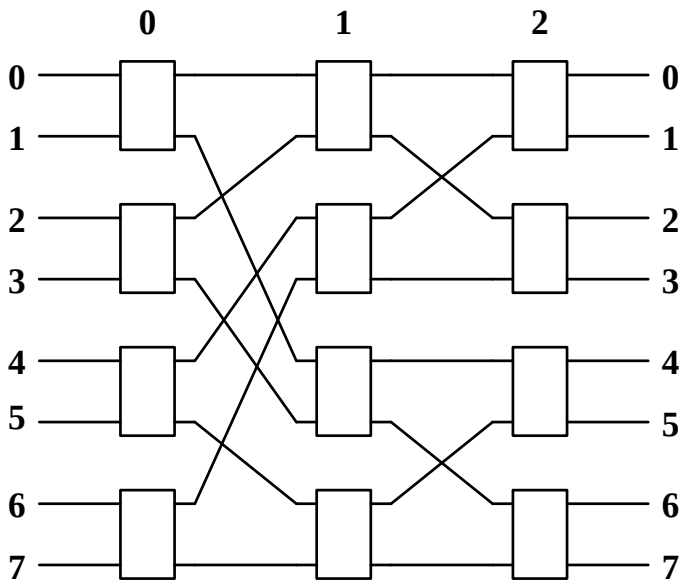

Figure 2: 8×8 Delta Network

Therefore, the general problem addressed in this paper can be summarized as follows:

> *Construct a single-fault-tolerant full-access MIN similar to a delta-network, with N inputs and outputs, and approximately $\log_b N$ stages, where b is the switching factor at each stage. The network should have sufficient speed and bandwidth to provide low-latency transfers for a high, uniform density of inter-processor messages. It should be able to maintain normal operation, without significant performance degradation, even when a single component (link or switch) anywhere in the network has completely failed.*

3 Previous Designs

This section reviews some of the more well-known fault-tolerant networks and examines the costs of the redundancy hardware needed to add the fault tolerance. Space limitations preclude formal definitions or implementation details of the networks, so readers should check the original sources cited. Only enough details are provided for measuring performance and redundancy costs.

There are two redundancy costs associated with a fault-tolerant network. The first is the extra hardware itself. The addition of redundant switches is an obvious increase in cost. However, the cost of redundant links should not be ignored.

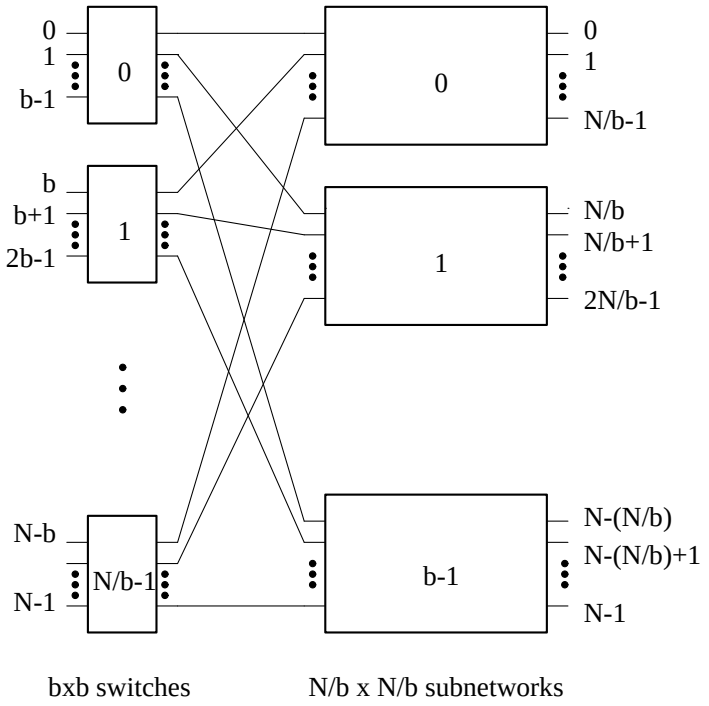

Figure 1: $N \times N$ Delta Network

i, exactly b^{n-i} outputs of the MIN are accessible. The switching element to which this port is an input partitions the set of accessible MIN outputs into b disjoint sets, each of size b^{n-i-1}.

One of the easiest delta networks to understand is the baseline network, so it is shown here to illustrate the class. The $b^n \times b^n$ baseline network is constructed recursively. Each input goes into one $b \times b$ switch, which directs it to one input of a $b^{n-1} \times b^{n-1}$ baseline network, as shown in Figure 1. An 8×8 network, made of 2×2 switches, is shown in Figure 2.

Many packet-switched delta networks are *self-routing*. Each packet contains the destination encoded as an n-digit radix-b number. When the packet reaches stage i, the switch in that stage sends the packet to the output specified by the i^{th} digit of the destination.

Since this paper is concerned with fault tolerance in MIN's, we assume that all components of the MIN, including switches and links, are subject to failure. If there is a failure of a link between two stages, or of a link between the first or last stage and the outside, then no messages may travel along that link. The chips that make up the switching stages may fail in two ways. A failure of a single input or output port, such as a break in a bonding wire or a failure of an I/O pad or buffer, is equivalent to a failure in the corresponding link. A failure of the crossbar switch makes the entire switch unusable, and no useful messages may flow through any pins on that chip.

2. *Fast and frequent communication:* In order to exploit parallelism effectively, processors will need to send messages to other processors quickly. Therefore, the data paths in the network should be wide enough so that a message can be transmitted in a small number of cycles. Furthermore, processors will need to exchange data frequently, so the network should have sufficient bandwidth to avoid being saturated by this traffic, or else the processors will sit idly waiting for data to arrive.

Delta networks [16] are the most common class of MIN's with $O(\log N)$ stages per path, and offer a good balance between latency and bandwidth characteristics and cost, especially when enhanced with techniques such as buffers in the switches to mitigate contention problems [6] and *wormhole routing* [5] to decrease latency. A packet-switched delta network is a MIN with the following properties:

1. There are N inputs and N outputs, where $N = b^n$ (b and n defined below).[1] Each set of ports is numbered from 0 to $N - 1$.

2. The fundamental building block of the MIN is a switching element with b input ports and b output ports ($b \geq 2$). Each set of ports is numbered from 0 to $b - 1$. A packet received at any of the b inputs can be sent to any output. (Henceforth, this element is called a $b \times b$ switch, or simply a switch when the size is unambiguous.)

3. The MIN contains n stages, each with b^{n-1} $b \times b$ switches. The network input ports are connected to the inputs of the first stage, and the network output ports are connected to the outputs of the last stage.

4. Each output of a stage is connected to exactly one input of the next stage. Thus, there are exactly N links between any two stages.

5. There is exactly one unique path between a given MIN input and a given MIN output. This path is described by an n-digit radix-b number, where each digit indicates the destination port of one of the switches in the path.

In this paper, the n stages of a delta network are numbered from 0 to $n - 1$ (the inputs connect to stage 0). The switches in each stage are numbered from 0 to $b^{n-1} - 1$.

There are many MIN's in this class, including the *baseline* [23], *SW-banyan* [8], *omega* [12], *flip* [3], and *indirect binary n-cube* [18] networks. They are topologically equivalent [2, 15, 23], so the choice between them is mainly a matter of which one is easiest to implement in a particular system. The important property they all have in common is that from any input port in stage

[1] This was later extended [17] to include nonsymmetrical, $a^n \times b^n$ networks.

been replaced by a technician will be sufficiently low as to eliminate the requirement that the system be able to tolerate multiple failures. Therefore, the system only has to tolerate single failures.

It is clear, though, that single failures will occur many times during a system's lifetime. Furthermore, several days may elapse between the time the failure occurs and the time it is repaired (for instance, if a part fails after the technicians have gone home for the weekend). There would be a severe loss of computing time if the system performance were to drop significantly during this time period. Therefore, our fault tolerance goals include the requirement that there is little or no drop in system performance after a single failure has occurred.

1.3 Synopsis

The remainder of this paper is divided into four sections. The first section gives a more detailed problem statement. It describes the *delta network*, which serves as the basis of most FTMIN's, including the W-Network, and presents the assumptions about what kinds of faults can occur (the *fault model*) and the fault tolerance criteria used in this study. Section 3 looks at many published fault-tolerant designs and compares their costs and performance levels. The W-Network is presented in Section 4, along with some possible extensions, including how the W-Network can be used in a complete system to achieve processor fault tolerance as well. The final section summarizes and concludes this paper.

2 Problem Statement

This paper is concerned with MIN's designed for large-scale multiprocessors used for executing highly-parallel applications. (For example, the W-Network was originally developed for a static dataflow machine with a large number of processors [21].) It is assumed here that such machines and their applications have the following properties and requirements:

1. *Uniform traffic:* A processor is likely to send messages to many different processors, not just a few nearest neighbors. Therefore, the performance measurements made in this paper use the following traffic model: If $M(i, j)$ is the average number of messages sent from processor i to processor j per unit of time, then $M(i, j)$ is the same for all i and j. (MIN's are most suited to this type of traffic. If the message traffic is more localized, then other network topologies, such as trees, meshes, or hypercubes, may be more appropriate.)

1.2 Fault Tolerance

Fault tolerance is an important aspect of any multiprocessor design. A massively parallel computer is likely to have hundreds or thousands of IC chips, Even if the reliability of a single chip is high, the presence of thousands of chips, each of which may fail, leads to a much lower aggregate level of reliability for the whole system. Thus, it is necessary to design a system which can continue to operate correctly even when failures occur in the hardware.

Since the MIN itself is made of failure-prone components, it must also be made fault-tolerant. Furthermore, a fault-tolerant MIN (FTMIN) should provide its fault tolerance at low cost. This means that if any extra hardware is required, the amount should be small. Also, the addition of fault tolerance should not lead to a significant reduction in performance, either before or after a failure has occurred.

Unfortunately, many of the proposed designs for fault-tolerant MIN's fail at least one of these two criteria. Either the hardware cost is excessive, or there are large losses in performance, or both. (This is analyzed in more detail in Section 3.) In some cases, the high cost or performance drop is unavoidable, because the design is required to have a very high degree of fault tolerance, i.e., to withstand a large number of faults without suffering a systemic failure. In other cases, the designs were analyzed using assumptions that were valid at the time, but have been made obsolete by changing technology.

Before comparing various fault-tolerant systems, it is necessary to state the criteria for successful fault tolerance. There is no universal standard for fault tolerance, and different applications can demand different types and degrees of fault tolerance depending on their reliability requirements, architectures, environments, and maintenance schedules. For instance, *critical* systems [20] are used in applications, such as the control of dynamically-unstable aircraft, where a failure can lead to disastrous consequences. Therefore, fault tolerance requirements are stringent.

The W-Network is primarily intended for *high-performance* computers, which are mainly used for large number-crunching problems. These machines don't have the stringent reliability requirements of critical machines. Hence, expensive techniques used by those other systems to ensure continuous operation may be replaced with cheaper designs that incur only minor amounts of down-time. What is important is the average throughput of the system.

In such a machine, components will fail independently and at random times, except when there are such common-mode failures as power-supply malfunctions. The machine's users will be able to perform some manual repairs of the system, such as running diagnostic routines and replacing Field-Replaceable Units that are pinpointed by the diagnostics. It is generally assumed that the probability of a second failure occurring before a first failed component has

1 Introduction

For large-scale multiprocessors, an efficient interconnection network is crucial to good performance, because the effective partitioning of most problems on parallel machines requires frequent, rapid exchange of data and commands among different processors and memory units. Fault tolerance, the ability of a computer to continue correct operation even when some components in the system have failed, is also important for such systems, because of the large number of components that can fail. The interconnection network in a massively parallel machine should be able to maintain full connectivity in the event of a fault, and should also provide an efficient means for the computer to replace faulty processors with spare units. Many methods have been proposed for adding fault tolerance to a particular class of network called Multistage Interconnection Networks (MIN's). This paper gives an analysis of the redundancy costs of some of these methods, then proposes a design for the *W-Network*, a new fault-tolerant MIN (FTMIN) which is well-suited to a multiprocessor, provides complete tolerance of single faults, includes the ability to swap in spare processors, yet is much cheaper than other designs.

1.1 Multistage Interconnection Networks

It is increasingly being recognized that network performance is essential to good multiprocessing. The current trend in interprocessor communication is toward fast, low-latency networks transmitting short messages [9, 22]. This makes it easier to exploit fine-grained parallelism. High bandwidth is also important because this exploitation generally requires frequent exchange of data among processors. A corollary is that it is also important that each processor can send a message to any other processor in a single pass through the network. The W-Network presented in this paper is primarily designed for multiprocessors needing this kind of communication performance.

For large-scale multiprocessors, bus-based systems don't have adequate performance. Complete point-to-point communication (e.g., a crossbar switch) is too expensive or impractical to build, due to the quadratic hardware growth. Therefore, many designers have settled on the MIN as a compromise between the two extremes. A MIN divides the task of routing among multiple stages which are built out of identical or nearly-identical components. The most common designs have $O(\log N)$ stages, where N is the number of processors. This paper is concerned with networks of this type. Furthermore, it assumes that the network is square (same number of inputs as outputs) and packet-switched, though the W-Network design can be adapted to other kinds of networks as well.

List of Figures

1 $N \times N$ Delta Network 5

2 8×8 Delta Network 6

3 8×8 Extra-Stage Cube Network 9

4 Top-level diagram showing W-Network and processors 14

5 The W-switch 14

6 A complete subnetwork 15

7 A complete network 16

8 Universal chip design for W-switch crossbar 17

List of Tables

1 Comparison of fault-tolerant networks 12

Contents

1 Introduction **1**

1.1 Multistage Interconnection Networks 1

1.2 Fault Tolerance 2

1.3 Synopsis 3

2 Problem Statement **3**

3 Previous Designs **6**

3.1 Multiple-Copy Networks 7

3.2 Extra-Stage Networks 8

3.3 Redundant-Link Networks 8

3.4 Redundant-Switch Networks 10

3.5 Comparison 10

4 The W-Network **12**

4.1 Design 13

4.2 Redundancy Costs 16

4.3 Extensions 17

5 Summary **18**

Large-scale multiprocessors require an efficient interconnection network to achieve good performance. This network, like the rest of the system, should be *fault-tolerant* (able to continue operating even when there are hardware failures in the system). Many methods have been proposed for adding fault tolerance to Multistage Interconnection Networks (MIN's), but these methods incur high redundancy costs. This paper presents the *W-Network*, a low-cost fault-tolerant MIN which is well-suited to a multiprocessor intended for high-performance scientific computations. It provides complete tolerance of single faults without any loss of access or performance, yet is substantialy cheaper than previous designs. The redundancy cost is 22% for a 64×64 network made of 2×2 switch chips, and only 13% for a 256×256 network made of 4×4 switch chips. This is a significant reduction compared with the previous designs, in which the lowest redundancy costs for corresponding networks are 54% and 67%, respectively, when all hardware costs and performance degradations are taken into account. For a slight additional increase in hardware, the W-Network can be enhanced with extra ports for replacing faulty processors with spare units.

McGill University
School of Computer Science

The W-Network: A New Low-Cost Fault-Tolerant Multistage Interconnection Network

Kevin B. Theobald

ACAPS Technical Memo 50
November 15, 1992

theobald@cs.mcgill.ca

Advanced Compilers, Architectures and Parallel Systems Group