Finding an optimum assignment of relays that minimizes the function $\hat{L}$ is difficult and expensive in terms of computation time[1]. Instead of an algorithm that finds an optimum assignment, we use an algorithm that sequentially selects relays in a greedy manner. We first compute the loads for the direct fault-free paths as follows. Let $C(p,q)$ denote the critical channel on the path from p to q. For all pairs (p,q) such that $p \rightarrow q$, we increment the load $L_{C(p,q)}$ by $1/N_A$. Next we pick pairs of PEs (p,q) such that $p \not\rightarrow q$ one at a time and assign relays for them. For each $r \in R_{p,q}$, we evaluate $\max(L_{C(p,r)}, L_{C(r,q)})$ based on the current loads on the critical channels. We choose that r for which this function is the smallest. If many PEs have the same minimum value, we choose one out them at random. We assign this PE as the relay for pair (p,q) and increment the load on the corresponding two critical channels by $1/N_A$. We repeat this until all relays have been selected.

Our relay selection algorithm has a time complexity of $O(N^3)$ and a space complexity of $O(N^2)$. Despite the high asymptotic time complexity, computing the tables for a machine of size 1024, and 200 faults, requires only about 10 Megabytes of memory and takes less than 15 minutes on a 16 Mhz RISC work-station.

8.3 Related work

A large amount of work has been done on the topic of fault-tolerance of butterfly networks. A significant fraction of this work consists of techniques that augment the network by adding extra links or additional stages. See [IAS87] for a survey.

Varma and Raghavendra [VR89] have studied a problem similar to the one discussed in this chapter. They provide a characterization of fault patterns in a butterfly network based on how many passes are required to guarantee full connectivity between PEs. They base their analysis on the absence of faults in the first and last columns of routing nodes and show that under certain conditions 3 passes are sufficient to establish full connectivity. If a set of weaker conditions is satisfied, they show that a maximum of $\log N - 2$ passes is sufficient. Their main concern is connectivity; balancing the load on the channels and maximizing network throughput is not considered. We limit ourselves to 2 passes through the network since increasing the number of passes can cause significant degradation in throughput. In practice, network throughput is probably as important as connectivity. If all messages require $\log N$ passes, the network throughput will degrade by

[1] We suspect that this problem is also NP-complete.

at least a factor of $\log N$, and if all the processors are active, it is likely they will run at $1/\log N$ of their normal speed.

In a recent paper [LMS92] Leighton et al. analyze the fault-tolerance properties of butterflies from a theoretical viewpoint. They consider a network that has a PE at each node of the butterfly. They prove that even when a large number of nodes are faulty, the network can emulate a fault-free butterfly with $O(1)$ slowdown. Their routing algorithm is similar to ours in the sense that messages essentially make two passes over the butterfly. They select relays at random and show that the congestion in the network is $O(\log N)$ with high probability. They then use Ranade's scheduling algorithm [Ran88] to route messages in $O(\log N)$ steps. Their results are not directly comparable to ours since our network architecture is different, but the underlying ideas are similar.

In [LLM90], Leighton et al. evaluate a network called the multibutterfly [Upf89] which has excellent fault-tolerance properties. A multibutterfly consists of a pair of superimposed butterfly networks, where the nodes of one of the networks are randomly permuted within each column. The random wiring between the stages helps in ensuring a high degree of fault tolerance, but it also destroys the partitionability of the network. The pin-requirements of a partitioned multibutterfly are significantly greater than a partitioned butterfly.

8.4 Performance results

The fault-tolerance scheme presented earlier can be applied to any of the networks presented earlier in this dissertation. In this chapter, we evaluate this scheme for the network *Butterfly.32.8*. This network has 1024 processors, divided among 32 modules, with 32 processors and memory units (processing elements) in each module. The intra-module channels are 32 bits wide and the inter-module channels are 8 bits wide. The network has a total of 5120 routing nodes, organized as 10 columns of 512 nodes each. Faults are placed on the routing nodes at random.

We first measured the cost of limiting the message paths to a maximum of two passes. If we permitted messages to make more than two passes we might potentially be able to include more PEs in the *alive* set. Figure 8.1 shows the fraction of PEs that remain alive in the presence of faults. The line labeled `Alive` is the actual fraction of PEs that remained alive under our scheme and the line labeled `Max alive` is the fraction of PEs that

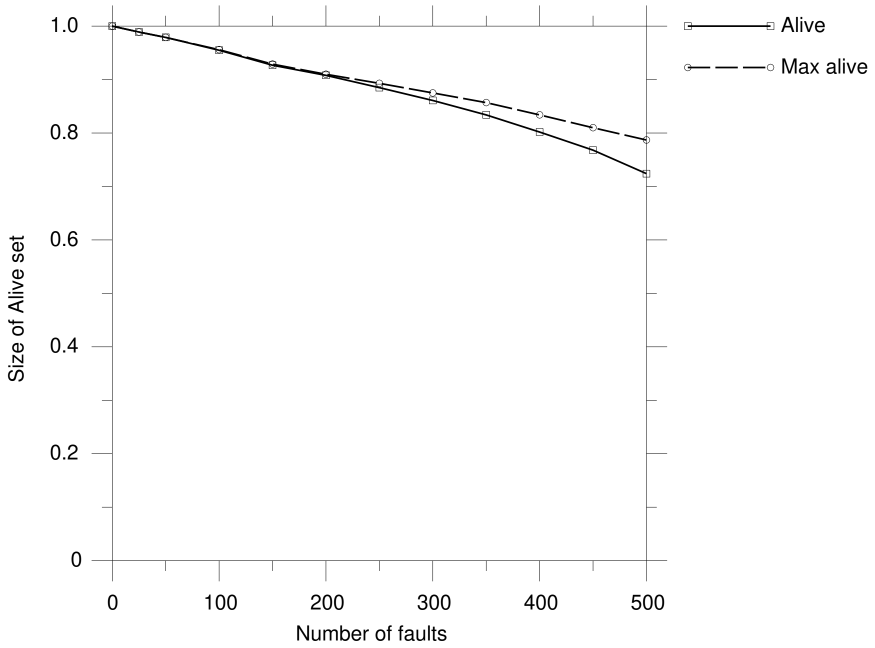

Figure 8.1: Cost of limiting paths to two passes

would remain alive if we had permitted an unlimited number of passes through the network. We can clearly see that when the number of faults is small (less than 200), two passes are sufficient. Almost all the PEs that are lost are those that are connected to faulty routing nodes in the first or the last column. Even when there are 500 faults, the discrepancy between the two lines is only 6% indicating that, in practice, two passes are sufficient.

In figure 8.2 the line labeled `Greedy` shows the value of the worst case load ($\hat{L}$) that was obtained by the relay selection algorithm described in the previous section. The line labeled `Random` shows the value of $\hat{L}$ for an algorithm that assigns relays at random from among the set of valid relays. The line labeled `Average` is the average load over all the critical channels. We can see that our algorithm performs much better than `Random`, and that the peak load is within 30% of the average when the number of faults is less than 200.

We first simulated the network under a saturating load where each *alive* PE sent a message to a random destination whenever the network was capable of accepting a message. We measured the peak throughput as the average number of messages delivered per unit time in steady state. Messages were routed using virtual cut-through routing (see section 3.4 for a description of the routing nodes, buffering, etc). Figure 8.3 shows the observed peak

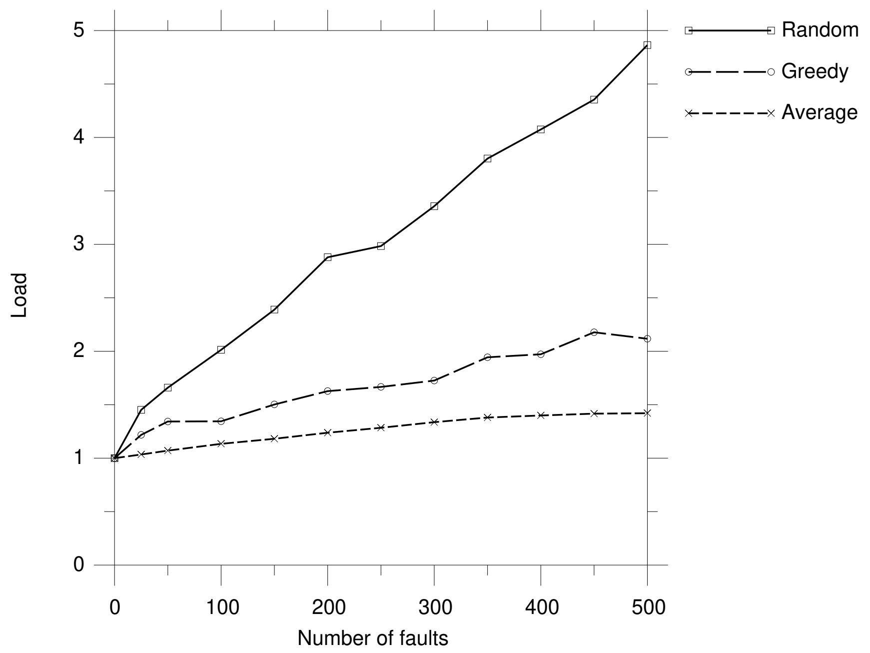

Figure 8.2: Performance of the relay selection algorithm

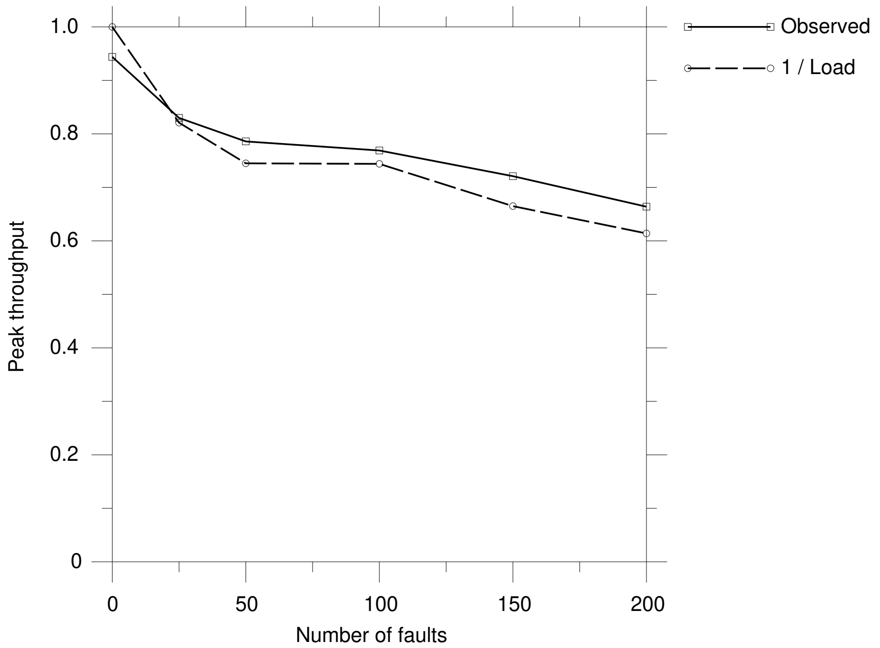

Figure 8.3: Peak throughput under saturation load

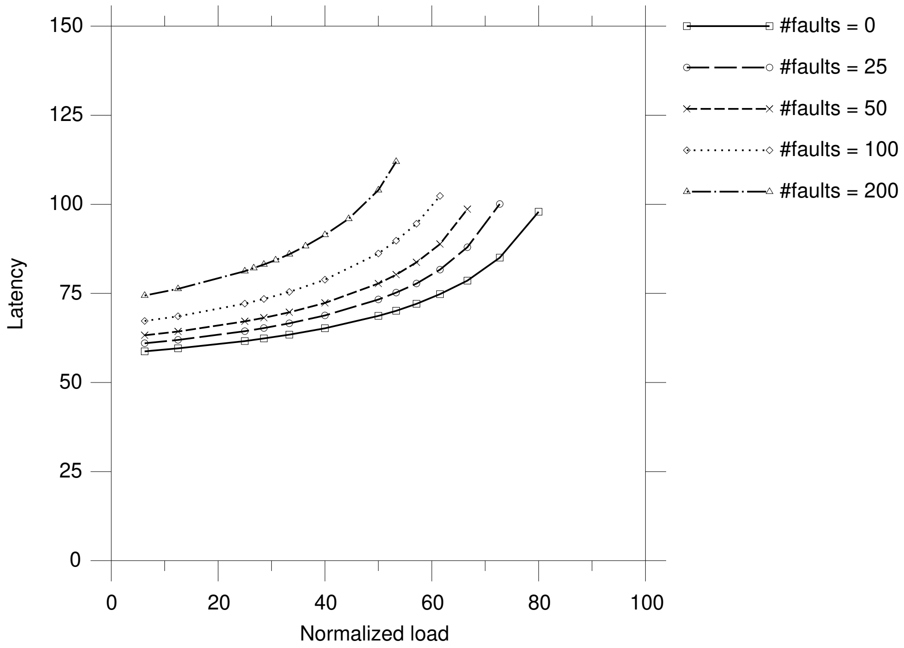

Figure 8.4: Latency versus message rate

throughput (on a per processor basis as a fraction of the capacity of a fault-free network) versus the number of faults. It also shows the throughput estimated as $1/\hat{L}$. We can see that $1/\hat{L}$ is a reasonable estimator of peak throughput and that the throughput degrades by about 20% with 100 faults and about 30% with 200 faults.

We also simulated the network under the open-network workload model. Figure 8.4 shows the results of the simulation. The message rate is shown as a fraction of the capacity of a fault-free network. We can see that as the number of faults increases, the average latency increases because more messages need two passes. As the number of faults increases we also see that the knee of the curve shifts to the left indicating that the peak throughput decreases. When there are 100 faults, there is approximately a 20% increase in latency and a 20% decrease in throughput.

Chapter 9

Conclusions

In this dissertation , we addressed the problem of designing efficient interconnection networks for large scale parallel machines. The main objective was to design networks which provided high performance for low cost.

We developed a model of packaging technology to characterize the network costs in the context of a multi-level packaging hierarchy. In order to model the general characteristics of a wide range of packaging technologies we defined a few generic packaging models that relied heavily on the property of packaging locality. The generic models based the computation of network cost on the usage of resources the higher levels of the hierarchy while ignoring the lower levels.

For each of the generic models, we evaluated different networks using detailed simulation. The simplest of our generic models used the number of top-level modules and the pin-count per module as metrics of network cost. Under this model, we found that the best interconnection network at the top level was a complete graph. A complete graph at the top level corresponded to a butterfly network between the processors, and we found that using wider channels at the lower levels of the packaging hierarchy helped in improving the utilization of the networks at the top level of the hierarchy.

In the second generic model, the computation of network cost was more complex, and included a third parameter, namely the number of top-level bundles. Under this model, we were unable to prove the optimality of any particular topology, but we were able to demonstrate that some classes of topologies were close to optimal. We evaluated in detail one such class of topologies called n-hop networks. This class of topologies has not received much attention in literature, even though they provide a good trade-off between pin-counts and number of bundles.

The third generic model dealt with costs at the top two levels of the hierarchy and illustrated an interesting trade-off between the number of bundles at the top level and the pin-counts at the next lower level. We found that by using a denser network at the lower level, we could reduce the number of bundles at the top level.

The results from the three generic models illustrate that although the top level network determines the performance of the network to a first approximation, we can get significant improvements in performance by making the lower level networks richer. We also found that, as a general rule, it was better to use fewer wider channels than many narrow channels. Wider channels were able to achieve higher utilization than narrow channels because more independent messages are needed to keep all the narrow channels busy. The results from the three models also illustrate that for shared memory based communication primitives, we can get better performance by using fewer memory units with wider ports to the network. This is due to a reason similar to the benefits provided by wider channels; it is easier to keep the memory units busy when there are fewer memory units with wider channels. This result suggests that it might be better to build networks out of small shared-memory multiprocessing units rather than single processor-memory pairs. Many of our network evaluations were based on multithreaded processors, and our results indicate that multithreading provides the processors with the ability to tolerate network latencies; latency tolerance translates to higher network performance.

We did not exhaustively compare all the network architectures at the lowest level of the packaging hierarchy, but we were able to demonstrate that some simple architectures achieved performance close to the upper bound. There may be other network architectures which have similar performance; for example it may be possible to build a 2D mesh at the lowest level, with sufficiently wide channels, so that it achieves performance close to the upper bound. When we consider the lowest level network in isolation, it is conceivable that many different networks achieve the same performance for different costs, and it may seem desirable to pick the one with the lowest cost. However, we believe that the cost of the lowest level network is only a small part of the overall network cost, and therefore, were ignored by our packaging models. If these costs did indeed matter, we could extend our model to include these costs, and analyze the cost-performance trade-offs at the lowest level[1].

[1] However, due to pin-count and bisection width considerations, similar to those considered in Chapter 4,

We formally defined scalability and divided networks into three broad categories based on the difficulty involved in scaling them. At the one end were networks in which the smaller networks were sub-portions of the larger networks. These networks can be easily scaled at the customer's site by simply adding on the additional parts required by a larger machine. At the other extreme were networks in which the larger networks had components that were different from the smaller ones. Scaling such networks essentially meant replacing the smaller network by a larger one. The other category of scalability represented networks that fell in between these two extremes. We evaluated the performance differences between the three categories of scalability and found that achieving greater scaling flexibility implied either an increase in cost or a reduction in performance. We provided a quantitative evaluation of the implications of designing networks with different levels of scaling flexibility.

We also presented a mechanism to tolerate faults in butterfly networks. The idea behind the fault-tolerance scheme was simple; any message that could not be sent directly was sent in two passes over the butterfly network. The difficult aspect of this fault-tolerance scheme, was the selection of paths for messages that were sent in two passes. These paths had to be chosen in a manner that minimized the performance degradation due to load imbalances in the network. In general, finding optimum paths for these messages was a difficult problem. We presented a few heuristics that selected paths in a manner that yielded a good balance in the load distribution in the network. We demonstrated that the network was able to provide good performance even in the presence of a large number of faults.

we expect that the 2D mesh is unlikely to be a contender for the network at the lowest level.

Bibliography

[A+90] R. Alverson et al. The TERA Computer System. In *1990 International Conference on Supercomputing*, pages 1–6, 1990.

[A+91] Anant Agarwal et al. The MIT Alewife Machine: A Large-Scale Distributed-Memory Multiprocessor. Technical Report MIT/LCS Memo TM-454, Massachusetts Institute of Technology, 1991.

[ACF90] Bowen Alpern, Larry Carter, and Ephraim Feig. Uniform Memory Hierarchies. In *Proceedings of the IEEE Annual Symposium on The Foundations of Computer Science*, pages 600–608, 1990.

[AG89] George S. Almasi and Allan Gottlieb. *Highly Parallel Computing.* The Benjamin/Cummings Publishing Company, 1989. Chapter 8.

[Aga91] Anant Agarwal. Limits on Interconnection Network Performance. *IEEE Transactions on Parallel and Distributed Systems*, 2(4):398–412, October 1991.

[AP90] S. Abraham and K. Padmanabhan. Constraint based evaluation of multicomputer networks. In *Proceedings of the International Conference on Parallel Processing*, 1990.

[Ath90] William C. Athas. Physically Compact, High-Performance Multicomputers. In *6th International MIT VLSI Conference*, pages 302–313, 1990.

[Bak90] H. B. Bakoglu. *Circuits, Interconnections, and Packaging for VLSI.* Addison-Wesley Publishing Company, 1990.

[BBDS92] David H. Bailey, Eric Barszcz, Leonardo Dagum, and Horst D. Simon. NAS Parallel Benchmark Results. Technical Report RNR-92-002, NASA Ames Research Center, Moffett Field, CA 94035, December 1992.

[BBN86] BBN Laboratories Inc. *Butterfly-TM Parallel Processor Overview*, 1986.

[BDQ86] J.-C. Bermond, C. Delorme, and J.-J. Quisquater. Strategies for Interconnection Networks: Some Methods from Graph Theory. *Journal of Parallel and Distributed Computing*, 3:433–449, 1986.

[Bel92] Gordon Bell. Ultracomputers: A Teraflop before its time. *Communications of the ACM*, 35(8):26–47, August 1992.

[BH86] Joshua E. Barnes and Piet Hut. A Hierarchical $O(n \log n)$ Force Calulation Algorithm. *Nature*, 324(4):446–449, 1986.

[BL84] Sandeep N. Bhatt and Frank Thomson Leighton. A Framework for Solving VLSI Graph Layout Problems. *Journal fo Computer and System Sciences*, 28(2):300–343, April 1984.

[Boo93] Robert Francis Boothe. *Evaluation of Multithreading and Caching in Large Shared Memory Parallel Computers*. PhD thesis, University of California, Berkeley, 1993. Computer Science Division, UCB//CSD–93–766.

[BP89] J.-C. Bermond and C. Peyrat. de Bruijn and Kautz Networks: A Competitor for the Hypercube? In F. André and J. P. Verjus, editors, *Hypercube and Distributed Computers*, pages 279–293. Elsevier Science Publishers B. V. (North Holland), 1989.

[CKP+92] David Culler, Richard Karp, David Patterson, Abhijit Sahay, Klaus Erik Schauser, Eunice Santos, Ramesh Subramoniam, and Thorsten von Eicken. LogP: Towards a Realistic Model of Parallel Computation. Technical Report UCB/CSD 92/713, University of California, Berkeley, 1992.

[Cyp90] R. Cypher. Theoretical aspects of VLSI pin limitations. In *6th International MIT VLSI Conference*, pages 314–327, 1990.

[Dal86] William Dally. *A VLSI Architecture for concurrent data structures*. PhD thesis, California Institute of Technology, 1986. Tech Report: 5209:TR:86.

[Dal87] W. J. Dally. Wire-Efficient VLSI Multiprocessor Communication Networks. In *Advanced Research in VLSI*, pages 391–415, 1987.

[Dan88] Sivarama Dandamudi. *Hierarchical Interconnection Networks for Multicomputer Systems*. PhD thesis, University of Saskatchewan, Canada, 1988. Department of Computational Science 88-18.

[DKN93] William J. Dally, John S. Keen, and Michael D. Noakes. The J-Machine Architecture and Evaluation. In *IEEE Computer Society International Computer Conference, COMPCON*, pages 183–188, 1993.

[Fen81] T. Feng. A survey of interconnection networks. *IEEE Computer*, 14(12):12–27, Dec 1981.

[FIR93] Steve J. Frank, Henry Burkhardt III, and James Rothnie. The KSR1: Bridging the Gap Between Shared Memory and MPPSs. In *IEEE Computer Society International Computer Conference, COMPCON*, pages 285–294, 1993.

[GGK+83] A. Gottlieb, R. Grishman, C. Kruskal, K. McAuliffe, L. Rudolph, and M. Snir. The NYU Ultracomputer - designing an MIMD shared memory parallel computer. *IEEE Transactions on Computers*, C-32:175–189, February 1983.

[GR87] Leslie Greengard and Vladimir Rokhlin. A Fast Algorithm for Particle Simulation. *Journal of Computational Physics*, 73(325), 1987.

[Gre90] Ronald I. Greenberg. The Fat-Pyramid: A Robust Network for Parallel Computation. In *6th International MIT VLSI Conference*, pages 195–213, 1990.

[H+86] J. P. Hayes et al. Architecture of a hypercube supercomputer. In *Proceedings of the International Conference on Parallel Processing*, pages 653–660, 1986.

[Hil85] W. Daniel Hillis. *The Connection Machine*. The MIT Press, 1985.

[Hil90] Mark D. Hill. What is Scalability? *Computer Architecture News*, 18(4):18–21, December 1990.

[IAS87] George B. Adams III, Dharma P. Agarwal, and Howard Jay Siegel. A Survey and Comparison of Fault-Tolerant Multistage Interconnection Networks. *IEEE Computer*, 20(6):14–27, June 1987.

[IEE] IEEE. IEEE Std 1596-1992 (Scalable Coherent Interface). Currently under development.

[Int86] Intel Corp. *Intel iPSC System Overview*, January 1986.

[Joe90] Christopher Frank Joerg. Design and implementation of a packet switched routing chip. Master's thesis, MIT, December 1990.

[Ken92] Kendall Square Research Corporation. *Kendall Square Research Technical Summary*, 1992.

[KK79] P. Kermani and L. Kleinrock. Virtual Cut-Through: A New Computer Communication Switching Technique. *Computer Networks*, 3:267–286, 1979.

[Kow85] Janusz S. Kowalik, editor. *Parallel MIMD computation : the HEP supercomputer and its applications*. The MIT Press, 1985.

[KS93] R. E. Kessler and J. L. Schwarzmeier. Cray T3D: A New Dimension for Cray Research. In *IEEE Computer Society International Computer Conference, COMPCON*, pages 176–182, 1993.

[KU91] Smaragda Konstantinidou and Eli Upfal. Experimental Comparison of Multistage Interconnection Networks. Technical report, IBM Research Division, Yorktown Heights, New York, 1991. Report No. RJ 8451 (76459).

[L+91] Daniel Lenoski et al. Overview and Status of the Stanford DASH Multiprocessor. In *International Symposium on Shared-Memory Multiprocessing*, pages 102–108, 1991.

[L+92] Charles E. Leiserson et al. The Network Architecture of the Connection Machine CM-5. In *Proc. of the 1992 ACM Symposium on Parallel Algorithms and Architectures*, pages 272–285, 1992.

[Lee89] Gyungho Lee. A performance bound of multistage combining networks. *IEEE Transactions on Computers*, C-38(10):1387–1395, October 1989.

[Lei82] Charles E. Leiserson. *Area-Efficient VLSI Computation*. PhD thesis, Carnegie-Mellon University, 1982. Also available as the ACM Doctoral Dissertation Award 1982, MIT Press.

[Lei84] F. T. Leighton. New Lower Bound Techniques for VLSI. *Mathematical Systems Theory*, 17(1):47–70, April 1984.

[Lei85] Charles E. Leiserson. Fat trees: Universal Networks for Hardware-Efficient Supercomputing. In *Proceedings of the International Conference on Parallel Processing*, pages 393–402, 1985.

[LKK86] Gyungho Lee, Clyde P. Kruskal, and David J. Kuck. The effectiveness of combining in shared memory parallel computers in the presence of 'hot spots'. In *Proceedings of the International Conference on Parallel Processing*, pages 35–41, 1986.

[LLM90] Tom Leighton, Derek Lisinski, and Bruce Maggs. Empirical evaluation of randomly-wired multistage networks. In *International Conference on Computer Design*, 1990.

[LM89] Tom Leighton and Bruce Maggs. Expanders might be practical: fast algorithms for routing around faults in multibutterflies. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 384–389, 1989.

[LMS92] T. Leighton, B. Maggs, and R. Sitaraman. On the Fault Tolerance of Some Popular Bounded-Degree Networks. Technical Report CS–TR–385–92, Department of Computer Science, Princeton University, Princeton, NJ, September 1992. Short version: Proceedings of the IEEE Annual Symposium on the Foundations of Computer Science, 1992, pp 542–552.

[LT88] Tom Lovett and Shreekant Thakkar. The Symmetry Multiprocessor System. In *Proceedings of the International Conference on Parallel Processing*, pages 303–310, 1988.

[McM92] R. J. McMillen. The Influence of Packaging Issues on the Design of the Massively Parallel NCR 3700 Computer. In *Workshop Report: Packaging, Interconnects and Optoelectornics for the Design of Parallel Computers Workshop*, 1992.

[MCS91] John M Mellor-Crummey and Michael L. Scott. Algorithms for Scalable Synchronization on Shared-Memory Multiprocessors. *ACM Transactions on Computer Systems*, 9(1):21–65, Feb 1991.

[Mey89] Ernest Meyer. Fine-pitch technology shrinks system size, boosts performance. *Computer Design*, pages 71–76, November 1989.

[NA91] Daniel Nussbaum and Anant Agarwal. Scalability of Parallel Machines. *Communications of the ACM*, 34(3):56–61, March 1991.

[ND90] Michael Noakes and William J. Dally. System Design of the J-Machine. In *6th International MIT VLSI Conference*, pages 179–194, 1990.

[Nef92] John A. Neff. Holographic Optical Interconnections in the 3-D Computer. In *Workshop Report: Packaging, Interconnects and Optoelectornics for the Design of Parallel Computers Workshop*, pages 61–65, 1992.

[Nic90] John R. Nickolls. The Design of MasPar MP-1: A Cost Effective Massively Parallel Computer. In *IEEE Computer Society International Computer Conference, COMPCON*, 1990.

[Nic92] John R. Nickolls. Interconnection Architecture and Packaging in Massively Parallel Computers. In *Workshop Report: Packaging, Interconnects and Optoelectornics for the Design of Parallel Computers Workshop*, pages 4–8, 1992.

[PBG+85] G. F. Pfister, W. C. Brantley, D. A. George, S. L. Harvey, W. J. Kleinfelder, K. P. McAuliffe E. A. Melton, V. A. Norton, and J. Weiss. The IBM Research Parallel Processor Prototype (RP3): Introduction and Architecture. In *Proceedings of the International Conference on Parallel Processing*, pages 764–771, 1985.

[PN85] G. F. Pfister and V. A. Norton. Hot-spot contention and combining in multistage interconnection networks. In *Proceedings of the International Conference on Parallel Processing*, pages 790–797, 1985.

[R+90] R. D. Rettberg et al. The Monarch Parallel Processor Hardware Design. *IEEE Computer*, pages 18–30, April 1990.

[Ran88] Abhiram G. Ranade. *Fluent Parallel Computation*. PhD thesis, Yale University, 1988. Department of Computer Science TR-663.

[RR90] M. T. Raghunath and Abhiram Ranade. A Simulation-Based Comparison of Interconnection Networks. In *Proceedings of the Second IEEE Symposium on Parallel and Distributed Processing*, pages 98–103. IEEE, December 1990.

[RR93] M. T. Raghunath and Abhiram Ranade. Designing Interconnection Networks For Multi-level Packaging. In *Supercomputing '93, Portland, Oregon*, November 1993. To appear.

[SBN82] Daniel P. Siewiorek, C. Gordon Bell, and Allen Newell. *Computer Structures: Principles and Examples.* McGraw Hill International Book Company, 1982. Chapter 41.

[TG92] D. Z. Tsang and T. J. Goblick. Free Space Optical Interconnections for Parallel Computers. In *Workshop Report: Packaging, Interconnects and Optoelectornics for the Design of Parallel Computers Workshop*, pages 56–60, 1992.

[Tho80] C. D. Thompson. *A Complexity Theory for VLSI.* PhD thesis, Carnegie-Mellon University, 1980.

[TMC91] TMC. CM5 Reference Manual. Technical report, Thinking Machines Inc, 1991.

[Ull84] Jeffrey D Ullman. *Computational aspects of VLSI.* Computer Science Press, 1984.

[Upf89] Eli Upfal. An O(log N) deterministic packet routing scheme. In *Proceedings of the ACM Annual Symposium on Theory of Computing*, pages 241–250, 1989.

[VR89] Anujan Varma and C. S. Raghavendra. Fault-Tolerant Routing in Multistage Interconnection Networks. *IEEE Transactions on Computers*, 38(3):385–393, March 1989.

[War92] Steve Ward. Toward Legoflops: A Scalable 3D Interconnect. In *Workshop Report: Packaging, Interconnects and Optoelectornics for the Design of Parallel Computers Workshop*, pages 9–14, 1992.

[WF80] C. Wu and T. Feng. On a class of multistage interconnection networks. *IEEE Transactions on Computers*, C-29:694–702, August 1980.

[YTL86] P. Yew, N. Tzeng, and D. H. Lawrie. Distributing hot-spot addressing in large-scale multiprocessors. In *Proceedings of the International Conference on Parallel Processing*, pages 51–58, 1986.