January 1987

Report No. STAN-CS-87-1144
Also numbered KSL-87-07

Considerations for Multiprocessor Topologies

by

Gregory F. Byrd and Bruce A. Delagi

Department of Computer Science

Stanford University
Stanford, CA 94305

Considerations for Multiprocessor Topologies*

Greg Byrd†
Knowledge Systems Laboratory
Stanford University
Stanford, CA 94305

Bruce Delagi
Worksystems Engineering Group
Digit al Equipment Corporation
Maynard, MA 01754

Abstract

Choosing a multiprocessor interconnection topology may depend on high-level considerations, such as the intended application domain and the expected number of processors. It certainly depends on low-level implementation details, such as packaging and communications protocols. We first use rough measures of cost and performance to characterize several topologies. We then examine how implementation details can affect the realizable performance of a topology.

1 Introduction-Design Constraints and Opportunities

The base for development of general purpose multiprocessor systems as for computer systems today generally is given by the design constraints and opportunities established by evolving semiconductor design and manufacturing processes. The VLSI design medium brings a new perspective on cost: switches are cheap; wires are expensive. In modern microprocessors, communication costs dominate those associated with logic. Power and cooling budgets are spent driving wires and overwhelmingly, chip area is dedicated to wiring rather than logic [17]. To an increasing degree, the dominant delays are associated with driving lines rather than the accomplishment of logic functions per se. One implication is that, all other things being equal, smaller, simpler processors can be expected to have shorter operation cycles than larger, more complex designs [18]. They are also likely to be available in a more recent, higher performance base technology.

*This work was supported by DARPA Contract F30602-85-C-0012, NASA Ames Contract NCC 2-220-S1, and Boeing Contract W266875.

†Supported by an NSF Graduate Fellowship and by the Stanford Dept. of Electrical Engineering.

At the system level, the consequence of relatively expensive communication is that performance is enhanced if the design establishes that whenever a lot of information has to move in a short time, it does not have to move far. Significant locality of high bandwidth links is a goal. Among the highest bandwidth links in a computer system is that connecting the processor and memory. Early computer systems separated these pieces and put a bottleneck between them to accommodate the packaging realities of the time: processors were implemented with electronic means, memory with magnetic, and their power requirements and EMI characteristics were best dealt with separately. There are new realities now: close coupling of processors with local memory is preferred.

With these design constraints in mind, we consider a multicomputer implementation based on a set of processor/memory pairs connected by a communications topology. Many topologies have been proposed [8] and have been compared in terms of theoretical cost and performance measures [16]. We argue, however, that the realizable performance of these topologies are closely linked to details of system packaging.

2 Interprocessor Connection Topologies

Connection schemes between processing sites can be compared with respect to their cost and performance as a function of the number of sites connected. For a particular connection scheme, if the cost grows no faster than the number of sites and the performance grows at least as fast, that scheme can be described as scalable. A rough measure of cost is the number of input-output ports required for connection. A rough measure of performance is the number of links in the topology divided by the largest number of links that must be traversed, and thus occupied to accomplish a transmission, in order to get from one node in the

network to another. This indication of the bound on the number of independent, concurrent transmissions we will call the concurrency of the network.

For some topologies, the concurrency of a network may *understate* performance as actually experienced in a given application: to the extent that there is locality of reference in transmissions, the number of links actually traversed may be better approximated by a constant than some function of the number of connected sites. Network concurrency may also over-&ale performance of one topology with respect to another: to the extent that the time to traverse links is not the same for all topologies, those that have non-uniform link costs (perhaps due to physical distance considerations applied to the realized lengths of links) will deliver less performance than the concurrency measure suggests. This is because in these cases, logical adjacency due to high dimensionality is merely apparent-embedding the topology in the dimensionality of space available tends to incur just those expenses related to physical distances that the topology was expected to eliminate.

2.1 Topologies With Scalable Concurrency

Several topologies are shown in Table 1 which have scalable concurrency. As the number of sites is increased, the network grows enough to support the consequential additional traffic. In fact, by this measure of performance, the last three of these four topologies scale performance equally well. However, as will be described, there are other considerations to weigh.

In the crossbar and completely connected topologies, the number of ports, a first approximation to cost, grows quadratically with the number of nodes in the network. Weighing cost and concurrency, then, we might prefer the banyan and boolean k-cube (also known as "hypercube") topologies.

By these measures, there does not seem to be a clear-cut choice between the banyan and the hypercube. A more sophisticated measure of cost would take into-account the area required for laying out the topology in a plane [11]. The banyan may have a slight edge in this category', but both layouts require relatively long wires, which is undesirable if link transit time dominates switching time.[2]

A major difference between the two topologies is that switching and routing are centralized at the processor in the hypercube, whereas the switching in the banyan is distributed throughout the network. To the extent that storage is required at the switch (as in [3]), it becomes more economical to centralize the switch and utilize the local storage of the processor. For this reason, we prefer the hypercube.

2.2 Topologies With Scalable Cost

There are alternative topologies not as richly connected as those just considered. The topologies in Table 2 all have fixed degree connectivity, so they all have scalable cost as measured by port count. Unfortunately, none of them has scalable concurrency. So, at least among the ten representative topologies discussed, there is no topology that has cost-performance characteristics intrinsically superior to all the others.

Concurrency for the ring and the bus topologies does not increase at all as the number of processors increases. Given no guarantee of transmission source to target locality, these seem unsuitable for systems with a large number of processors (e.g., > 100).

The perfect shuffle and cube-connected cycles (CCC) topologies emulate the O(log n) latency of the hypercube, but the number of links is linear with the number of processors, so concurrency does not scale. Also, if we measure cost in terms of layout area, the cost of the perfect shuffle ($0(\frac{n^2}{\log^{3/2} n})$) and CCC ($O(\frac{n^2}{\log^{3/2} n})$) [15] do not scale and so will not be considered further.

The tree, grid, and torus topologies all have fixed degree connectivity and have the optimum O(n) area requirement.The tree has a slightly better *capacity measure and a lower latency bound. Note, however, that the tree provides no alternate communication paths (useful in network balancing and defect tolerance) and has a bottlenecking root.[3] Connections might be added to provide alternate paths, but, as we-will see in the next section, physical link considerations may make the grid or torus a better choice.

The area required to lay out a hypercube in a plane is $0(n^2)$ [2], where n is the number of processors. Since "banyan" actually denotes a class of interconnections it is difficult to make a general statement about its layout. However, let us consider a particular banyan network, the omega network [10], which is logn stages of perfect shuffle connections. The perfect shuffle has area $0(\frac{n^2}{\log^{3/2} n})$ [15], so we would expect log n perfect shuffles to require area $0(\frac{n^2}{\sqrt{\log n}})$, which is a slightly better bound than for the hypercube. Other types of banyans, with different fan-in, fan-out, and connectivity characteristics might have even smaller bounds.

[2] See Section 3.

[3] We might be able to deal with this by increasing the bandwidth of the links as we proceed toward the root, for example with "fat trees" [12].

3 Link Costs-Examining The Free Lunch

Most studies of topologies assume a constant cost for link traversals as the number of links increases. This is a useful approximation if the time to drive and receive link signals is constant with link length and large compared to signal transit time on the link. However, this is increasingly not a good assumption both as the underlying feature size of the component technology decreases and as we consider larger numbers of sites in a system. Given a fixed circuit feature size, topologies with scalable concurrency, as discussed in Section 2.1 suffer increased link lengths and thus longer signal transit times-with possibly increasing drive times—as the number of processors increases. Alternatively, given a fixed volume of circuits in these topologies and decreasing circuit feature size, the number of processors in the system increases but so does the ratio between link lengths and feature size. Thus relative to the circuit delay times which are dependent on (and decrease with) circuit feature size, the link transit times become increasingly a more important consideration.[4]

Topology has to be viewed as a dependent variable determined principally by the packaging technology of the system. As an example, consider the recursive-H layout for the binary tree (Figure 1) under the assumption that link transit time dominates switching time. Now consider the grid in Figure 2, which can be laid out in the same area. If transit times dominate, then shorter links and more switching sites will likely shorten the point-to-point communications cycle time and improve the realized capacity of the network.[5] Furthermore, additional data paths allow dynamic routing of messages, and additional computing resources make the grid potentially more powerful than the tree.

Though the torus appears to suffer from extremely long wires which "wrap around" the edges, a simple renumbering of the processors in a grid brings each one within two hops of its logical neighbors[6] (see Figure 3). Thus, we can effectively create a torus by changing the routing algorithm of a grid. Alternatively, we could keep the original torus connections and lay out the processors as in Figure 3(b), resulting in links which are at most twice as long as those for a grid. In the remainder of the paper, we will speak of the grid bearing in mind construction of the torus in these terms.

4 A Packaging Example

We are now faced with two topologies: one with scalable performance—the hypercube-and one with scalable cost-the grid. The arguments presented above suggest that, all else being equal, the communication cycle time for the hypercube would be greater than that of the grid, due to its long links. Even so, the average message latency of the hypercube may still be smaller, due to its high connectivity. To get a better understanding of the relative performance of the two systems, we should examine how they might actually be implemented in near-future technology.

In the mid-1990's we would expect a 0.5-μm MOS fabrication process to be available [7]. We will assume that the complexity of our processor is comparable to today's typical 32-bit microprocessor. The MicroVAX 78032 chip [4], for example, is implemented in 3-μm technology; it measures about 8.5 mm on a side. Using 0.5-μm technology, we could expect a similar processor to require around 1.5 mm on a side. Let us allow 256K bytes (2M bits) of local memory for our processor. Fujitsu's megabit RAM using 1.4-pm technology takes 54.7 mm^2 [6]. If the dimensions of the Fujitsu chip are about 10 mm by 5.5 mm, then a 0.5-μm version would be 3.6 mm by 2.0 mm. Two of these (since we want 2M bits) would be around 3.6 mm by 4 mm. As an approximation, then, each processing element, including a processor, 256K bytes of local memory, and switching and routing circuitry could be expected to fit onto a 5 mm x 5 mm piece of silicon.

Even as devices shrink, die sizes continue to grow. By the mid-90's, the state-of-the-art chips may be as large as 15 mm on a side. Each chip would be expected to have 400-600 I/O pads [14]. Therefore,

[4] The dependence of communication delays on signalling lengths as circuit feature size decreases depends on assumptions made on the thickness and thus the resistivity of associated interconnects. Uniform scaling leads to relative signalling times that increase quadratically with distance [19]. Detailed analysis of the equations of voltage and current in VLSI wire implementations (including consideration of the non-linear characteristics of signal drivers) demonstrated linear dependences [1] but were done assuming that the interconnect (and field oxide) thicknesses did not decrease at all while all other dimensions scaled with the circuit feature size of the technology [17]. Another approach imagines a hierarchy of interconnect of increasing thicknesses with distance [13] to achieve signalling times that grow only with the logarithm of the distance. Yet another approach accepts resistive links but given control over both minimum and maximum wire lengths and use of high impedance receivers, notes that it is possible to counter dispersive losses with reflective voltage doubling at the receiving end of a point to point link [9].

[5] The assumption made here is that the message routing is relatively independent of the computing activities at a processing site, so there is no penalty associated with being routed at a processing site rather than a switch.

[6] This approach is attributed to R. Zippel.

we could put up to nine processing sites on a single die.

The dice could be flip-mounted on a silicon [5] or ceramic [9] substrate with thin-film transmission lines and integrated capacitors. In [9], the maximum length for 5-μm-thick lines is around 20 cm, so we will assume a 10x10 cm module size, on which we can easily place up to 36 dice. We will assume on the order of 1000 I/O pins per module [5].

Consider first packaging a (32x32) 1024-element octal grid, in which each processor is connected to eight neighbors. With nine processors (arranged as a 3x3 grid) on a die, 32 (bi-directional) communication links must come off the chip through the I/O pads, so no more than 18 pads could be used per channel. A module can carry 324 processors, arranged as an 18x18 grid. The entire system, then, could fit on four modules (with room to spare). The communications links from two sides of the 18x18 grid (105 bidirectional channels) must go off-module. Thus, each channel could use 10 pins-one pin for clock and status information and four for data, in each direction.

Now consider a 1024-element hypercube (a "10-cube"). To allow for more complex wiring and easier packaging, we will assume that each die contains eight processors, and each module will hold 32 dice, for a total of 256 processors per module. (Extra space might be used to provide redundant processors for fault tolerance.) Again, only four modules are required to package all 1024 processors. Each processor has ten bidirectional links to its logical neighbors. If the eight processors on a die are wired as a 3-cube, then seven channels from each processor must go off-chip. Five of these channels are connected to other processors on the same module, but two must go off the module. With only $\sim$ 1000 I/O pins for 512 bidirectional channels, it appears that a l-bit combined control/data stream is all that can be supported for the hypercube communications. If we decrease the number of processors per die to four (and possibly add more memory), we can use separate wires for control and data but the wires will be longer.

Note that in both cases the module pin-out is the limiting factor for channel width, rather than the chip pin-out. If more off-module I/O pins are available, things will look better, but there will still be around a 5-to-1 ratio of the number of required off-module channels in the hypercube as compared to the grid. As mentioned before, the average interconnect length for the grid will be much shorter than that for the hypercube. Therefore, the grid offers shorter (i.e., faster) and wider communication paths than the hypercube when implemented in projected near-future technology.

5 Beyond Topology

As the previous example indicates, the electrical and physical characteristics of the circuit packaging in a system may dictate the scheme used to wire the nodes together. In addition, the communications protocol, that is, the actual signalling on the links are an important component of achievable performance. There are many relevant details-for example:

- Dynamic routing, selecting available links as needed, is useful in balancing load and thus allows more of communication resources of the system to be well used throughout a computation.

- Cut-through routing, making a routing decision on the fly as a packet is received, reduces buffer requirements in the system and minimizes latency experienced in network transit.

- Local flow control, signalling transmission delays back to the source based on local blockage information, together with single "word" buffering and transmission validation at each network input and output port allows the source to complete a validated transmission in a time that does not depend on the size of the network.

- Point to point multicast, sending (approximately) the same packet to multiple targets using common resources to the largest degree possible-coupled with dynamic, cut-through routing, flow control, and word level buffering and transmission validation-provides "virtual busses" precisely as and when they are needed.

A point-to-point protocol utilizing these mechanisms is described in [3].

6 Conclusion

Communications performance of practical systems depends first of all on available packaging technology and second on protocol considerations. No topology considered here has both scalable cost and performance, so the topology chosen must be in the context of the number of processors targetted. For a thousand processors or so, given the assumptions on mid-1990's technology discussed earlier, the grid (or torus) seems an appropriate choice. The performance of the grid will depend on the signalling protocol and will be best predicted through application simulations detailed enough to relect design decisions made at that level.

References

[1] G. Bilardi, M. Pracchi, and F. P. Preparata. A critique and an appraisal of VLSI models of computation. In H. T. Kung, B. Sproul, and G. Steele, editors, *VLSI Systems and Computations*, pages 81-88, Computer Science Press, Inc., Rockville, MD, 1981.

[2] G. Brebner. Relating routing graphs and two-dimensional grids. In P. Bertolazzi and F. Luccio, editors, *VLSI: Algorithms and Architectures*, pages 221-231, Elsevier Science Publishers B.V., Amsterdam, 1985.

[3] G. T. Byrd, R. Nakano, and B. A. Delagi. *A Point-to-point Muticast Communications Protocol*. Technical Report KSL-87-02, Knowledge Systems Laboratory, Stanford University, January 1987.

[4] D. W. Dobberpuhl, R. M. Supnik, and R. T. Witek. The MicroVAX 78032 chip, a 32-bit microprocessor. *Digital Technical Journal*, (2):12–23, March 1986.

[5] Capt. B. J. Donlan, J. F. McDonald, R. H. Steinvorth, M. K. Dodhi, G. F. Taylor, and A. S. Bergendahl. The wafer transmission module. *VLSI Systems Design*, 7(1):54–58, 88-90, January 1986.

[6] Electronic News, July 1,1985.

[7] C. K. Lau, et. al. A high performance half-micron gate CMOS process for VLSI. In *Proceedings of the 1985 International Conference on Computer Design: VLSI in Computers*, IEEE, October 1985.

[8] T. Feng. A survey of interconnection networks. *Computer*, 12-27, December 1981.

[9] C. W. Ho, D. A. Chance, C. H. Bajorek, and R. E. Acosta. The thin-film module as a high--performance semiconductor package. *IBM Journal of Research and Development, 26(3):286-*296, May 1982.

[10] D. H. Lawrie. Access and alignment of data in an array processor. *IEEE Transactions on Computers*, C-24(12):1145-l 155, December 1975.

[11] C. E. Leiserson. *Area-Efficient Graph Layouts (for VLSI)*. Technical Report CMU-CS-80-138, Carnegie-Mellon University, August 1980.

[12] C. E. Leiserson. Fat-trees: univeral networks for hardware-efficient supercomputing. In *Proceedings of the 1985 International Conference on Parallel Processing*, pages 393-402, IEEE, 1985.

[13] C. Mead and M. Rem. Minimum propagation delays in VLSI. In *Caltech Conference on VLSI*, pages 433-439, January 1981.

[14] D. Nelsen. Personal Communication.

[15] F. P. Preparata and J. Vuillemin. The cube-connected cycles: a versatile network for parallel computation. *Communications of the ACM*, 24(5):300–309, May 1981.

[16] D. A. Reed and H. D. Schwetman. Cost-Performance bounds for multimicrocomputer networks. *IEEE Transactions on Computers*, C-32(1):83–95, January 1983.

[17] C. L. Seitz. Ensemble architectures for VLSI—a survey and taxonomy. In *1982 Conference* on *Advanced Research* in *VLSI*, MIT, January 1982.

[18] C. L. Seitz. Experiments with VLSI ensemble machines. *Journal of VLSI and Computer* Science, 1(3), 1984.

[19] C. L. Seitz. Self-timed VLSI systems. In *Caltech Conference on VLSI*, pages 345-355, January 1979.

Topology	Number of Ports	Longest Path	Concurrency
Completely connected	$O(n^2)$	$O(1)$	$O(n^2)$
Crossbar	$O(n^2)$[a]	$O(1)$	$O(n)$
Banyan	$O(n \log n)$	$O(\log n)$	$O(n)$
Boolean k-cube ($n = 2$)	$O(n \log n)$	$O(\log n)$	$O(n)$

[a] The number of links is O(n).

Table 1: Scalable Concurrency Topologies. [n = # processors]

Topology	Sumber of Ports	Longest Path	Concurrency	Area
Ring	$O(n)$	$O(n)$	$O(1)$	$O(n)$
Global bus	$O(n)$	$O(1)$	$O(1)$	$O(n)$
Perfect shuffle	$O(n)$	$O(\log n)$	$O(\frac{n}{\log n})$	$O(\frac{n^2}{\log^2 n})$
Cube-connected cycles	$O(n)$	$O(\log n)$	$O(\frac{n}{\log n})$	$O(\frac{n^2}{\log^2 n})$
Binary tree	$O(n)$	$O(\log n)$	$O(\frac{n}{\log n})$	$O(n)$
Grid/Torus	$O(n)$	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(n)$

Table 2: Scalable Cost Topologies. [n = # processors]

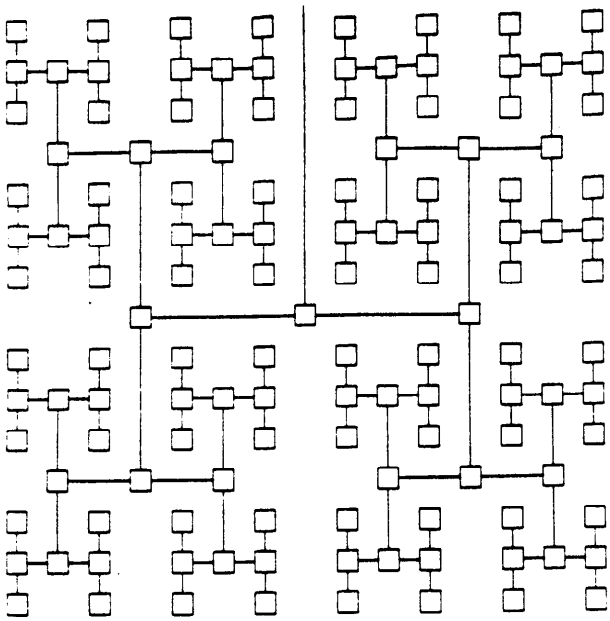


Figure 1: Recursive-H binary tree.

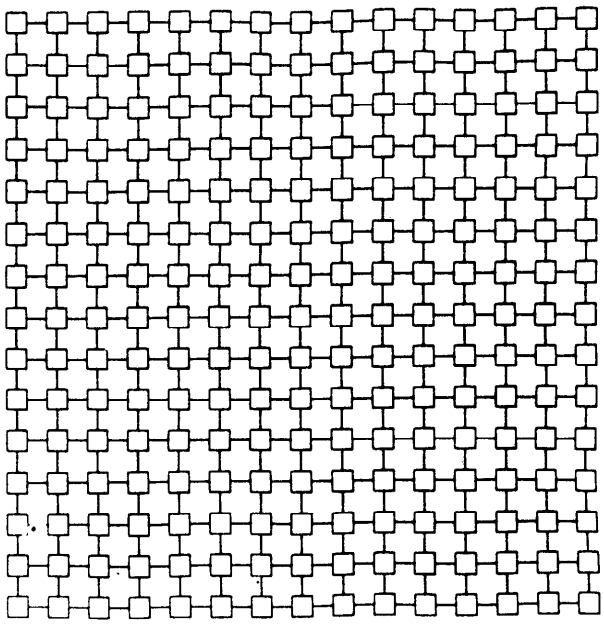
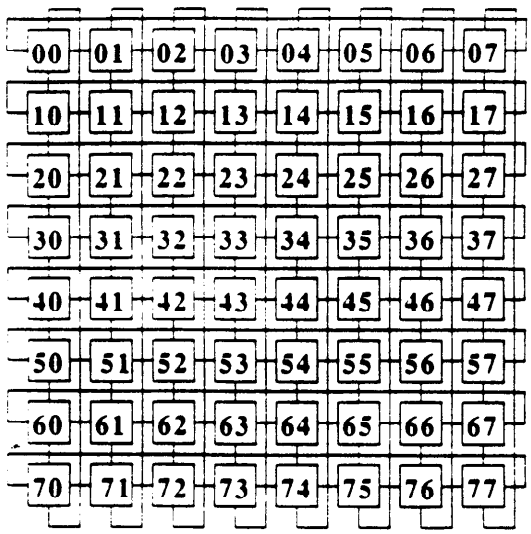
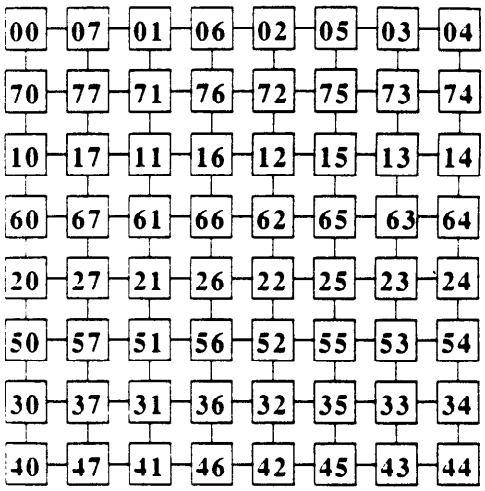


Figure 2: Two-dimensional grid.



(a)



(b)

Figure 3: Torus (a) and renumbered grid (b).