

Lecture #8 and 9: Interconnection Networks

*Professor: S. Lennart Johnsson**TA: Wei Ding*

1 Networks

A bus based parallel architectures is practical only for a small number of processors. For several thousand processors the bus would need to have a width of tens of thousands, or possibly a few hundred thousand wires (bits). Similarly, it is not possible to provide complete interconnectivity for highly parallel systems. For $N = 10,000$, 100,000,000 channels would be required. Not only is the total number of channels unrealistic, but so is also the number of channels per node. One of the major constraints in system design is the packaging. The number of connections between a chip and a printed circuit (PC) board, or between a PC board and a backplane, or between backplanes, are limited by ensuring a sufficient mechanical strength of the connectors, enforcing a minimum width of each connector. Pin Grid Arrays (PGA) offer 200 – 400 pins per chip package, while current state-of-the art so-called Land Grid Arrays (LGA) offers up to 2,000 connectors with spacings at about 1 mm, requiring an area of close to 2,500 mm². Board connectors may offer about 2 – 3 pins per mm, or about 1,000 for a standard PC board. Thus, all highly parallel systems use some form of sparse network to interconnect the processors and memory modules.

Locality of reference is important for performance at all levels in most systems. With respect to memory chip technology accessing elements within the same DRAM page offers a significantly faster access time than if data is scattered over different pages. Reducing the demands on memory bandwidth by reusing register contents, and the content of intermediate fast memory in the form of cache is important as well. Limitations on the data transfer rates are imposed not only by the fundamental characteristics of chip technologies, but also by packaging technologies. In the following we will assume that memory is distributed among the processing nodes such that if there is locality of reference and it is properly exploited through data mapping and scheduling of operations, then the demands on the communication system is reduced. The architectural model is shown in Figure 1.

Though it is true in most of the current generation of distributed memory architectures that a processor is involved in the data transfer to or from the memory associated with it, there are also efforts under way to allow for remote direct memory accesses which conceptually results in a distributed memory systems architecture as shown in Figure 2. Separate communications processors (not shown) handles remote requests for memory accesses. In the BSP, the network clearly was an integral part of the memory system, since it had to provide the required alignment for data accesses. But, conceptually, the BSP was a physically shared memory system rather than a distributed memory system. The alignment network had to support the full bandwidth of the memory system. Distributed memory systems are usually designed with the assumption that there is a certain amount of locality of reference, and that the network need not support

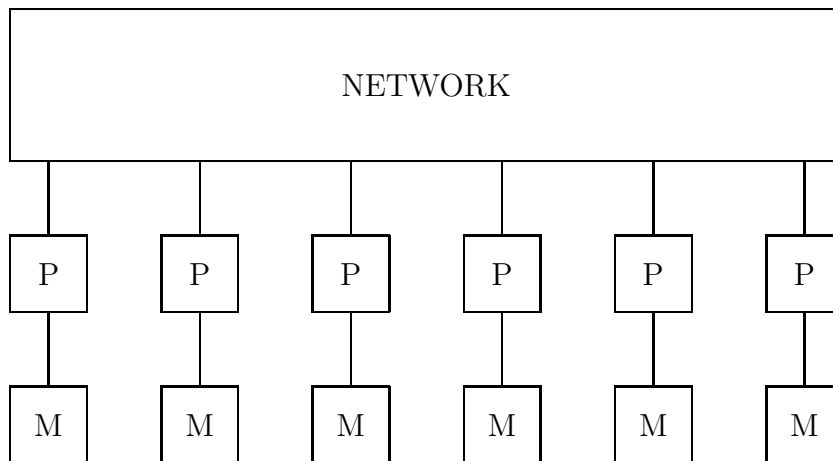


Figure 1: A generic model of distributed memory architectures.

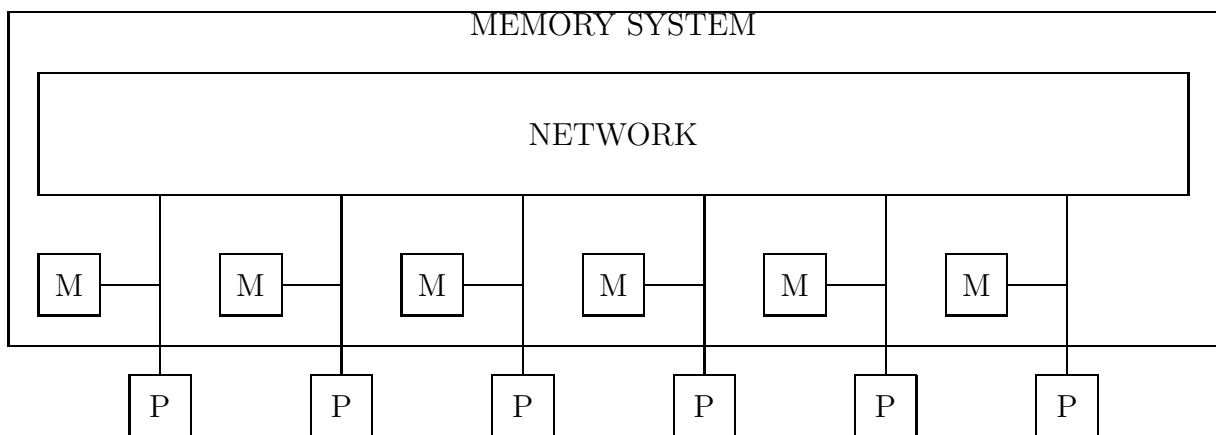


Figure 2: The memory system for distributed memory architectures.

the aggregate bandwidth of the memory modules. Thus, when there is no locality of reference, the performance is typically limited by the network, either by the bandwidth of the ports to the local memory, or by contention in the network.

Note that even in a memory systems as shown in Figure 2, the concurrency in communication and computation is limited. The single line into each memory unit symbolizes a single port to memory. Each local memory system can support either the local processor or remote accesses. However, the architecture shown in Figure 2 allows the overhead associated with remote requests to be overlapped with local execution. The overhead with remote access consists in address computations, which may be quite substantial since it involves a translation between local and global address spaces, creating packets to be transmitted, executing the selected communication protocol and providing queuing and possibly also routing services. In most of today's systems, the overhead far exceeds the latency in the network. The ratio is typically about a factor of

10 for simple, fairly low level protocols, about a factor of 100 for typical message passing calls known as “send” or “receive”, or about a factor of a 1,000 for some high level protocols.

Selecting a network implies a tradeoff between cost and utility. Traditionally area or volume is a good indicator of the cost of a network. The area/volume is largely determined by the wiring needs. At the lowest level, the chip level, the area is typically entirely determined by the wiring needs. The logic can be made to fit “under” the wires, which is apparent from looking at a microphotograph of almost any modern chip. Similarly, the PC board area is largely determined by the wires making up the different layers of the board. Thus, in estimating the cost of a network we will rely on a simple model for wire placement.

Another characteristic related to cost is the number of channels per node, since the size of a packet may entirely be determined by the area or perimeter required to fit all the connectors (pins). Supporting concurrent communication on all the channels of a node requires many independent data paths in a node. If the network provides several edge-disjoint paths between pairs of nodes it is desirable to make effective use of this property. We will discuss this issue further in the context of routing in networks.

Modularity is also important with respect to cost; can the network be partitioned into identical modules, in particular modules that would be the same regardless of the system size being constructed? The latter property would allow for the use of mass produced parts. Moreover, if the parts have no dependence on size, then the parts place no upper limit on the size system that can be constructed on one hand, and on the other hand, small systems do not exhibit any penalty for the ability to build large systems of the parts.

Measuring the utility of the networks is a quite difficult undertaking. If a limited set of well defined tasks should be performed, then it may be possible to find optimum data allocations for those tasks and associated algorithms. In determining the optimum allocation it is also necessary to determine how data associated with remote references should be routed through the network in order to minimize the communication time. Today, the utility of networks is largely determined by defining a set of data reference patterns in the index space of the computations, then finding good, possibly optimum, data allocations and associated routing schemes for the networks being considered. This approach is useful when the data reference patterns are sufficiently simple that they can be easily parameterized and analyzed. For more complex data reference patterns, a more conservative approach is often taken. Instead of striving for the best possible performance for fairly regular data accesses, such as nearest neighbor references on a grid on two or three dimensions, an approach guaranteeing an acceptable worst case behavior may be followed. We will discuss both approaches in the following.

Determining the utility of a network by determining how well it can perform a certain computation is often referred to as emulation. For instance, for computations for which the dominating data reference pattern is nearest neighbor communication on a grid in one or several dimensions, it is of interest to determine how well such a grid can be emulated on the network being considered. Some of the grid edges may be mapped to network nodes far apart, which may introduce high latencies and contention for network channels and nodes. As we will see later, it is important to consider both low and high load in the network. A high load is typically associated with nodes with a substantial memory. In such a case, many data elements may

need to be sent between a pair of nodes, or gathered or scattered by a node. For instance, in a node with 32 Mbytes of memory, a $64 \times 64 \times 64$ subgrid with four variables in 64-bit precision would occupy 25% of the memory. The number of variables to be moved between a pair of nodes for a shift operation with shift distance of one is $64 \times 64 \times 64 = 16384$, or a total of 128 kbytes.

For light load, latency tends to be the factor determining the performance, while for high load the bandwidth provided by the nodes or the network are most important. Topological properties of the networks are often used to aid in determining the (best possible) performance. Below, we define the most important ones.

- **Network Diameter**

This is the maximum distance between any pair of nodes in the network. The distance between a pair of nodes is the smallest number of wires that have to be traversed to get from one node to the other. A small diameter is desirable because it is the lower bound for worst-case node-to-node communication time.

- **Network Bisection Width**

This is the minimum number of wires that have to be removed to break the network into two halves, with identical (within one) numbers of nodes. A large bisection width is desirable because it is the minimum bandwidth available between two halves of the network.

- **Maximum Edge Length.** The length of the longest edge interconnecting a pair of nodes when laid out in a plane or a three-dimensional volume.

- **Network Area or Volume.** The area or volume consumed by the network when laid out in a plane or a three-dimensional volume.

- **Edge or channel width.** The width in bits of each edge connecting a pair of nodes.

The maximum edge length is of importance with respect to how fast the network can be operated in a synchronous mode. In a self-timed system it establishes a lower bound on performance.

Finally we mention the concept of a *universal* network, which is a network that, in a given volume, can simulate any other network with a slowdown that in the worst case is proportional to the logarithm of the volume it occupies.

Below, we present a number of common interconnection networks. For some of the networks, it is often helpful to view the nodes as being factored into rows and columns, possibly in several dimensions. Then, the total number of nodes is factored as $N = N_0 \times N_1 \times N_2 \times \dots \times N_{d-1}$, where d is the number of axes in the array representation of the nodes.

2 The cost of a network

In order to get a basis for the cost of a network we will use a simple, but formal model for the layout of networks in the plane. A similar model can be defined for three dimensions. The

formal model is often referred to as the Thompson Grid point model.

In VLSI (Very Large Scale Integration) technology used for chip manufacturing [7], layouts are usually wire limited, i.e., the area is determined by the wiring requirements. The logic, in the form of transistors, can fit under the wires without appreciably increasing the chip area. The so-called Thompson grid model [11] is a simplified model for device and circuit layout aimed at capturing the area requirements for different networks.

The Thompson grid point model is as follows:

- The layout medium has two layers for wires connecting nodes. One layer is used for vertical tracks, one layer for horizontal tracks.
- Nodes, like transistors or processors in a macroscopic model, are placed at intersections between wires in vertical and horizontal tracks. Nodes are represented as points. Wires on different layers are connected through contact cuts.
- A track can only be used for a single wire at a time, i.e., the width of a track only holds one wire.
- Tracks are spaced according to the minimum pitch required by the technology.

In VLSI technology there are usually more than two layers. In generic MOS technology, as described in [7], there is three layers: metal, poly, and diffusion. On printed circuit boards there are often many layers, say 10 or more. The grid point model results in what is often called Manhattan geometry, because of the regular vertical and horizontal track model. In Boston geometry, wires are allowed to take any path obeying minimum width and minimum separation rules. However, such geometries may be considerably more expensive to use for chip manufacturing, since for chips each path is broken down into a sequence of rectangles. The exposure time per rectangle is independent of its shape.

Thus, Manhattan geometry is dominating chip manufacturing. And even though more than two layers may be possible, the number of layers is very limited compared to the number of nodes or tracks. The additional layers are not expected to significantly change the results obtained by using the simple grid point model by more than a small constant factor.

The area of a chip has a very strong influence on its cost. Chips are produced by photo lithography, with chips placed on a wafer, which today is 12 - 15 cm in diameter on state of the art fabrication lines. Thus, the bigger the chip the fewer the number of chips per wafer. The wafer fabrication cost is independent of what is on it.

Moreover, the manufacturing process is not perfect, so some chips will have defects. The likelihood for defects grows exponentially in the area. Hence, the cost of a working chip grows very rapidly as a function of its size, see section 2.3 [3]. Doubling the die area may increase the cost per die five-fold!

One concern in chip manufacturing are contact cuts, which both adds area, and tend to reduce the yield. Minimizing the number of contact cuts is important in chip layout. However, we will not pursue this issue.

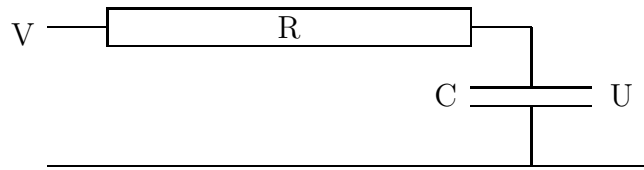


Figure 3: Circuit model for charging a gate in MOS technology.

Another important factor with respect to performance is the wire length, in particular the maximum wire length since it may determine the clock frequency.

MOS (Metal Oxide Semiconductor) VLSI technology is essentially a charge transfer technology. In silicon based MOS technologies, a transistor is formed by crossing a wire in *polychrystalline* silicon with another wire in diffused silicon, or *diffusion* for short. In addition, wires in aluminum, known as *metal*, is used to carry both signals, clock, and power. Aluminum is a much better conductor than either *poly* or *diffusion*.

The gate of a transistor formed by the crossing of poly and diffusion acts as a capacitor. When charged, the transistor is open, allowing current to flow in diffusion through the gate. When the capacitor is drained, then the transistor is turned off, and the gate is closed. No current can flow passed the gate in the diffusion layer. The charging of the capacitor can be modeled as shown in Figure 3. The voltage across the capacitor follows the equation

$$U = V(1 - e^{-\frac{t}{RC}})$$

for an initial charge of 0. The value RC is known as the *time constant*. At time $t = RC$ the voltage has reached 63% of its final value. R is directly proportional to the length of the wire. Thus, long wires means slow circuits in MOS technology. There exist driver techniques that can overcome this problem in part [7], but at best the time to charge the gate may be proportional to the logarithm of the wire length.

Note that if speed of light is the limiting factor, then the wire length is again very important. Hence, in addition to area, we will also discuss the maximum wire length of the layouts we consider.

The Thompson Grid model can be used to establish some area and wirelength properties for networks as a function of their topological characteristics.

Theorem. The minimum area A of a network with bisection width B satisfies the relation $A \geq B^2$.

Proof. With the network laid out on a grid, consider a vertical cut of minimum height. This cut must cross at least $B - 1$ horizontal tracks, since there is at most one jog to split the nodes in the left and right parts. A total of B tracks must be cut since the minimum bisection width is B . The same argument can be used for a horizontal cut, which hence must have width at least $B - 1$ tracks. Hence,

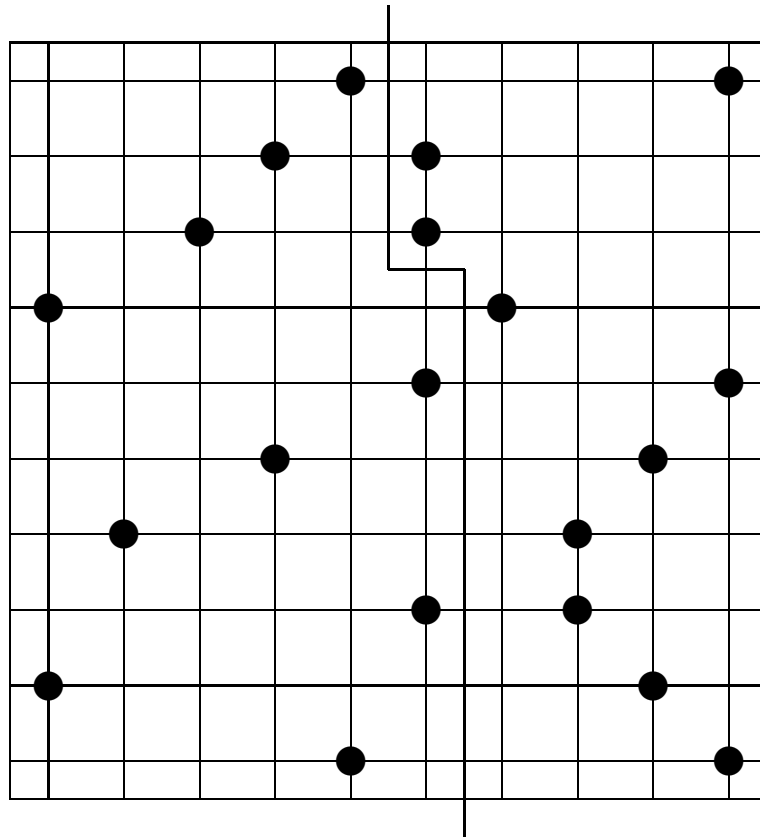


Figure 4: Bisecting a collection of nodes laid out on a planar grid.

$$A = \text{height} \times \text{width} \geq (B - 1)^2 = O(B^2).$$

QED

Figure 4 illustrates the idea in the proof.

The theorem above allows us to establish an interesting relation between area and speed. In a computation like sorting, in the worst case half of the data must be moved from one half to the other. Thus,

$$T \geq O(N)/B \quad \text{or} \quad BT \geq O(N)$$

Squaring this relation yields

$$B^2T^2 \geq O(N^2) \quad \text{or} \quad AT^2 \geq O(N^2)$$

The implication of this relationship, commonly known as the AT^2 bound is that there is a tradeoff between area and speed. A small area must result in a relatively large running time. Conversely, a small running time must result in a relatively large area.

Turning to the wire length we first establish a general lower bound for grid based layouts.

Theorem A lower bound for the maximum wire length L is $L \geq \frac{\sqrt{A}}{D}$, where D is the diameter of the graph.

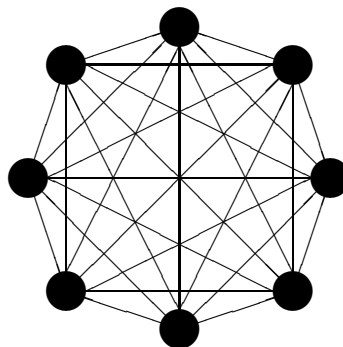
Proof. Assume an approximately square layout. Then, there are \sqrt{A} tracks. Thus, there are two nodes in the graph that are at distance of at least \sqrt{A} . In the graph, these nodes are at most D links apart. Thus, the minimum length of a wire is at least \sqrt{A}/D . QED

From the theorem it follows that $LT \geq O(\frac{N}{D})$. Thus, for a graph of a given diameter, the faster the circuit the longer is the minimum wire length. Also, as the diameter of the graph is reduced, the longer is the shortest wire for a give time T .

The average wire length can also be used as a lower bound for the maximum wire length. Thus, with the total wire length W and the total number of wires being M , we have $L \geq O(W/M)$.

3 Completely Connected Networks

In a completely connected network or *crossbar* each node is connected to every other node. For an N node network, each node has degree $N - 1$. The diameter is 1 and the bisection width is $\lceil \frac{N}{2} \rceil \times \lfloor \frac{N}{2} \rfloor$. Due to the large degree of the nodes, these networks are impractical for more than a few nodes. An 8 node network is shown below:



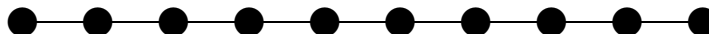
4 Array Networks

4.1 Linear Array

An N node linear array has the following connectivity:

$$i \rightarrow \begin{cases} i \pm 1 & 0 < i < N - 1, \\ i + 1 & i = 0 \\ i - 1 & i = N - 1 \end{cases}$$

A linear array with 10 nodes is shown below:



4.2 Ring

A ring is simply a linear array with wraparound. The connectivity is:

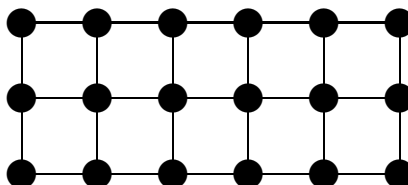
$$i \rightarrow (i \pm 1) \bmod N$$

4.3 Mesh

A mesh is a 2-dimensional array where the lengths of the axes are N_0 and N_1 . There are $N = N_0 \times N_1$ nodes. For a $N_0 \times N_1$ mesh ($N_0, N_1 > 1$), the connectivity is:

$$(i, j) \rightarrow \begin{cases} (i \pm 1, j) & 0 < i < N_0 - 1, & 0 \leq j < N_1 \\ (i + 1, j) & i = 0, & 0 \leq j < N_1 \\ (i - 1, j) & i = N_0 - 1, & 0 \leq j < N_1 \\ (i, j \pm 1) & 0 \leq i < N_0, & 0 < j < N_1 - 1 \\ (i, j + 1) & 0 \leq i < N_0, & j = 0 \\ (i, j - 1) & 0 \leq i < N_0, & j = N_1 - 1 \end{cases}$$

A 3×6 mesh is shown below:



In practice, a linear array may in fact be laid out in a two or three dimensional gridlike manner. In such situations, it is clearly of interest to find out both what the gains would be from actually creating the additional connections inherent in a two-dimensional or three dimensional grid, as

well as the associated cost. One additional aspect of using grids of two or higher dimensions is, that for certain computations, a good algorithm for linear arrays may be known, and it is of interest to find out how the linear array can be emulated on the higher dimensional arrays.

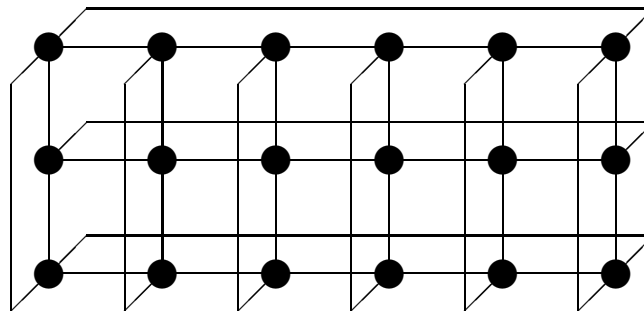
The emulation of linear arrays and rings on higher dimensional arrays is first a question of whether the graph is Hamiltonian or not, second a question of finding an embedding efficient with respect to expansion and dilation. It is also of interest to consider the emulation of a mesh of two or more dimensions on another mesh of two or more dimensions, possibly different from the number of dimensions of the mesh to be emulated. We will discuss such emulations in later lectures.

4.4 Torus

A torus is a mesh with wraparound along both axes. The connectivity is:

$$(i, j) \rightarrow \begin{cases} ((i \pm 1) \bmod N_0, j) & 0 \leq i < N_0, \quad 0 \leq j < N_1 \\ (i, (j \pm 1) \bmod N_1) & 0 \leq i < N_0, \quad 0 \leq j < N_1 \end{cases}$$

A 3×6 torus is shown below:



4.5 Twisted Torus

A twisted torus is a mesh with wraparound in which the wrapping is skewed. With a skewing distance of one, the last element in a column is connected to the first element in the next column and similarly for the rows. If we let s_c be the skewing distance between columns and s_r the skewing distance between rows, then the connectivity is:

$$(i, j) \rightarrow \begin{cases} (i \pm 1, j) & 0 < i < N_0 - 1, \quad 0 \leq j < N_1 \\ (i + 1, j) & i = 0, \quad 0 \leq j < N_1 \\ (N_0 - 1, (j - s_c) \bmod N_1) & i = 0, \quad 0 \leq j < N_1 \\ (i - 1, j) & i = N_0 - 1, \quad 0 \leq j < N_1 \\ (0, (j + s_c) \bmod N_1) & i = N_0 - 1, \quad 0 \leq j < N_1 \\ (i, j \pm 1) & 0 \leq i < N_0, \quad 0 < j < N_1 - 1 \\ (i, j + 1) & 0 \leq i < N_0, \quad j = 0 \\ ((i - s_r) \bmod N_0, N_1 - 1) & 0 \leq i < N_0, \quad j = 0 \\ (i, j - 1) & 0 \leq i < N_0, \quad j = N_1 - 1 \\ ((i + s_r) \bmod N_0, 0) & 0 \leq i < N_0, \quad j = N_1 - 1 \end{cases}$$

Two-dimensional arrays with wraparound, known as *tori*, can be laid out using $2N_0$ horizontal tracks and $2N_1$ vertical tracks for wires for a total area proportional to $4(N_0 \times N_1)$. The maximum wire length is two. Such a layout can be created by simply applying the technique we used for laying out a ring in one dimension to both dimensions of the tori.

What is the area requirement for a twisted torus as a function of the twisting factors?

4.6 Three dimensional meshes

From a construction point of view, a chip and a printed circuit board are essentially two-dimensional media, even though each may have several layers. But, for the construction of systems larger than what fits in either of these media, three spatial dimensions are used. Since the diameter of a three dimensional cubic mesh of N nodes is $3\sqrt[3]{N}$, which compares favorably with $2\sqrt{N}$ for a square mesh, it is natural to explore the merits of building and using three dimensional meshes.

The bisection width of a cubic mesh with an even number of nodes along each axes is $N^{\frac{2}{3}}$. Thus, one lower bound for sorting on a cubic mesh is $\frac{1}{2}N^{\frac{1}{3}}$. The diameter is another lower bound. Both bounds are of order $O(N^{\frac{1}{3}})$. Thus, if an algorithm can be found that sort in a time proportional to the lower bound, it can be expected to be faster than sorting on a mesh. Each node in a three-dimensional mesh has six channels, the boundary nodes excepted. Thus, if the unit to be packaged into a planar unit is a single node, then the width of each channel is a third of that of a channel of a linear array, and $\frac{2}{3}$ rd of that of a channel of a two-dimensional array. If the subsystem packaged onto a planar unit consists of a cubic submesh of M nodes, each channel is of width $\frac{w}{6M^{\frac{2}{3}}}$ compared to $\frac{w}{2}$ for the linear array and $\frac{w}{4\sqrt{M}}$ for the two-dimensional mesh. w is the total width of all channels for the packaging unit. Thus, for a fixed perimeter and planar layouts, optimal sorting times for the linear array, the two-dimensional mesh and three-dimensional meshes compares as $O(N)$, $O(M^{\frac{1}{2}}N^{\frac{1}{2}})$ and $O(M^{\frac{2}{3}}N^{\frac{1}{3}})$.

We will later discuss sorting algorithms of optimal order. Another issue in constructing three-dimensional meshes is their area requirements when laid out in a plane. Whereas an M node two-dimensional mesh can be laid out in area $O(M)$, that is not true for a three-dimensional mesh with M nodes. Also, a two-dimensional mesh can be laid out in the plane with all

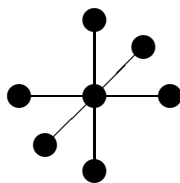


Figure 5: A 7–point stencil in two dimensions.

channels being of order $O(1)$, even if a reshape is required. However, this is not possible for three–dimensional arrays laid out in the plane.

Regular discretization of three–dimensional domains leads to regular three–dimensional meshes. Using Jacobi iterations to solve a set of partial differential equations discretized by 7–point stencils leads to the need to emulate three–dimensional meshes. The common 7–point stencil in Figure 5 is derived similarly to the 5–point stencil in two dimensions.

Another example of where three–dimensional arrays leads to fewer steps than the same computation on two–dimensional arrays is matrix–matrix multiplication. We will discuss matrix–matrix multiplication on three–dimensional meshes later.

For the layout of a three–dimensional array, assume for simplicity that $N_0^3 = N_1^3 = N_2^3 = N^{\frac{1}{3}}$. Then, an approximately square layout is obtained by laying out the nodes on a plane perpendicular to the surface of the layout, and parallel to the horizontal tracks between tracks representing points of the cube in the plane of the embedding surface, as shown in Figure 6. The other planes perpendicular to the embedding surface, but aligned with the vertical tracks are treated analogously. Thus, $N^{\frac{1}{3}}$ tracks are needed in both the horizontal direction and the vertical direction between each pair of tracks representing rows or columns of the plane aligned with the embedding surface. Thus, a total of $N^{\frac{1}{3}} \times N^{\frac{1}{3}} = N^{\frac{2}{3}}$ horizontal and vertical tracks are required. The area of this layout is $N^{\frac{2}{3}} \times N^{\frac{2}{3}} = N^{\frac{4}{3}}$. Hence, an upper bound on the area is $O(N^{\frac{4}{3}})$. The maximum wire length is $O(N^{\frac{1}{3}})$.

This straightforward layout is indeed optimal with respect to area and wire length. It is easy to see that the bisection width is $N^{\frac{2}{3}}$ and that the area thus must be $O(N^{\frac{4}{3}})$.

For our three–dimensional mesh laid out in the plane the Thompson grid model yields a maximum wire length of at least $O(N^{\frac{1}{3}})$.

Thus, the layout is optimal with respect to both area and maximum wire length.

It is easily verified that if instead the planes parallel to the embedding surface are placed along one axis to form a layout with $N^{\frac{1}{3}} \times N^{\frac{2}{3}}$ nodes, then $N^{\frac{1}{3}}$ tracks are still required between each pair of rows in the plane of the embedding surface, since there are $N^{\frac{1}{3}}$ nodes all of which must connect to another plane placed at a distance of $N^{\frac{1}{3}}$.

We note, to strictly adhere to the Thompson grid model, in the layouts we have described each node should be represented as a 2×2 subgrid, and a total of $2N^{\frac{2}{3}}$ horizontal and vertical tracks are required for an area of $2N^{\frac{2}{3}} \times 2N^{\frac{2}{3}} = 4N^{\frac{4}{3}}$.

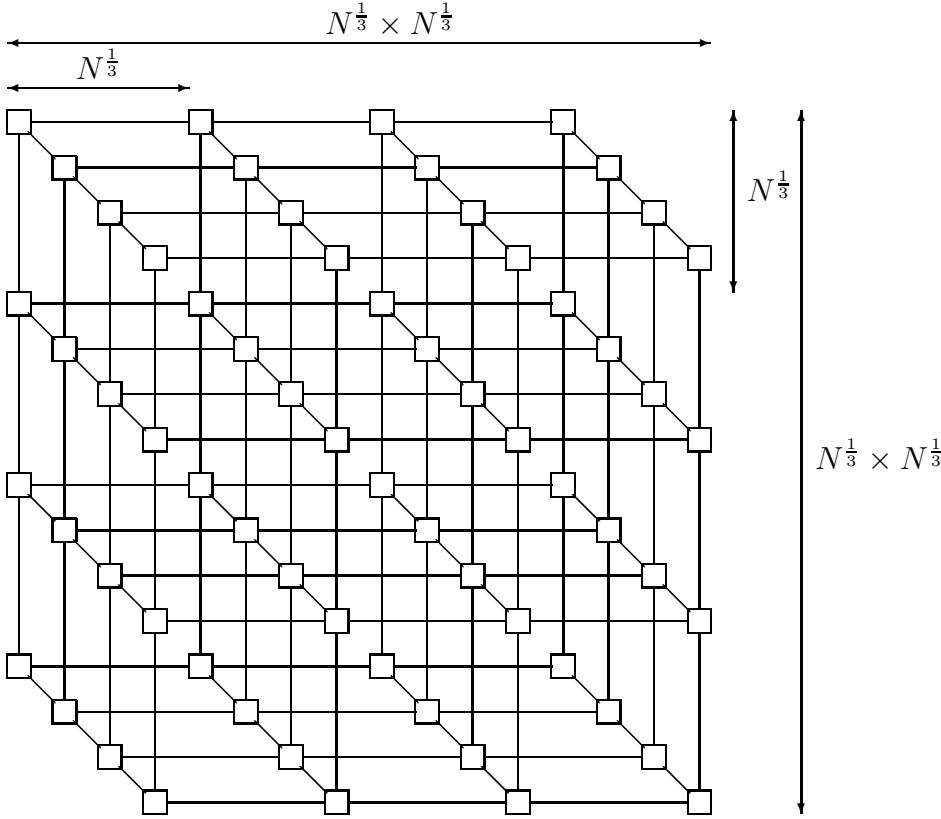


Figure 6: Layout of a cube in two dimensions.

4.7 Multidimensional Arrays

These are straightforward generalizations of a mesh to more than two dimensions.

4.8 Multidimensional Tori

These are straightforward generalizations of a torus to more than two dimensions.

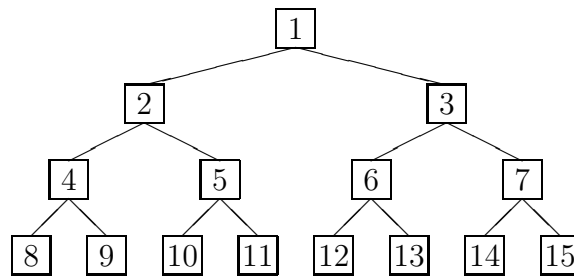
5 Tree Networks

5.1 Complete Binary Trees

A nodes of a d -level complete binary tree with can be labeled from 1 to $2^d - 1$, with the connectivity being:

$$i \rightarrow \{ 2i, 2i + 1 \} \quad 0 \leq i < 2^{d-1}$$

A $2^4 - 1$ node complete binary tree is shown below:



5.1.1 Layout of complete binary trees

A complete binary tree layout is shown in Figure 7. The width of this layout is $(N + 1)/2$ and its height is $\log_2(N + 1) - 1$, where $N = 2^{h+1} - 1$. The area is $O(N \log_2 N)$. The maximum wire length is $(N + 1)/4$. One of the merits of this layout is that all the leaf nodes are on the boundary. If the leaf nodes are used for input and output, then this feature is highly desirable.

The layout in Figure 7 is straightforward. However, compared to the layout of the linear array and the two-dimensional mesh, this complete binary tree layout requires both larger area and longer wires. It is natural to ask:

1. Does there exist a layout with area $O(N)$?

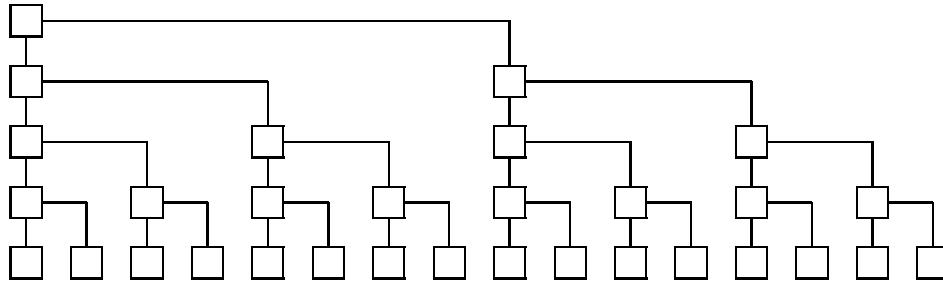


Figure 7: A binary tree.

2. What is the minimum maximum wire length?
3. Does there exist layouts that minimizes both the area and the maximum wire lengths?
4. How can the binary tree be partitioned into parts that can be replicated and used for the construction of arbitrarily large trees?

In an attempt to reduce the area we can try to place the leaf nodes along all four boundaries of the bounding box for the tree. Placing half of the leaf nodes along the top boundary and half along the bottom boundary reduces the width by a factor of two, but doubles the height. Thus, though the aspect ratio improved, the area remained the same. A few leaf nodes ($O(\log_2 N)$ nodes) can be moved from the bottom to one of the sides without increasing the height. A similar rearrangement can be made for the top nodes. This rearrangement reduces the area by a lower order term ($O(\log_2^2 N)$). The area remains $O(N \log_2 N)$. Moving additional nodes increases the height, and the area. Distributing the leaf nodes evenly along all four boundaries results in an area of order (N^2).

The layout in Figure 7 indeed has an area of optimal order with leaf nodes assigned to the boundary.

Theorem. [2, 12] Any complete binary tree with N -nodes laid out with the leaf nodes on the boundary of a rectangle must have area $O(N \log_2 N)$.

Proof. The proof is based on the total wire length for the complete binary tree with the leaf nodes on a line. Let the total wire length of a tree of height h be $W(h)$. Let $M(h)$ be the lengths of all wires in a tree of height h with a longest path from the root to a leaf excluded. Then,

$$M(h) \geq W(h - 1) + M(h - 1)$$

This fact can be seen as follows. Combine two $h - 1$ level trees to create one level h tree. Exclude a longest path from the root to one leaf in the tree of height h . Then, the total wire length in one of the two original subtrees is $W(h - 1)$, and in the other it is $M(h - 1)$. The idea is illustrated in Figure 8

In addition, the following relationship also holds,

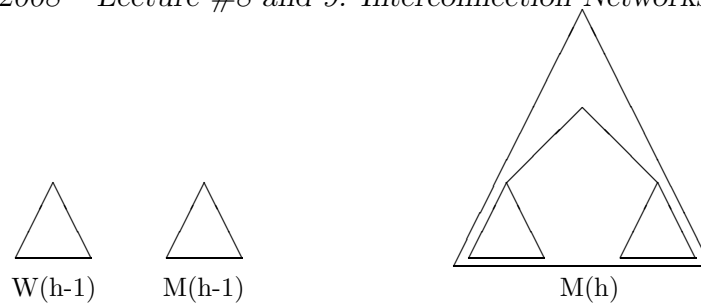


Figure 8: Recursive relationships of wire lengths with longest path from root to a leaf excluded.

$$W(h) \geq 2M(h - 1) + 2^{h-1}$$

To see that this relationship indeed is true, consider the two subtrees of the root. Let v be the leftmost leaf node and let u be the rightmost leaf node of the other subtree. Since u is the rightmost leaf node of the other subtree, the distance between nodes v and u must be sufficiently large to leave room for all leaf nodes, which in a tree of height h is 2^{h-1} . Thus, v and u are at least a distance of 2^{h-1} apart with all leaf nodes placed on a line. Now, remove the path from v to u . This path does not necessarily contain the longest path in either subtree. Furthermore, the removed path must be at least of length 2^{h-1} . Hence, $W(h) \geq 2M(h - 1) + 2^{h-1}$.

Substituting the second expression into the first expression gives

$$M(h) \geq 2M(h - 2) + M(h - 1) + 2^{h-2}$$

$M(0) = 0$ and $M(1) = 1$. By induction, $M(h) \geq h2^h/6$. It follows that the total wire length $W(h)$ satisfies the same relation. Hence, since at least half of the wire length must be either in horizontal or vertical tracks, the area is at least $W(h)/2$ and we have proved that the area for a complete binary tree with the nodes on a line has an area of at least $O(N \log_2 N)$. With the arguments preceding the theorem we have now proved that any complete binary tree with the leaf assigned to the boundary of a rectangle must have area of at least $O(N \log_2 N)$. QED

With leaf nodes along both the top and bottom boundaries of Figure 7, the maximum wire length is $\sim N/8$.

In the above tree layout we imposed a constraint in the placement of the leaf nodes. In discussing the layout of other networks we did not impose any constraint on the placement of nodes. However, if we impose the constraint that all nodes of an N node mesh are placed on the boundary, then an area of order $O(N\sqrt{N})$ would be required for a two-dimensional square mesh with all nodes placed on a line. The maximum wire length would be $O(\sqrt{N})$ instead of $O(1)$. What is the minimum area required for a d dimensional mesh with the constraint of all nodes on the boundary? What is the minimum maximum wire length?

Relaxing the constraint on the placement of the tree nodes allows for layouts with an area of order $O(N)$, and a maximum wire length of optimal order, $O(\sqrt{N} \log_2 N)$.

The so-called H-tree layout, shown in Figure 9, requires area $O(N)$.

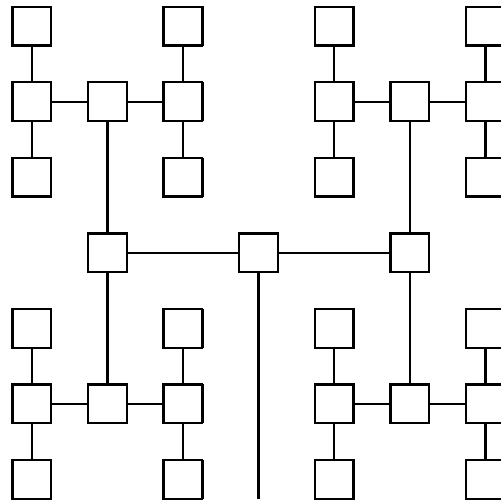


Figure 9: H-tree layout of a complete binary tree.

Theorem. An N node complete binary tree can be laid out in area $O(N)$.

Proof. The number of nodes in the H-tree is $2 \cdot 4^i - 1 = 2^{2i+1} - 1$. The number of levels in the complete binary tree is $2i$. The number of horizontal and vertical tracks is $2^{i+1} - 1$ for a complete binary tree of height $2i$. This is easily shown by induction. Hence, the area is $2^{2i+2} - 2 \cdot 2^{i+1} + 1 \approx 2N$. QED

The wire length in the H-tree layout in Figure 9 doubles for every pair of levels of the binary tree. Thus, the maximum wire length is 2^{i-1} for a tree of height $2i$. Thus, the maximum wire length is $\frac{\sqrt{N}}{2\sqrt{2}}$. Compared to the layout with all nodes on the boundary, the maximum wire length is reduced by a factor of $\sim \frac{\sqrt{N}}{2\sqrt{2}}$. Relaxing the constraint to place the nodes on the boundary yields a substantial improvement in both area and maximum wire length.

The area is of optimal order, but the wire length is not. A lower bound on the wire length based on the diameter is $O(\sqrt{N}/\log_2 N)$.

Does there exist an area optimal layout with minimum maximum wire length?

The answer is yes [8, 10, 12]. A schematic layout is shown in Figure 10.

Theorem. An N node complete binary tree can be laid out in the plane in area $O(N)$ with a maximum wire length of $O(\frac{\sqrt{N}}{\log_2 N})$.

Proof. The root of the tree is in, say, the bottom left corner of the horizontal box. This box contains a tree of $h/4$ levels, and thus $2^{h/4}$ leaf nodes, all of which are placed on the upper boundary. This boundary is of length $2^{h/2}$, and easily holds the leaf nodes of the tree in the bottom box. The height of the bottom box is made to be $2^{h/4}$ to allow for flexibility in the tree layout in the box. This generous height will not increase the order of the layout area, but allow a layout that does not exceed the optimal maximum wire length.

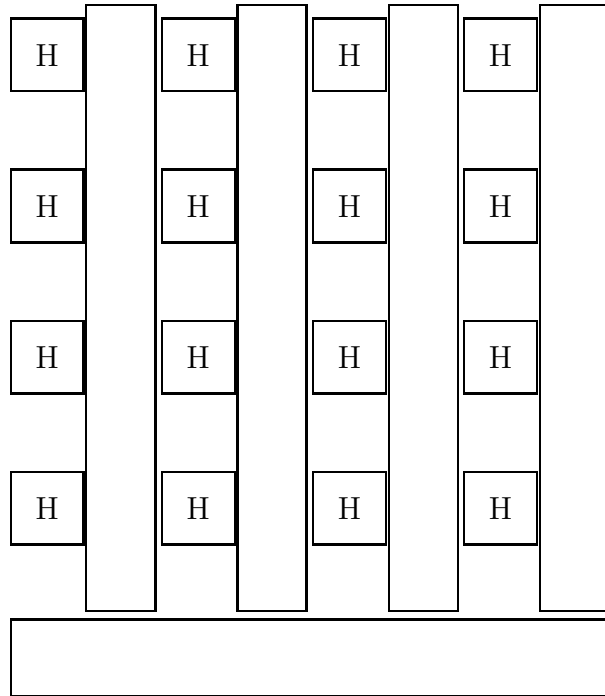


Figure 10: Maximum wire length optimal H-tree layout of a complete binary tree.

Each of the vertical boxes are just like the horizontal box. The $2^{h/4}$ leaf nodes in each of these trees are placed on the left boundary. Each of these nodes form the root of an H-tree with $h/2$ levels of a complete binary tree. Each H-tree has a side of $2^{h/4}$. The total height of the H-trees connecting to a vertical box is $2^{h/2}$, which matches the height of the vertical box. The width of our layout is $(2 \cdot 2^{h/4}) \times 2^{h/4} = 2 \cdot 2^{h/2}$, which matches the width of the bottom box within a factor of two. The height is $2^{h/2} + 2^{h/4}$, which is $O(2^{h/2})$. Thus, we know that the area is of optimal order.

The maximum wire length in each H-tree is $O(\sqrt{2^{h/2}}) = O(2^{h/4})$, which is below the bound $O(2^{h/2}/h)$. Thus, the maximum wire length must occur in the rectangular boxes. Since there are $h/4$ levels, no horizontal wire needs to be longer than $2^{h/2}/(h/4)$ for the bottom box. This quantity is of optimal order. Vertical wires are at most $2^{h/4}$ in length. There is ample room to move the nodes at the bottom edge to assure that the maximum horizontal wire length constraint is satisfied. The number of horizontal tracks is equal to the number of leaf nodes, and conflicts can be avoided. Thus, we have a layout that is optimal with respect to both area and maximum wire length. QED

For additional results on binary tree layouts see [1, 13].

5.1.2 Partitioning

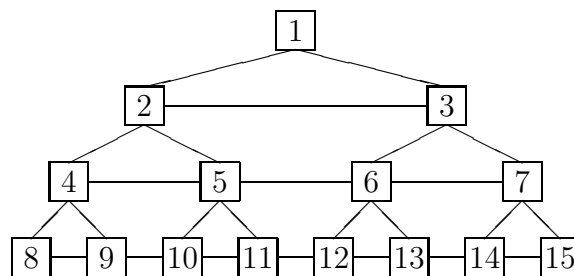
Complete binary trees can be partitioned in a scalable way. It is possible to construct trees such that four channels per partition suffices, regardless of what size subtrees fits within the partition, and regardless of what size complete binary tree is constructed out of the partitions. One spare per partition suffices to accomplish this goal. All connections necessary for the assembly of the partitions into a large complete binary tree are external to the partitions. This very nice partitioning property of complete binary trees, may make them competitive with, for instance, two-dimensional meshes when channel width is important for performance. With a total of w bits crossing the partition boundary, each channel is of width $w/4$ for the complete binary tree, but only of width $w/(4\sqrt{N})$ for a two-dimensional square mesh, and of width $w/(6N^{\frac{2}{3}})$ for a three-dimensional cubic mesh.

5.2 X-Trees

X-trees are complete binary trees with the nodes of each level of the tree connected as a linear array. The connectivity is:

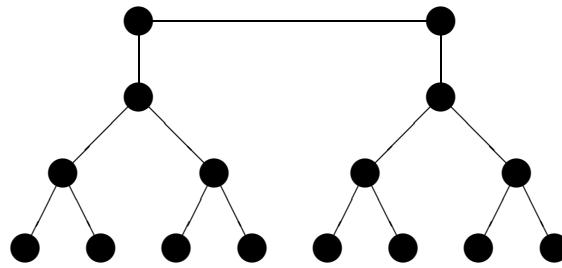
$$i \rightarrow \begin{cases} 2i, 2i + 1 & 0 \leq i < 2^{d-1} \\ i + 1 & 2^j \leq i < 2^{j+1}, 0 < j < d \end{cases}$$

A $2^4 - 1$ node X-tree is shown below:



5.3 Two-Rooted Trees

Two-rooted trees are complete binary trees in which the root has been split into two nodes. A 16 node two-rooted tree is shown below:



6 Hypercubic Networks

The hypercube is one of the most powerful networks for parallel computation. It is well suited for many different tasks. More important, the hypercube can efficiently simulate any other network of the same size. Unfortunately, as we shall soon see, the degree of hypercube nodes is not constant. To address this problem, several other related networks have been proposed. These networks are all capable of emulating the hypercube with constant or logarithmic slowdown. Together with the hypercube, these related networks form the hypercubic class of networks. For an extensive discussion of these networks, see Chapter 3 of [4].

6.1 The Hypercube

The hypercube is also known as the *boolean* or *binary* cube. These networks are multidimensional arrays with an axis extent of two for each axis. Thus, $N = 2^d$, where d is the number of axes (or dimensions), and $N_j = 2$ for $0 \leq j < d$.

Let a node index be $i = (i_{d-1}i_{d-2} \dots i_0)$, where i_j is the index value along axis j . Note that $i_j \in \{0, 1\}$ and that the Cartesian coordinate representation of a node index is identical to the binary decomposition of i .

The connectivity can be represented as:

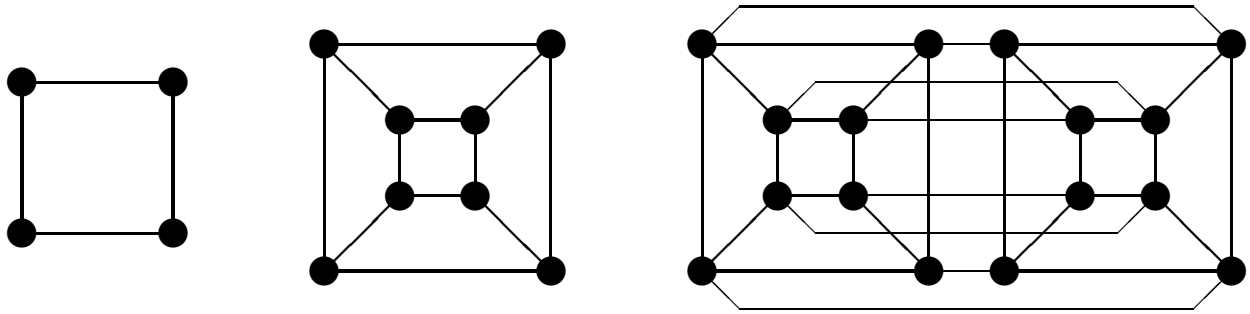
$$i \rightarrow i \oplus 2^j \text{ for } j = \{0, 1, \dots, d - 1\}, \text{ and } 0 \leq i < N$$

where \oplus denotes the bitwise exclusive OR operation. An alternative definition is:

$$i = (i_{d-1}i_{d-2} \dots i_j \dots i_0) \rightarrow (i_{d-1}i_{d-2} \dots \bar{i}_j \dots i_0) \text{ for } j = \{0, 1, \dots, d - 1\}, \text{ and } 0 \leq i < N$$

where \bar{i}_j denotes the complement of i_j .

Two, three, and four dimensional hypercubes are shown below:



6.1.1 Layout

For the layout of binary cubes with an even number of dimensions we place the nodes on a $\sqrt{N} \times \sqrt{N}$ grid. Each node is represented by a $\frac{1}{2} \log_2 N \times \frac{1}{2} \log_2 N$ subgrid in order to allow for the $\log_2 N$ channels per node. Figure 11 shows the layout of a 64 node binary cube.

The number of tracks for routing between nodes within a row is one for the first dimension, two for the second dimension, four for the third dimension, etc. Thus, the total number of tracks for routing between nodes in a row is $1 + 2 + 4 + 8 + \dots + 2^{\log_2 \sqrt{N}-1} = \sqrt{N} - 1$. It follows that the total height of the layout is $\sqrt{N}(\frac{1}{2} \log_2 N + \sqrt{N} - 1) \approx N$. The width is also N . Thus the area is N^2 .

This layout has an area of optimal order, since the bisection width is $N/2$.

The wire length is $N/2$. The diameter of the binary cube is $\log_2 N$, and the total number of wires is $\frac{1}{2} N \log_2 N$. Thus, a lower bound for the maximum wire length is $N/\log_2 N$, whether determined by the average wire length, or by the diameter. Hence, our layout is nonoptimal with respect to wire length by at most a factor of $O(\log_2 N)$.

Is there a better lower bound, or a layout with shorter maximum wire length?

Comparing a three-dimensional mesh and a binary cube occupying the same area, we notice that we can fit $A^{\frac{3}{4}}$ nodes of the three-dimensional mesh, while only $A^{\frac{1}{2}}$ binary cube nodes. With respect to the maximum wire length, it is $A^{\frac{1}{4}}$ for the mesh while it is $A^{\frac{1}{2}}$ for the binary cube.

With respect to partitioning of the binary cube, the width of each channel that must leave the partition depends both on the number of nodes in the partition, and the number of nodes in the total system. If A_p is the area for a partition, then each of the $A_p^{\frac{1}{2}}$ nodes in the partition requires $n - \log_2 A_p^{\frac{1}{2}}$ channels to nodes not contained in the partition. Hence, the number of channels leaving the partition is $A_p^{\frac{1}{2}}(n - \frac{1}{2} \log_2 A_p)$ compared to $6A_p^{\frac{1}{2}}$ for a three-dimensional mesh. For both these estimates we assumed for simplicity that channels leaving the partition A_p do not consume any area in the partition. This assumption is not valid when all connectors are confined to the perimeter of the partition.

As an example, consider a partition that holds a $4 \times 4 \times 4 = A^{\frac{3}{4}}$ submesh. Then, the partition

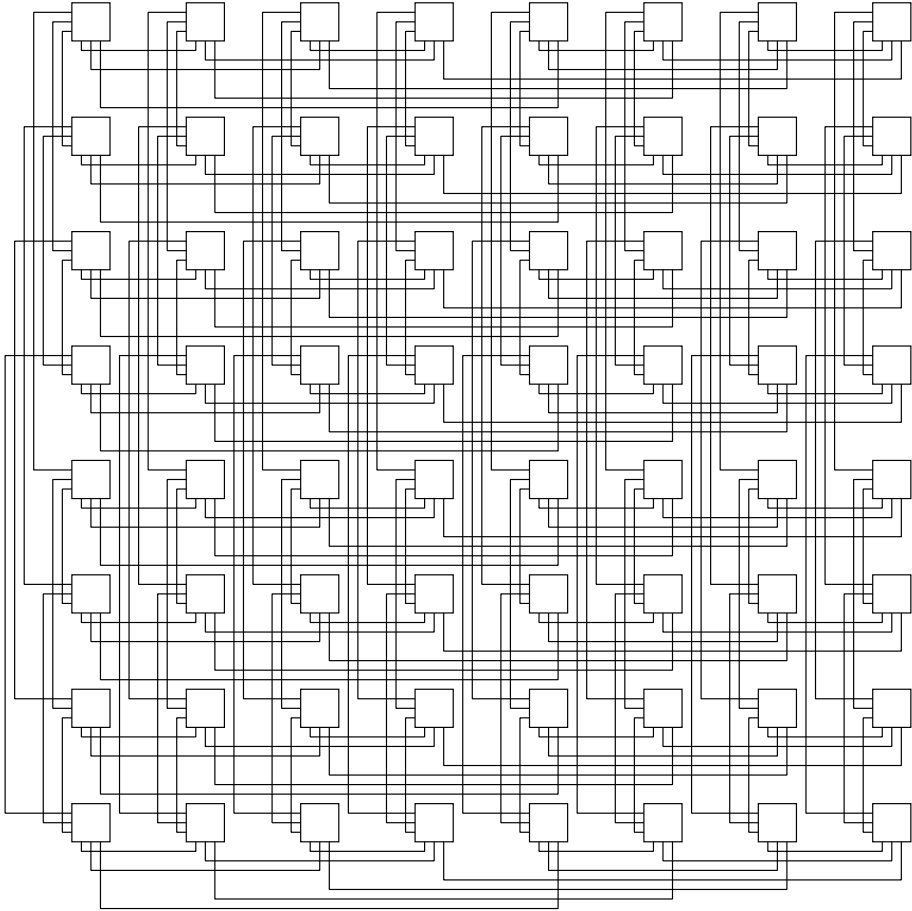


Figure 11: Layout in the plane of a binary cube.

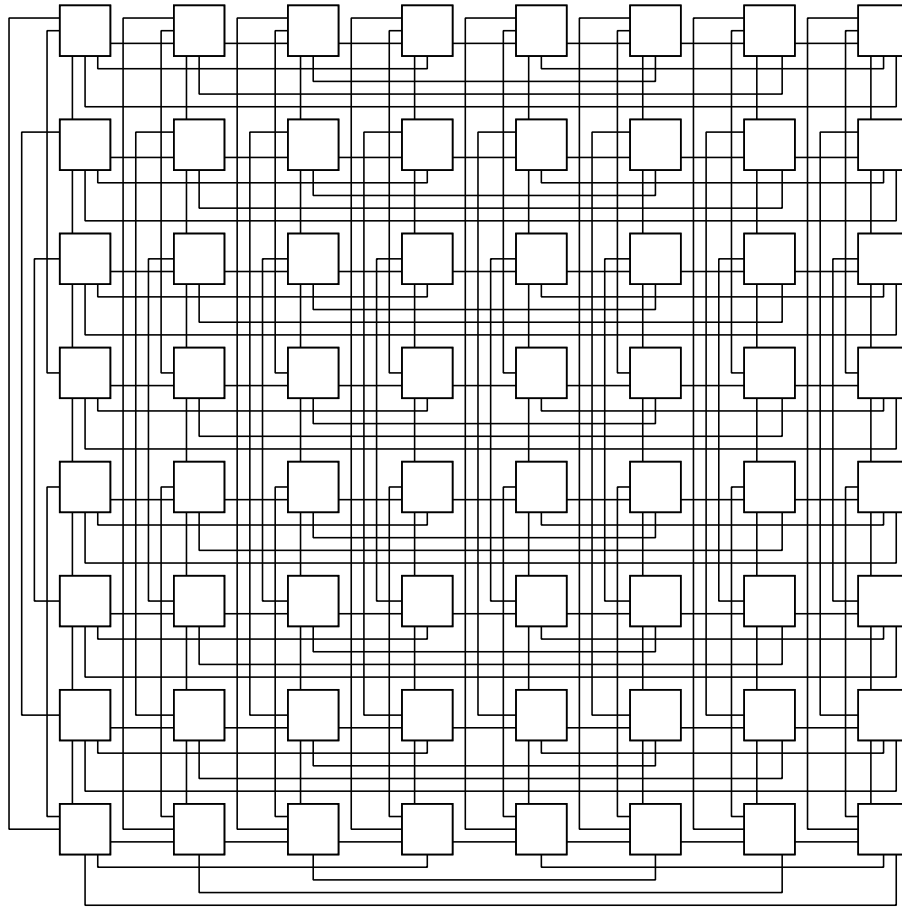


Figure 12: Gray code layout in the plane of a binary cube.

would hold 16 binary cube nodes. The three-dimensional mesh would require $6 \times 16 = 96$ channels off the partition. For a 64k node system, i.e., a thousand partitions in the system, the binary cube network would require $16(16 - 4) = 196$ channels. The width of each channel is half of that of the three-dimensional mesh.

We leave it to the reader to verify that a Gray code based layout requires $\sqrt{N} - \log_2 \sqrt{N}$ tracks for routing between nodes in rows and columns, respectively. Thus, the area is only improved by a lower order term. A Gray code layout is shown in Figure 12.

Note however, that even though the Gray code layout does not yield any substantial reduction in area and increases the maximum wire length by a factor of two, the Gray code layout allows a $\sqrt{N} \times \sqrt{N}$ mesh to be emulated using only wires of unit length.

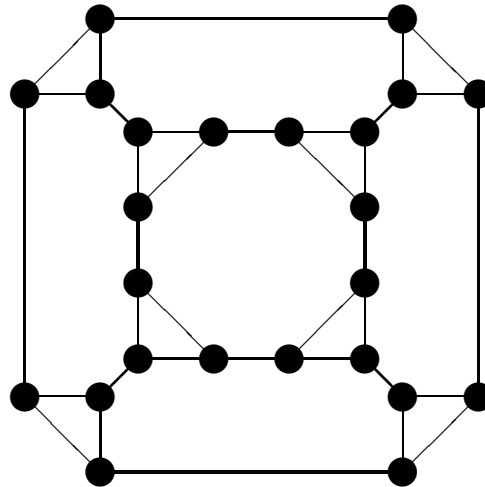
The binary cube is a special case of a hypercube. A k -dimensional hypercube is a mesh of shape $M \times M \times M \times \dots \times M = M^k$. For the binary cube $M = 2$. It can be shown that the hypercube can be laid out in area $O(M^{2(k-1)})$, $k > 1$. With a bisection width of $M^{(k-1)}$ this area is optimal.

6.2 Cube Connected Cycles (CCC) Networks

The cube connected cycles networks are derived from Boolean cube networks by replacing each node of a Boolean cube network with a ring of nodes. Each ring has one node for each edge connecting the replaced cube node with its adjacent (replaced) nodes. Hence, a d dimensional CCC network has $d \cdot 2^d$ nodes. The nodes in the CCC network are represented by two indices (i, j) , where i denotes the replaced cube node (i.e., selects the ring), and j represents the location within the ring. $0 \leq i < 2^d$ and $0 \leq j < d$. The connectivity is:

$$(i, j) \rightarrow \begin{cases} (i \oplus 2^j, j) & 0 \leq i < 2^d, 0 \leq j < d \\ (i, (j \pm 1) \bmod d) & 0 \leq i < 2^d, 0 \leq j < d \end{cases}$$

The first set of edges are the *cube* edges; the second set of edges are the *cycle* edges. A 3-dimensional CCC is shown below:



The CCC can also be drawn in a way similar to the way in which we drew the butterfly network described in the next subsection. The modifications are as follows:

- each row is made into a ring instead of a linear array, i.e., the output is connected to the input in the same row.
- instead of the cross-connections between a pair of rows a single connection is used, and one rank is omitted.

A CCC drawn in this way has d ranks of 2^d nodes each. Figure 13 shows a CCC drawn in this manner.

From Figure 13 it is clear that the layout properties of the CCC is similar to that of the butterfly network.

The relationship of the butterfly network to the binary cube is also clear. If each of the cycles are broken by removing one edge, for instance the edge that connects the last and first

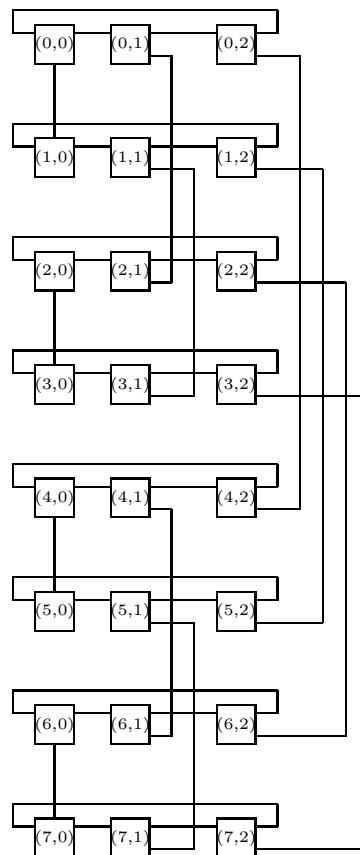


Figure 13: A CCC drawn similarly to a butterfly network.

ranks in Figure 13, then the only difference between the CCC and the butterfly network is the replacement of the cross-connections in the butterfly network by the vertical lines in Figure 13. Identifying all nodes on a line in Figure 13 yields the binary cube network.

6.3 Butterfly Networks

The butterfly network, also known as the FFT network, can be defined in terms of two indices (i, j) , where i is the row index, $0 \leq i < N_0 = 2^d$ and j is the column index, $0 \leq j \leq d = N_1 - 1$. Thus, the total number of nodes N is $(d + 1)2^d$. The connectivity of the butterfly network is:

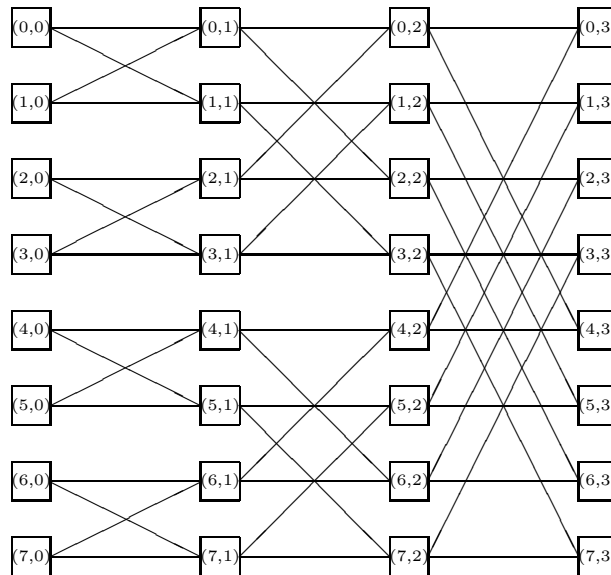
$$(i, j) \rightarrow \begin{cases} (i, j + 1) & 0 \leq i < 2^d, 0 \leq j < d \\ (i \oplus 2^j, j + 1) & 0 \leq i < 2^d, 0 \leq j < d \end{cases}$$

Butterfly networks can also be defined with wraparound, in which case, column d is identified with column 0, i.e., there are only d columns and $0 \leq j < d$. The total number of nodes is $d2^d$.

$$(i, j) \rightarrow \begin{cases} (i, (j + 1) \bmod d) & 0 \leq i < 2^d, 0 \leq j < d \\ (i \oplus 2^j, (j + 1) \bmod d) & 0 \leq i < 2^d, 0 \leq j < d \end{cases}$$

As with the multidimensional arrays, wrapped butterfly networks can be defined with skewing.

A butterfly network is shown below:



6.3.1 Layouts of butterfly networks

Butterfly networks are closely related to binary cube networks. For the butterfly network an area optimal layout with minimum maximum wire length $O(N/\log_2 N)$ is known. The wire maximum length is within a factor of $O(\log_2 N)$ of the lower bound, just as for the binary cube.

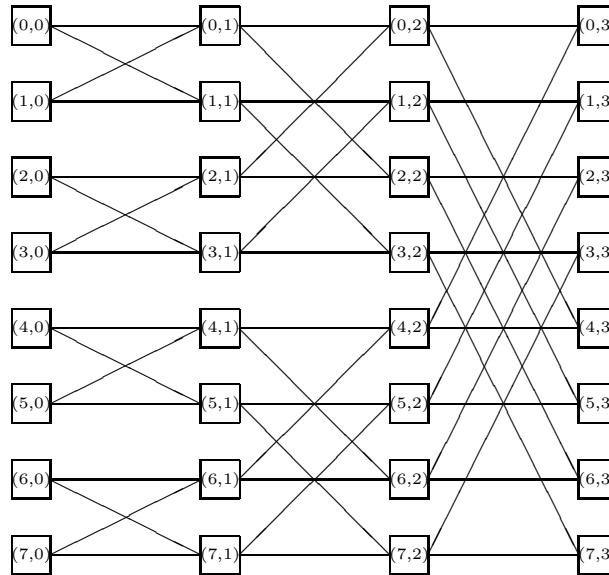


Figure 14: A 8×4 butterfly network.

The butterfly network, also known as the FFT network, is usually defined in terms of two indices (i, j) , where i is the *row index*, $0 \leq i < N_0 = 2^d$ and j is the *column index*, $0 \leq j \leq d = N_1 - 1$. The total number of nodes N is $(d + 1)2^d$. The connectivity of the butterfly network is:

$$(i, j) \rightarrow \begin{cases} (i, j + 1) & 0 \leq i < 2^d, 0 \leq j < d \\ (i \oplus 2^j, j + 1) & 0 \leq i < 2^d, 0 \leq j < d \end{cases}$$

Butterfly networks can also be defined with wraparound, in which case, column d is identified with column 0, i.e., there are only d columns and $0 \leq j < d$. The total number of nodes is $d2^d$.

$$(i, j) \rightarrow \begin{cases} (i, (j + 1) \bmod d) & 0 \leq i < 2^d, 0 \leq j < d \\ (i \oplus 2^j, (j + 1) \bmod d) & 0 \leq i < 2^d, 0 \leq j < d \end{cases}$$

As with multidimensional arrays, wrapped butterfly networks can be defined with skewing.

A butterfly network for $d = 3$ and no wraparound is shown in Figure 14.

For the layout of a butterfly network without wraparound we allow the nodes to be three units high to accommodate the necessary number of horizontal tracks per node. With nodes labeled (i, j) there is one connection to node $(i, j + 1)$ for every node in column j , $0 \leq j \leq d$. This set of wires account for one horizontal track per node. The wires $(i, j) \rightarrow (i \oplus 2^j, j + 1)$ account for the other two horizontal tracks per node. There are 2^{j+1} vertical tracks between columns j and $j + 1$. The total number of vertical tracks for wiring between columns is $2 + 4 + \dots + 2^d = 2(2^d - 1)$. In addition, there are $d + 1$ vertical tracks reserved for the nodes. Thus, the total width of the layout is $2^{d+1} + d - 1$, and the total height is $3 \cdot 2^d$.

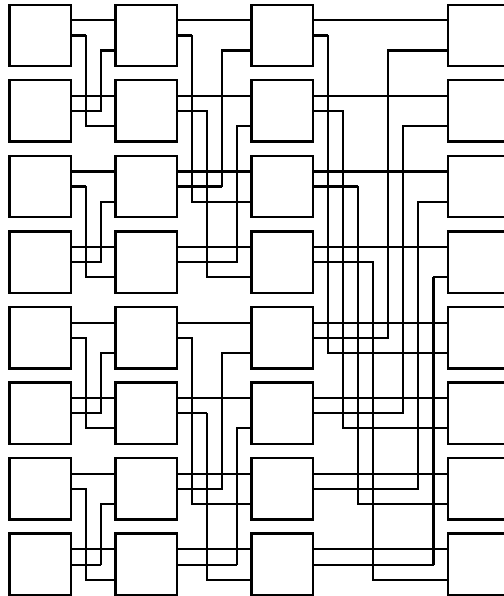


Figure 15: Area optimal layout of a butterfly network.

The layout area is $6 \cdot 2^{2d} + 3d2^d - 3 \cdot 2^d = O\left(\frac{N^2}{\log_2 N}\right)$. This area is of optimal order, since the bisection width is $2^d \approx \frac{N}{\log_2 N}$.

The maximum wire length in our layout is $3 \cdot 2^{d-1} + 2^d = O\left(\frac{N}{\log_2 N}\right)$. The lower bound, based on either the diameter or the average wire length, is $O\left(\frac{N}{\log_2 N}\right)$. Thus, our area optimal layout is nonoptimal with respect to wire length by at most a factor of $\log_2 N$.

6.4 High Radix Butterfly Networks

The butterfly network defined above is a radix-2 network. The fan-in and the fan-out to each node is two. Similar networks can be defined for a higher fan-in and fan-out. A fan-in and fan-out equal to some power of two is most common, such as radix-4, 8, and 16. The number of columns are decreased accordingly.

6.5 Beneš Networks

Beneš networks are two butterfly networks of the same size connected together to form a network of $2d + 1$ columns and 2^d rows. The second network is the mirror image of the first network, i.e., the column index runs backwards.

6.6 Shuffle–Exchange Networks

The shuffle exchange network is often defined in terms of two sets of edges: *shuffle edges* and *exchange edges*. The number of nodes $N = 2^d$. The binary decomposition, $i = (i_{d-1}i_{d-2} \dots i_0)$, is used for the definition.

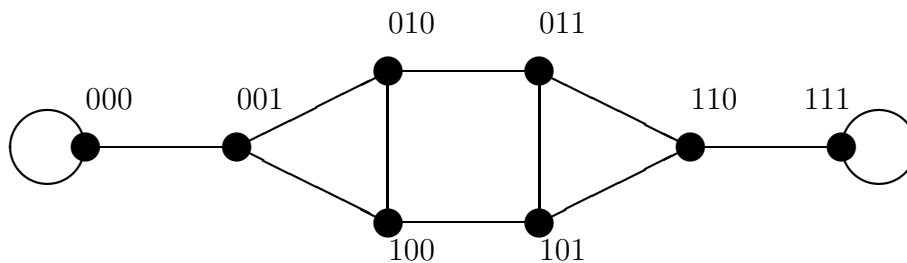
The *shuffle* edges are:

$$i = (i_{d-1}i_{d-2} \dots i_0) \rightarrow \begin{cases} (sh(i) = i_{d-2}i_{d-3} \dots i_0i_{d-1}) & 0 \leq i < 2^d \\ (sh^{-1}(i) = i_0i_{d-1}i_{d-2} \dots i_2i_1) & 0 \leq i < 2^d \end{cases}$$

The *exchange* edges are:

$$i \rightarrow i \oplus 1 \text{ for } 0 \leq i < 2^d.$$

An 8–node shuffle–exchange network is shown below. Thick lines denote shuffle edges and thin lines denote exchange edges.

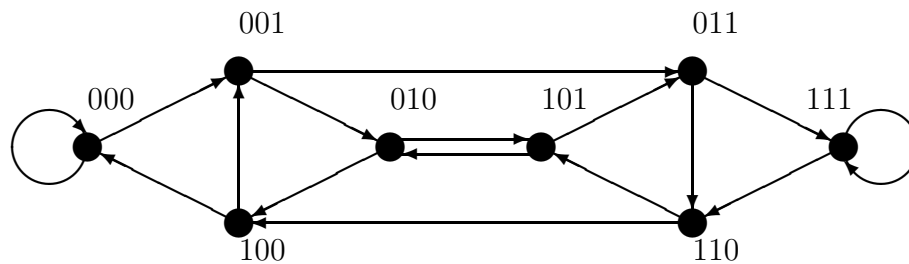


6.7 de Bruijn Networks

The de Bruijn network is related to the shuffle-exchange network. However, unlike the shuffle–exchange, all of the edges are directed. The connectivity is:

$$i = (i_{d-1}i_{d-2} \dots i_0) \rightarrow \begin{cases} sh(i) = i_{d-2}i_{d-3} \dots i_00 & 0 \leq i < 2^d \\ sh(i) = i_{d-2}i_{d-3} \dots i_01 & 0 \leq i < 2^d \end{cases}$$

A $d -$ –dimensional de Bruijn network can be obtained from a $d + 1$ dimensional shuffle–exchange graph by merging nodes that share an exchange edge. An 8–node de Bruijn network is shown below:



6.8 Ω -Networks

An Ω -network is a multistage network like the butterfly network. The connections between stages are defined as shuffle-connections. There are 2^d rows and $d+1$ columns. The connections between columns are the same for all successive column pairs:

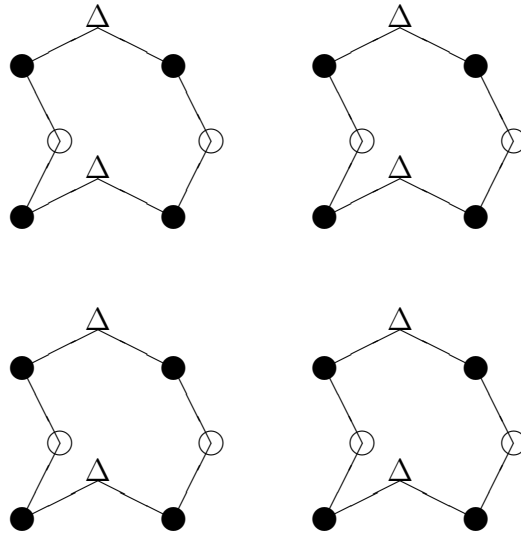
$$(i, j) \rightarrow \{ (sh(i), j + 1) \quad 0 \leq i < 2^d, 0 \leq j < d$$

The Ω -network and the butterfly network are actually identical. They are just drawn (and labeled) differently.

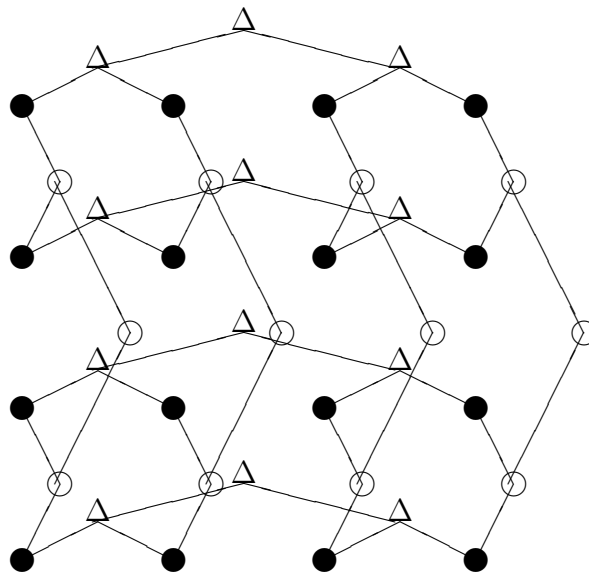
6.9 Mesh of Trees

An n -dimensional mesh of trees of $N = \sum_{i=1}^n 2^{k_i}$ leaf nodes consists of an n -dimensional grid with 2^{k_i} nodes along the i th axis. The leaf nodes along each axis are connected as a complete binary tree. Leaf nodes can be given addresses in a multidimensional address space, such that a node address is $(a_{n-1}, a_{n-2}, \dots, a_0)$, where $0 \leq a_i < 2^{k_i}$. Leaf nodes are in the same tree if and only if they differ in precisely one digit position. For example, all nodes with address $(a_{n-1}, *, a_{n-3}, \dots, a_0)$ are in the same tree.

Four 2×2 mesh-of-tree networks



A 4×4 mesh-of-tree networks



6.10 Fat-Trees

Fat trees are trees in which the width of the channels increases towards the root. Fat-trees can be constructed as ordinary trees, such as binary trees, or in general k -ary trees. A binary fat-tree is shown in Figure 16. In this case the width of the channels doubles for every level of the tree. Thus, for this tree, each leaf node in effect has its own channel to the root. There is no contention for channels even if all nodes in one subtree of the root must exchange its data with the other subtree of the root. Compared to a binary tree, this is an improvement in the data motion capacity of the root by a factor equal to the number of leaf nodes. However, it is not inherent in the definition of a binary fat-tree that the capacity between a node and its

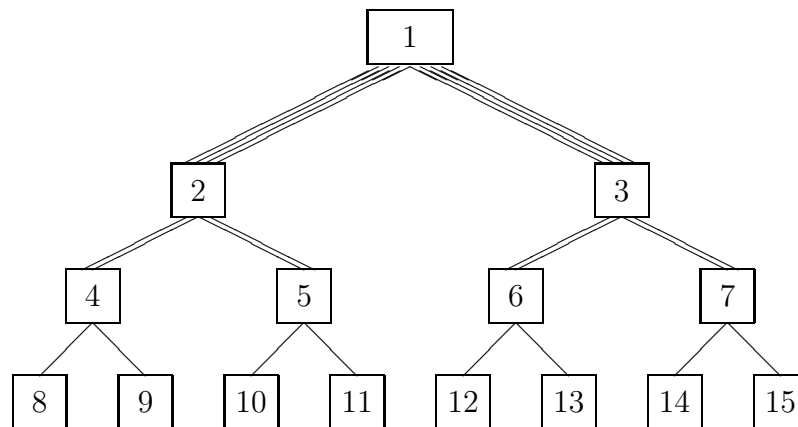


Figure 16: A binary fat-tree with channel widths doubling for each level towards the root.

parent must double. However, there must be some growth in the channel width from leafs to root for a tree to be considered a fat-tree.

Since most nodes in a fat-tree has multiple inputs and outputs, they require some routing capability between input and output channels. It is not necessary to guarantee that every input channel can route to any output channel in every node in order to assure that a message from any leaf node can be routed to any other leaf node. We will discuss the routing issues further later.

Figure 17 shows a two-level 4-ary fat-tree. Each node has four children. The number of channels from a node to its parents doubles for every level in this 4-ary fat-tree. Thus, the total number of channels is reduced by a factor of two for each level in moving from the leafs towards the root. Though this may at first seem like no gain compared to a complete binary tree, it indeed represents an improvement since the tree is a 4-ary tree. For an ordinary 4-ary tree $N/4$ leaf nodes may be competing for a channel at the root. With the fat-tree in Figure 17 at most $\sqrt{N}/2$ leaf nodes may compete for a channel at the root, an improvement by a factor of $\sqrt{N}/2$.

The internal node structure in the fat-tree in Figure 17 is a one-stage butterfly network. There are two connections between a leaf and its parents. Each leaf node consists of a 2-input 2-output butterfly network. The number of input and output nodes in a node doubles for each level of the tree. Thus, in the level above the leaf nodes, each node consists of a butterfly stage with four inputs and four outputs. In the next level up, each node consists of an 8-input 8-output butterfly stage. The fat-tree in Figure 17 can be viewed as a *slender* butterfly network.

The Connection Machine CM-5 network is a fat-tree network. It is based on a 4-ary tree. As in Figure 17, the number of channels is reduced by a factor of two between a pair of levels, but only for the two levels closest to the leafs. For the higher levels of the tree, the number of channels is the same for all levels. Such fat-trees are scalable with respect to communications bandwidth. The contention for communication channels in the worst case is a constant factor,

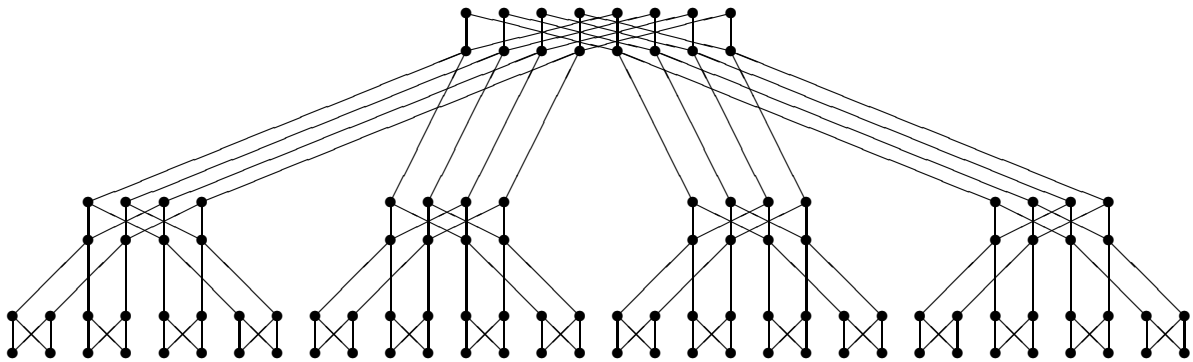


Figure 17: A 4-ary fat-tree of height two.

Network	Area A	Bisection width, B	Diameter D	Max wire length, L	Lower bound max(D, N/B)	Part. size M wires cut
1-D array	N	1	N	1	$N/2$	2
2-D array	N	$N^{1/2}$	$2N^{1/2}$	1	$2N^{1/2}$	$4M^{1/2}$
3-D array	$4N^{4/3}$	$N^{2/3}$	$3N^{1/3}$	$N^{1/3}$	$3N^{1/3}$	$6M^{2/3}$
k-D array	$(d-1)^2 N^{2-\frac{2}{k}}$	$N^{1-\frac{1}{k}}$	$kN^{1/k}$	$N^{1-\frac{2}{k}}$	$kN^{1/k}$	$2kM^{1-\frac{1}{k}}$
Compl. bin. tree	$4N$	1	$\log_2 N$	$\frac{N^{1/2}}{\log_2 N}$	$N/2$	4
Binary cube	N^2	$N/2$	$\log_2 N$	$N/2$	$\log_2 N$	$M(\log_2 N - \log_2 M)$
Butterfly	$\frac{N^2}{\log_2^2 N}$	$\frac{N}{2 \log_2 N}$	$2 \log_2 N$	$\frac{N}{2 \log_2 N}$	$2 \log_2 N$	$2M / \log_2 M$
CCC	$\frac{N^2}{\log_2^2 N}$	$\frac{N}{2 \log_2 N}$	$2 \log_2 N$	$\frac{N}{2 \log_2 N}$	$2 \log_2 N$	
Shuffle exch.	$\frac{N^2}{\log_2^2 N}$	$\frac{N}{2 \log_2 N}$	$2 \log_2 N - 1$	$\frac{N}{2 \log_2 N}$	$2 \log_2 N$	

Table 1: Some properties of layouts of in the plane.

regardless of the tree size.

7 A layout comparison

Some of the network characteristics we have derived are summarized in Table 1.

Of the networks considered so far, the binary cube has both a small diameter and a high bisection width resulting in good lower bounds whether based on the diameter of the graph, or the bisection width. These bounds are both important for operations such as sorting. But, the area required for the binary cube is high, indeed highest of all networks that we considered.

The butterfly network, the cube-connected-cycles network and the shuffle-exchange network all have the diameter that only is twice that of the binary cube, and a bisection width that is a factor of \log_2 less than that of the binary cube. But, the area required is a factor of $\log_2^2 N$ less.

In VLSI technology, the cost increases very rapidly with the chip area due to reduced yield and

fewer chips per wafer. What is on the chip does not have any significant impact on its cost. And, the chip area is largely determined by wiring needs. Transistors only account for a few percent of the total area. The cost of printed circuit boards also increase with the area, though not as dramatically as for chips. However, the cost increases at least in proportion to the area.

Area or volume are good indicators of the cost of a computing system. Hence, an interesting question to ask which network yields the best performance for a fixed area (cost)? To answer this question faithfully, it is necessary to not only account for the difference in the number of nodes that can fit in a fixed area for different networks, but also how wire lengths and partitioning the system into parts with a fixed, limited perimeter affects the performance. For simplicity, we will assume here that all channels are equally fast, and ignore the effects of partitioning, though we will comment on it later.

Which network will be the fastest also depends upon the computations to be performed. For instance, assume that we are choosing between building a two-dimensional mesh or a binary cube and the main task is to solve Poisson's equation in two dimensions with Jacobi's method, as described before. Then, according to the Thompson grid point model and our layout results, we can construct a mesh with $O(A)$ nodes, or a binary cube with $O(\sqrt{A})$ nodes in area A . Then, each binary cube node must emulate \sqrt{A} nodes of the mesh. With respect to computation time, the binary cube is a factor of \sqrt{A} slower. With respect to communication time, for a fixed area the width of each of the channels of the cube is a factor of $A^{\frac{1}{4}}$ narrower than that of the mesh. If the subgrid for the mesh machine is of shape $M \times M$, then the subgrid for the binary cube is of shape $MA^{\frac{1}{4}} \times MA^{\frac{1}{4}}$. Communicating the boundary of the local mesh requires a time that is $A^{\frac{1}{4}} \times A^{\frac{1}{4}} = A^{\frac{1}{2}}$ longer in the binary cube case, assuming that only a single channel between a pair of nodes is used to emulate the mesh connection. Using all channels of the binary cube will improve the communication time by at most a factor of $\log_2 A^{\frac{1}{4}}$.

Thus, even though the mesh can be embedded in the binary cube preserving adjacency (using a Gray code), the binary cube is slower than the mesh by a factor of \sqrt{A} both with respect to computation and communication. In addition, the mesh only has short wires, while some of the binary cube wires are long, and may reduce the performance further.

Emulating a two-dimensional mesh on a binary cube and comparing the performance to that of a two-dimensional mesh is unfair to the binary cube. It cannot be expected that a computation for which a two-dimensional mesh is optimal will perform as well on a binary cube. Thus, let us instead consider sorting. Then, the mesh may require a time proportional to \sqrt{A} . The constant of proportionality depends upon whether the diameter or the bisection width determines the time in the case of one element per node. With $P > A$ elements distributed evenly, and the communication time determined by the bisection width, the mesh may require a time proportional to $\frac{P}{2\sqrt{A}}$. The binary cube will require a time of $\frac{P}{2(\sqrt{A}/2)} = \frac{P}{\sqrt{A}}$, since there are \sqrt{A} nodes in area A , and hence the bisection width is $\frac{1}{2}\sqrt{A}$. Thus, in this case, the communication time for the binary cube and the mesh are of the same order, with the lower bound for the mesh being slightly better. With respect to comparisons, each binary cube node must perform the work of \sqrt{A} mesh nodes. Hence, for sorting, if the communication time dominates then the binary cube may be faster if the diameter determines the communication time, while if the bisection width is the determining factor, then mesh may have a slight advantage. On the

other hand, of the comparison time determines the performance, the mesh is expected to have an advantage by a factor of \sqrt{A} , using straight emulation techniques.

The area expense for the binary cube network is in many situations not paying off in performance for a fixed area (cost). Hence, let us consider a butterfly network instead of the binary cube. Then, in area A we can lay out a network with $\sqrt{A} \log_2 A$ nodes. The bisection width is $\frac{1}{2}\sqrt{A}$ as for the binary cube. Computations limited by the bisection width would still require time $\frac{P}{\sqrt{A}}$. Compared to the mesh, each node in the butterfly network must emulate $\frac{\sqrt{A}}{\log_2 A}$ nodes. Thus, we again find that the butterfly network, though less area consuming does not offer any advantage over the mesh, unless the computation time is determined by the diameter.

Meshes cannot be embedded in butterfly networks with constant dilation. With respect to computation, the butterfly network must emulate $O(\frac{\sqrt{A}}{\log_2 A})$ mesh nodes.

In these examples we see that if the area is the same for a mesh, a binary cube and a butterfly network, then the time for operations that are bandwidth limited is of the same order, while the mesh is faster if the desired computation indeed is that of a mesh emulation. The speed advantage of the mesh may be as much as \sqrt{A} for the binary cube, and $\frac{\sqrt{A}}{\log_2 A}$ for the butterfly network. And again, the mesh only has short wires.

But, a mesh may be considerably slower than the other networks when the diameter of the network is the limiting factor. The diameter may be the factor determining the speed in a lightly loaded network.

The problems with the networks we have discussed so far is that the networks with a diameter of order $O(\log_2 N)$ have a relatively large area requirement, or a bisection width that is somewhat too large for the number of nodes in the network. On the other hand, the complete binary tree has desirable area requirements, but the bisection width is too small. It is desirable that it is at least as large as for the mesh.

Does there exist any N node networks with diameter $O(\log_2 N)$, area $O(N)$ and a bisection width of order $\frac{\sqrt{N}}{\log_2 N}$? The area of such a network must be at least $N/\log_2^2 N$. Since the bisection width is a factor of $\log_2 A$ less than the maximum possible, computations for which the bisection width determines the performance cannot be suboptimal by more than a factor of $O(\log_2 A)$. Since we have $O(N)$ nodes in area A , the computational slowdown is at most a constant factor.

Networks that can simulate any other network with at most a logarithmic slowdown $O(\log_2 A)$ are known as *area universal networks*.

8 Area Universal networks

Fat-tree networks can be made area universal [5]. We have earlier shown both a binary fat-tree in which the channel capacity doubled for each level proceeding from the leaves to the root, and a 4-ary fat-tree where the bandwidth was reduced by a factor of two for each level, Figure 17.

The binary fat-tree in Figure 16 cannot be laid out in area $O(N)$, since the bisection width

is $O(N)$. Thus, to achieve our goal of area universality we must reduce the total bandwidth between levels as we move from the leafs towards the root. We will now show that an N node 4-ary fat-tree can be laid out in area $O(N)$, just as a complete binary tree can be laid out in area $O(N)$. We will also show that the bisection width is $\frac{\sqrt{N}}{\log_2 N}$.

In the 4-ary fat-tree each node has four children. Each node in the 4-ary fat tree must be capable to route a message from any of its children to any other child node. Similarly, it must be possible to route a message from any child to a nodes parent. Since, in general, there are several channels between any pair of nodes, it is also necessary that the load can be balanced among the channels between a pair of nodes. Thus, each internal node of the fat-tree must possess good switching capabilities, though a full crossbar switch is not necessary. In the 4-ary fat-tree in Figure 17, each leaf node has two connections to its parent node, while a parent of the leaf nodes has a total of eight connections going to children nodes. The parent nodes of the leaf nodes have four connections to their parents, which in turn have 16 connections to their children. The number of connections between a node and its parent doubles for every level of the tree.

Each node in our 4-ary fat-tree consists of one stage of a butterfly network. Each leaf node consists of a 2-input 2-output butterfly stage. In the level above the leaf nodes, each node consists of a 4-input 4-output butterfly stage. In the next level up, each node consists of an 8-input 8-output butterfly stage. Thus, the number of butterfly inputs and outputs per node doubles for every level of the 4-ary tree. The 4-ary fat-tree in Figure 17 can be viewed as a *slender* butterfly network.

A k level 4-ary tree has 4^k leaf nodes. An N leaf node 4-ary tree has height $k = \log_4 N$. The number of nodes in such a tree is $(4N - 1)/3$. The length of a path from the root to a leaf is $\log_4 N$. In our 4-ary fat-tree, the leaf nodes contains four butterfly nodes, the next level of nodes contains eight butterfly nodes, the next 16 butterfly nodes, etc. Thus, in our 4-ary fat-tree there is a total of $4 \cdot 4^k$ butterfly nodes at the leaf level. At the level above, there are 4^{k-1} tree nodes, each of which has 8 butterfly nodes, for a total of $8 \cdot 4^{k-1}$ butterfly nodes. The total number of butterfly nodes is

$$4 \cdot 4^k + 4 \cdot 2 \cdot 4^{k-1} + 4 \cdot 2^2 \cdot 4^{k-2} + \dots 4 \cdot 2^k = 4^{k+1} \left(2 - \frac{1}{2^k}\right).$$

With each tree node containing one butterfly stage, the path length from the tree root to a tree leaf node is $2k - 1$, since one link must be traversed within each internal tree node. With respect to butterfly nodes, the maximum distance form root to leaf is $2k + 1$. The diameter of the fat-tree is $\sim 4 \log_4 N = 8 \log_2 N$.

The number of input nodes at the tree root is $2 \cdot 2^k = 2\sqrt{N}$. The bisection width is the same, i.e., $2\sqrt{N}$. A lower bound on the area is $2N$.

The fat-tree can be laid out in an H-tree like manner. Each node in the H-tree represents a 2×2 butterfly network switch. Each such switch can be laid out in on a 5×2 subgrid using the Thompson grid model. At the lowest level, four such switches connect to two 2×2 switches at the level above. The two 2×2 switches are obtained through a simple permutation of the columns in an internal tree node as drawn in Figure 17. At the next level, four sets

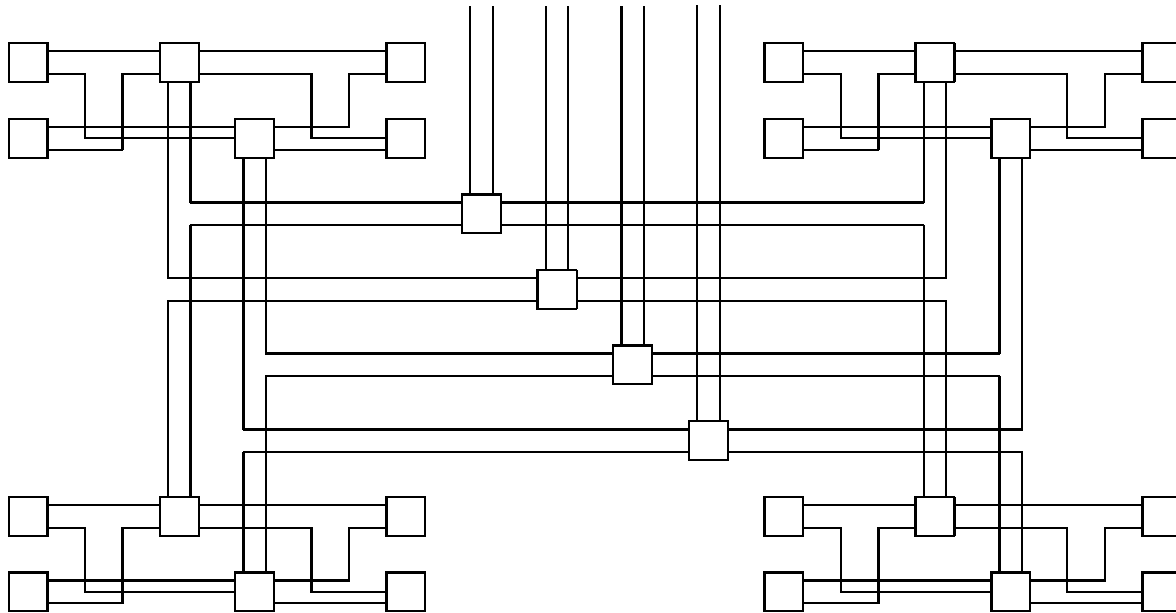


Figure 18: H-tree layout of the 4-ary fat-tree.

each containing two 2×2 switches connect to four nodes of 2×2 switches, then four sets each containing four 2×2 switches connects to eight 2×2 switches, etc. A fat-tree layout is shown in Figure 18. Each of the boxes in this figure represent a 2×2 butterfly switch. To arrive at this layout from Figure 17, the nodes at each level of the fat-tree can be labeled from 0 in a left-to-right order. Then, reorder the nodes within a level by bit-reversing the node index.

The width of the H-tree layout for the fat-tree is

$$w(N) = 2w(N/4) + O(\sqrt{N})$$

This recurrence has a solution of the form $w(N) = \sqrt{N} \log_2 N + \sqrt{N} w(1)$. Now, let each leaf node in fact consist of a $\log_2 N \times \log_2 N$ array. Then $w(1) = \log_2 N$, and the area is $A = O(N \log_2^2 N)$, which is of the same order as the total number of nodes. The fat-tree had $O(N)$ leaf nodes before adding the $\log_2 N \times \log_2 N$ arrays at the leafs. Adding the arrays increases the diameter by a constant factor. Thus, our H-tree layout of the fat-tree is area optimal.

What is the maximum wire length?

As for the emulation capacity, the maximum bisection width of any other network in area $O(N \log_2^2 N)$ is at most $O(\sqrt{N} \log_2 N)$. The bisection width of the fat-tree is \sqrt{N} . Thus, the slow down caused by the emulation on the fat-tree is at most $O(\log_2 N)$.

We earlier showed that a two-dimensional mesh, a binary cube, and a butterfly network could perform bisection width limited computations in time $O(\sqrt{A})$ in area A . The fat-tree can perform the same computations in a time of at most $O(\sqrt{A} \log_2 A)$, i.e., a slowdown by a factor of at most $O(\log_2 A)$. Compared to the complete binary tree, the fat-tree offers a speedup of

$O(\frac{\sqrt{A}}{\log_2 A})$ for computations limited by the bisection width.

We note a few properties of the fat-tree:

- A k -level fat-tree contains a $2^{k+1}(k+2)$ node butterfly network as a subgraph.
- A k -level fat-tree contains a $2k+1$ -level complete binary tree from any of its root nodes.
- There exist a unique path from any leaf node to any root node. Making random choices on the path to the root will select the root randomly. The path to another leaf is fully determined once the root is chosen.

How should a fat-tree be partitioned with respect to minimizing the number of edges cut?

The fat-tree used in the Connection Machine system CM-5 is described in [6].

9 Partitioning

One desirable characteristic of a network is modularity in the sense that big networks can be built from smaller networks, and that the smaller parts are independent of the size of the larger network. Meshes and butterfly networks have this property, while binary cubes do not. The largest network to be built has to be anticipated at design time, and built into the subcubes. Binary trees of arbitrary size can be constructed from parts with a constant number of connectors, regardless of the tree size constructed and the subtree on the part.

Another important issue is the channel width. With a fixed perimeter of a chip, or a multichip carrier, or a board edge, a large number of channels cut implies that each of the channels becomes narrower the larger the number of channels cut. In [9] we in show that in two networks of the same area, the same number of nodes, but different channel widths for a fixed bisection width, the mesh and the binary cube can emulate a butterfly network in the same time, if the dependence upon the wire length is ignored. Factoring in the dependence upon the wire length results in a speed advantage for the grid by a factor of $O(\log_2 A)$ or $O(\sqrt{A})$. The first factor applies for a logarithmic time model for the channel transfer time as a function of its length. The second factor applies for a linear time dependence.

References

- [1] Sandeep N. Bhatt and Charles E. Leiserson. How to assemble tree machines. In *Fourteenth Annual ACM Symposium on the Theory of Computing*, 1982.
- [2] Richard P. Brent and H.T. Kung. On the area of binary tree layouts. *Information Processing Letters*, 11(1):44–46, 1980.
- [3] John L. Hennessy and David A. Patterson. *Computer Architecture: A Quantative Approach*. Morgan Kaufmann Publishers, Inc, 1990.

- [4] F. Thomson Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. Morgan Kaufmann, 1992.
- [5] Charles E. Leiserson. Fat-trees: Universal networks for hardware-efficient supercomputing. *IEEE Trans. Computers*, 34:892–901, October 1985.
- [6] Charles E. Leiserson, Zahi S. Abuhamdeh, David C. Douglas, Cral R. Feynman, Mahesh N. Ganmukhi, Jeffrey V. Hill, W. Daniel Hillis, Bradley C. Kuszmaul, Margaret A St Pierre, David S. Wells, Monica C. Wong, Shaw-Wen Yang, and Robert Zak. The network architecture of the Connection Machine CM-5. In *SPAA '92*, pages 272–285. ACM Press, 1992.
- [7] Carver A. Mead and Lynn Conway. *Introduction to VLSI Systems*. Addison-Wesley, 1980.
- [8] M.S. Paterson, W.L. Ruzzo, and Larry Snyder. Bounds on minimax edge length for complete binary trees. In *Proc. of the 13th Annual Symposium on the Theory of Computing*, pages 293–299. ACM, 1981.
- [9] Abhiram Ranade and S. Lennart Johnsson. The communication efficiency of meshes, Boolean cubes, and cube connected cycles for wafer scale integration. In *1987 International Conf. on Parallel Processing*, pages 479–482. IEEE Computer Society, 1987.
- [10] Snyder L. Ruzzo W.L. Minimum edge length planar embeddings of trees. In *VLSI Systems and Computations*, pages 119–123. Computer Sciences Press, 1981.
- [11] C.D. Thompson. A complexity theory for VLSI. Technical report, Dept. of Computer Science, Carnegie-Mellon Univ., 1980.
- [12] Jeffrey D. Ullman. *Computational Aspects of VLSI*. Computer Sciences Press, 1984.
- [13] Andrew C. Yao. The entropic limitations of vlsi computations. In *The Thirteenth Annual ACM Symposium on the Theorey of Computation*, pages 308–311. ACM, 1981.