The Fat-Pyramid and Universal Parallel Computation Independent of Wire Delay

Ronald I. Greenberg*
Department of Electrical Engineering
and Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742
rig@umiacs.umd.edu

IEEE Trans. Computers, 43(12):1358–1364, December 1994

Abstract

This paper shows that a *fat-pyramid* of area $\Theta(A)$ requires only $O(\log A)$ slowdown to simulate any competing network of area A under very general conditions. The result holds regardless of the processor size (amount of attached memory) and number of processors in the competing network as long as the limitation on total area is met. Furthermore, the result is valid regardless of the relationship between wire length and wire delay. We especially focus on elimination of the common simplifying assumption that unit time suffices to traverse a wire regardless of its length, since the assumption becomes more and more untenable as the size of parallel systems increases. This paper concentrates on simulation using transmission lines (wires along which bits can be pipelined) with the message routing schedule set up off line, but it also discusses the extension to on-line simulation. This paper also examines the capabilities of a fat-pyramid when matched against a substantially larger network and points out the surprising difficulty of doing such a comparison without the unit wire delay assumption.

1 Introduction

This paper shows that the *fat-pyramid* network is a good candidate as the basis for a general-purpose parallel computer, because it can efficiently simulate any network of comparable area under general conditions. The basic structure of the fat-pyramid network was suggested by Charles Leiserson and Tom Cormen and is related to the fat-tree introduced by Leiserson [18]. The fat-pyramid may be viewed as a fat-tree in the style introduced in [12, Sec. 7] (the *butterfly fat-tree*) augmented by hierarchical mesh connections as illustrated in Fig. 1. (A variation on the butterfly-fat-tree has recently been adopted for the network structure of the CM-5 parallel computer of Thinking Machines Corporation [20].)

Ignoring the mesh connections, shown with thick lines, the fat-pyramid may be viewed as based upon a 4-ary tree in which each internal node is replaced by a collection of switches and processors are placed at the leaves. A collection of wires corresponding to an edge in the underlying 4-ary tree is referred to as a channel, and the number of wires in a channel is called its capacity. By using two types of switches with a constant number of inputs and outputs, it is possible to build fat-pyramids with essentially arbitrary channel capacities as illustrated in [19]. In this paper it suffices to make only a simple modification to the network shown in Fig. 1, in which channel capacities double at each level of the 4-ary tree.[1] A precise mathematical

*Supported in part by NSF grant CCR-9109550.

[1] The choice of the name "fat-pyramid" for this network stems from the observation that if all channel capacities were equal to one, the result would be a network which has been called the "pyramid" by Tanimoto (and earlier a "recognition cone" by

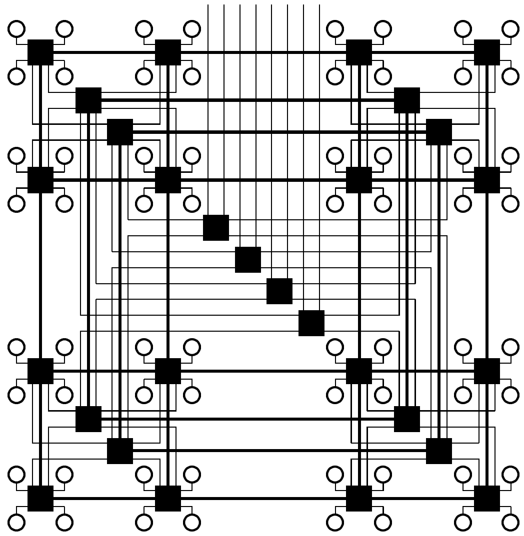

Figure 1: A fat-pyramid. Processors are placed at the leaves, represented by circles; the squares are switches. This network is obtained by superposing hierarchical mesh connections on a butterfly fat-tree. The original fat-tree connections are represented by thin lines and the mesh connections by thick lines. (A different layout of the fat-pyramid is used to obtain results independent of wire delay.)

description of the interconnection pattern for the switches in Fig. 1 can be given as follows. We begin with a collection of ordinary two-dimensional meshes at levels $0, 1, \ldots, \log_2 \sqrt{n/4}$ representing distance from the leaves in the underlying tree structure. At level h, there are 2^h copies of a $\sqrt{n/4}/2^h \times \sqrt{n/4}/2^h$ mesh. We then denote a switch by an ordered 4-tuple (h, c, x, y), where h is the level, c is the copy number of the mesh $(0 \leq c < 2^h)$ at this level that contains the switch, and x and y specify a mesh position in an ordinary Cartesian coordinate system $(0 \leq x, y < \sqrt{n/4}/2^h)$. Then for $0 \leq h < \log_2 \sqrt{n/4}$, switch (h, c, x, y) is connected to $(h+1, c, \lfloor x/2 \rfloor, \lfloor y/2 \rfloor)$ and $(h+1, c+2^h, \lfloor x/2 \rfloor, \lfloor y/2 \rfloor)$.

An important feature of the fat-pyramid is its status as a universal network without the usual assumption that unit delay suffices to traverse a wire of any length. This issue becomes more and more important as we move towards increasingly massive parallelism. In this regard, the fat-pyramid has an advantage over the fat-tree, though these networks are both appropriate to a view in which hardware cost is measured as area or volume under standard VLSI models. By accounting for the difficulty of running the wires in a massive network, these VLSI models provide a more realistic measure of cost than merely bounding processor count and node degree. Though the results in this paper are stated in terms of area in a two-dimensional design space (constant number of chip layers), the extension to three-dimensions is fairly straightforward [9] using the ideas in [11].

The basic mode of operation assumed for fat-pyramids and any other parallel computer will be as in the distributed random-access machine (DRAM) model of Leiserson and Maggs [21]. All memory is local to the processors, and a processor can read, write, and perform arithmetic and logical functions on values stored in its local memory. It can also read and write memory in other processors by routing messages through an underlying network. In the bulk of this paper, messages are viewed as being composed of indivisible "packets", and delay along wires is measured in terms of the time to transmit a complete packet; the end of Section 3 explains why there is limited change to the results if we switch from this "word model" to a "bit model". There is also no need to worry about messages having varying length; we can think of messages as being divided up into packets of standard size and henceforth treat "message" as synonymous with "packet".

Uhr) [25, p. 3]. The addition of mesh connections to the fat-tree is also similar to the introduction of "brother" connections in trees to obtain the X-Tree network [8, 23]. (The term "fat-pyramid" should not be confused with a recent independent use of the term by other authors.)

It is also convenient to assume that operation of competing networks is divided into separate (alternating) phases of intraprocessor computation and interprocessor communication. Thus, to bound asymptotic simulation time, it will suffice to take the maximum of the overheads for simulation of the computation and simulation of the communication. In fact, this approach is valid even if the competing network interleaves computation and communication in a more complicated fashion. The validity of the simplification is established rigorously in [9, Sec. 4.5] in the case of unit wire delay; the case of nonunit delay can be handled by a similar approach of prioritizing instructions and packet deliveries according to their completion times in the competing network. Throughout this paper, we shall say that network A can simulate network B with overhead μ if, for any t, the operations performed by network B in time t can be performed by network A in time $t\mu$.

Intuitively, the fat-pyramid combines the strengths of the fat-tree and the mesh. A fat-tree can efficiently simulate any network of comparable area under the unit wire delay assumption [3, 12, 14, 18]. But the straightforward layout of the fat-tree has wires of length $\Theta(\sqrt{A})$ near the root (and little improvement is possible). If the fat-tree is used to simulate a mesh, any mapping of processors in the mesh to processors in the fat-tree will place on "opposite sides of the root" some processor pairs that are adjacent in the mesh. If the time to transmit a packet is a linear function of wire length, the fat-tree will require $\Omega(\sqrt{A})$ time to route messages between such a pair of processors. But the mesh network could be performing a computation requiring only nearest neighbor communication so that the fat-tree simulation would have polynomial ($\Omega(\sqrt{A})$) overhead, which is much worse than the polylogarithmic overhead attainable in the unit wire delay case. The mesh, on the other hand, is universal in the case of linear wire delay. But if delay is less sensitive to wire length, the mesh may also suffer polynomial slowdown as can be seen by considering simulation of a tree. Since a tree of area A (using the H-tree layout illustrated in e.g. [17]) contains essentially the same number of processors as a mesh of area A, the mapping of tree processors to the mesh will expand some routing path between processors from $O(\log A)$ switches in the tree to $\Theta(\sqrt{A})$ in the mesh. The fat-pyramid, in contrast to the fat-tree or the mesh, can achieve polylogarithmic simulation overhead under essentially any interesting model of wire delay.

The remainder of this paper is organized as follows. Section 2 discusses the fat-tree variation used as the basis for the fat-pyramid and its ability to efficiently simulate any network of comparable area given unit wire delay. Section 3 shows how the fat-pyramid can be used to extend the ideas of Section 2 to nonunit wire delay. Section 4 considers the overhead required by a universal network to simulate a network of larger area; these results are proved only for unit wire delay. Section 5 contains concluding remarks and open questions.

2 Routing and simulation overhead on the fat-tree

To facilitate explanation of the universality properties of the fat-pyramid, we first examine details and properties of the particular variation of the fat-tree used as its basis in this paper. In this section, we retain the assumption of unit wire delay.

We begin by considering the area requirements of the fat-tree. A standard modeling approach involves thinking of network nodes as points in a grid and wires as edge-disjoint paths through the grid, but it is somewhat unrealistic to view the network nodes as occupying constant area. Since we generally want each processor to be capable of addressing each other processor, a fat-tree of area A should have $\Omega(\log A)$ memory per processor.[2]

In order to pack in a maximum number of processors of area $\Theta(\log A)$, we consider a fat-tree with channel capacities doubling at increasing levels of the underlying 4-ary tree as in Fig. 1, except that each circle in that figure is replaced by an H-tree layout of $\log_2 A$ processors. Each such H-tree block is of size $\Theta(\log A) \times \Theta(\log A)$, and we can derive the area of the fat-tree by solving a recurrence relation for the side

[2]The requirement is actually $\Omega(\log n)$, where n is the number of processors. But $\log n$ is $\Omega(\log A)$ unless n is $o(A^\epsilon)$ for every $\epsilon > 0$, which would imply that competing networks of the same area could not be simulated in reasonable time since they could have $\omega(A^{1-\epsilon})$ times as many processors. Comments preceding Theorem 2 and in the beginning of the proof of Lemma 3 provide some justification for assuming henceforth that processors occupy a square of area $\Theta(\log A)$; if more area is required to include sophisticated operations in the processor instruction set, the situation can be handled as in [9, 10].

length $S(b)$ of a fat-tree with b H-tree blocks. Since the channel capacities above the H-tree blocks double at every level, we have a channel capacity proportional to $\sqrt{b}$ at the root, and we can write the recurrence

$$S(b) = 2S(b/4) + O(\sqrt{b}) \text{ , and } S(1) = \Theta(\log A) \text{ ,}$$

with solution

$$S(b) = \Theta(\sqrt{b} \log A) \text{ .}$$

Thus, we can build a fat-tree on $A/\log_2 A$ processors of area $\Theta(\log A)$ in area $\Theta(A)$ since that corresponds to b=$A/\log_2^2 A$. (Two notes are in order. First, the recurrence assumes switches of constant area, but switches of area $\Theta(\log A)$ to examine and compare full processor addresses or message priorities can be accommodated in the modified layout discussed in Section 3. Second, because of the need to precisely bound the wire density, we should think of packet transmission as occurring in a bit-serial fashion. We can still think of unit time as the time to transmit a packet of $\Theta(\log A)$ bits along a wire.)

Now we can examine the simulation overhead required by a fat-tree of area A to simulate other networks of area A. We can perform this comparison without placing any restriction on the number or size of processors of the competing network. Here, processor size refers solely to the amount of attached memory; we assume that processors of networks to be compared have the same instruction set and are equally well-engineered to provide the same operations at the same cost in time and space. If the competing network has larger processors than the fat-tree, we can subdivide the memory of these processors and view them as a larger number of smaller processors, so we concentrate on the situation in which the processors of the competing network are no larger than those of the fat-tree. We can use a straightforward geometric mapping of processors of the competing network to fat-tree processors and powerful packet routing results [15, 16] to show that the fat-tree is universal under unit wire delay. For now, only the off-line result is considered, i.e., that there exists an efficient means of scheduling the messages of competing networks on the fat-tree. For this result, the following packet routing lemma suffices; the term *congestion* refers to the maximum number of packets that must cross a single edge (wire) in the network, and *dilation* refers to the maximum number of edges that must be traversed by a single packet.

Lemma 1 ([15]) *For any set of packets with edge-simple paths having congestion c and dilation d, there is a schedule having length $O(c+d)$ and maximum queue size $O(1)$ in which at most one packet traverses each edge of the network at each step.*

We can now proceed with the fat-tree universality result:

Theorem 2 *A fat-tree F of area $\Theta(A)$ can simulate any network of area A with $O(\log A)$ overhead.*

Proof. Given a competing network R of area A, we recursively bisect it in the straightforward geometric fashion, cutting nonsquare pieces in the shorter direction, until we have $A/\log_2 A$ pieces. These pieces of R are mapped to the $n = A/\log_2 A$ processors of F so that the recursive bisection of R matches the natural recursive bisection of F obtained by cutting at the root and then at the roots of the subtrees and so on. This yields at most $\log_2 A$ processors of R whose computation must be simulated by a single processor of F.[3]

Having obtained a computation overhead of $O(\log A)$, we now analyze the communication overhead involved in routing messages of R through F. The message traffic in channels of F can be bounded by using an assumption that is certainly reasonable for VLSI technologies: the number of packets that can leave an area in unit (packet transmission) time is proportional to the perimeter of the area. The perimeter of a piece of R corresponding to a subtree of $n/2^l$ processors in F is $O(\sqrt{A}/2^{l/2})$ (excluding the perimeter of R itself if R is nonsquare). Since the capacity of a channel on top of a subtree of $n/2^l$ processors in F is at least $\sqrt{(n/\log_2 A)/2^l}$ and $n = A/\log_2 A$, the number of packets entering or leaving a subtree in unit time divided

[3] A processor that is split by one or more of the bisection lines in R can be mapped to any one of the processors in F to which its pieces would correspond. This does not alter the bounds on computation or communication overhead, because a bisection line can only split a number of processors equal to its length and because the extra number of messages generated by the split processors in unit time is at most proportional to the number of such processors.

by the number of wires in the corresponding channel is $O(\log A)$. In fact, the packets can be routed through F so that there are $O(\log A)$ packets on any single wire. Indeed, this outcome occurs with high probability if each message is routed by picking at random the root switch it should pass through (which fully determines its path) [14, 16].

We have now established that for a set of packets traveling in R during any unit period of time, at most $O(\log A)$ packets must traverse any given wire in F. Furthermore each packet must traverse at most $O(\log A)$ wires to get from its source to its destination in F. In other words, the congestion and dilation are at most $O(\log A)$. Thus, by Lemma 1, in $O(\log A)$ time steps, we can completely route in F all the messages that travel in R during any unit of time. ■

3 The fat-pyramid and nonunit wire delay

In this section we consider the effect of dropping the unit wire delay assumption. The general graph layout framework developed by Bhatt and Leighton [6] shows that there is enough room in our fat-tree layouts to build sufficiently large drivers for each wire to keep the wire delay constant in the capacitive model. This section shows that even if this constant switching time is not the dominant determiner of wire delay, an appropriate layout of the fat-pyramid yields a universal network with $O(\log A)$ simulation overhead. The key to this result is that a routing path of length δ in a competing network of area A corresponds to a path of length $O(\delta + \log A)$ in the fat-pyramid.

It should be noted that it is reasonable to assume wire delay to be no worse than linear in wire length, since repeaters (extra switches) can always be used to reduce delay to linear. Linear wire delay would be the correct model if technology could be improved to the point where only speed of light limitations constrain the time to switch a length of wire. Then, the measure of unit time would be much smaller, but linear wire delay would be required of any competing network.

It is also helpful to assume a mild "regularity" condition on the wire delay function. (Similar regularity conditions are used elsewhere in the literature (e.g., [5, 7, 17],[1, p. 280]) in order to obtain results about large classes of functions.) Specifically, let $w(\delta)$ denote the time required to transmit a packet along a wire of length δ; then we seek two properties for the function w. First, w should be nondecreasing, and second it should satisfy the following condition:

Definition: A function w is said to satisfy Condition C1 if there exists a constant c such that

$$\frac{w(\delta + x)}{w(\delta)} \leq \frac{\log_2 \delta + x}{\log_2 \delta}$$

for all $x \geq 0$ and $\delta \geq c$.

It should be noted that Condition C1 is satisfied by most functions likely to be of interest in the context of wire delay. For example, it is satisfied by all functions $w(\delta)$ of the form $c\delta^q \log_2^k \delta$ for constants c, q, and k such that either $q < 1$, or $q = 1$ and $k \leq 0$. One way to see that all of these functions satisfy Condition C1, is to observe that they satisfy a simpler regularity condition C2, which implies C1.

Definition: A function w is said to satisfy Condition C2 if there exists a constant c such that

$$\frac{w(\delta + x)}{w(\delta)} \leq \frac{\delta + x}{\delta}$$

for all $x \geq 0$ and $\delta \geq c$.

Condition C2 implies condition C1 because $1 + x/\delta \leq 1 + x/\log_2 \delta$ for any $x \geq 0$ and $\delta > 0$.

(Without changing the asymptotic results given below, we can actually weaken conditions C1 and C2 in order to admit an even larger class of functions than already mentioned. Specifically we could define conditions C1 and C2 to be that the old conditions are satisfied to within a constant factor. Then the

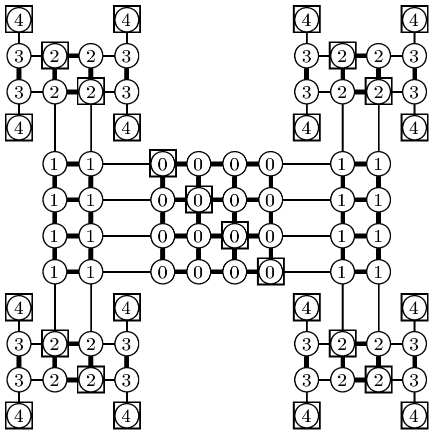

Figure 2: Embedding the fat-pyramid in the tree of meshes. The squares represent switches of the fat-pyramid; laying out the edges connecting switches in the butterfly fat-tree requires using each edge of the tree of meshes at most twice.

conditions are satisfied by any function w satisfying $c_1\delta^q \log_2^k \delta \leq w(\delta) \leq c_2\delta^q \log_2^k \delta$ for sufficiently large δ, with q and k as before and positive constants c_1 and c_2.)

To obtain results independent of wire delay, we must consider a regular layout of a fat-pyramid, that is, one in which the switches at any given level of the tree are regularly spaced throughout the layout. We can produce such a layout by using the "fold-and-squash" technique of Bhatt and Leighton [6, pp. 325–326] and Thompson [26, pp. 36–38]. To see the effect on wire lengths in the fat-pyramid, it is helpful to think of embedding the butterfly fat-tree into the tree of meshes graph, performing the fold-and-squash transformation, and then adding the fat-pyramid's hierarchical mesh connections. We will actually embed just the switches shown as black squares in Fig. 1, while keeping in mind that each such bottom-level switch must be attached in the final layout to four $\Theta(\log A) \times \Theta(\log A)$ H-trees composed of $\log_2 A$ processors of area $\Theta(\log A)$. Fig. 2 illustrates the embedding of switches of the fat-tree into a 4×4 tree of meshes. The fold-and-squash transformation of the tree of meshes is performed by first folding the connections between the mesh at level zero (comprised of all the nodes labeled zero in Fig. 2) and the meshes at level one so that the two smaller meshes fit on a second layer directly over the large mesh. Then the meshes at level two are folded onto a third layer and so forth up to $\log_2(A/\log_2^2 A) = O(\log A)$ levels. Finally, the layers are offset and projected onto the plane as illustrated in Fig. 3. The maximum length of tree-of-meshes edges becomes $O(\log A)$ even with the four $\Theta(\log A) \times \Theta(\log A)$ H-trees clustered around the bottom-level switches of the fat-tree.[4] Finally, only a constant factor increase in area is required to add the fat-pyramid's hierarchical mesh connections. (In addition, since the switches at any given level are separated by a distance of $\Theta(\log A)$, there is room to expand the switches to occupy area $\Theta(\log A)$; the layout can also be massaged to make such switches square if desired.)

In the regular layout of a fat-pyramid of area A, wires connected to a switch h levels up from the H-tree blocks in the underlying 4-ary tree, are of length $O(2^h \log A)$. To see this, observe that each edge of the fat-pyramid that is h levels up is mapped to a path of $O(2^h)$ tree-of-meshes edges.

Messages in the fat-pyramid are routed over the same paths as in the fat-tree, except that we allow each message to take one shortcut via one or two of the new hierarchical mesh edges. More precisely, the routing path is formed by going up fat-tree edges towards a randomly selected switch at the root of the underlying 4-ary tree until a switch is reached that is adjacent horizontally, vertically, or diagonally in the mesh at that level to a switch from which the destination can be reached by going down fat-tree edges. Certainly the dilation is not increased by incorporating such shortcuts, and, in fact, the congestion for any set of messages in the fat-pyramid is also within a constant factor of the congestion in the fat-tree. The congestion in tree

[4] An explanation at greater length of the fold-and-squash technique can be found in [11].

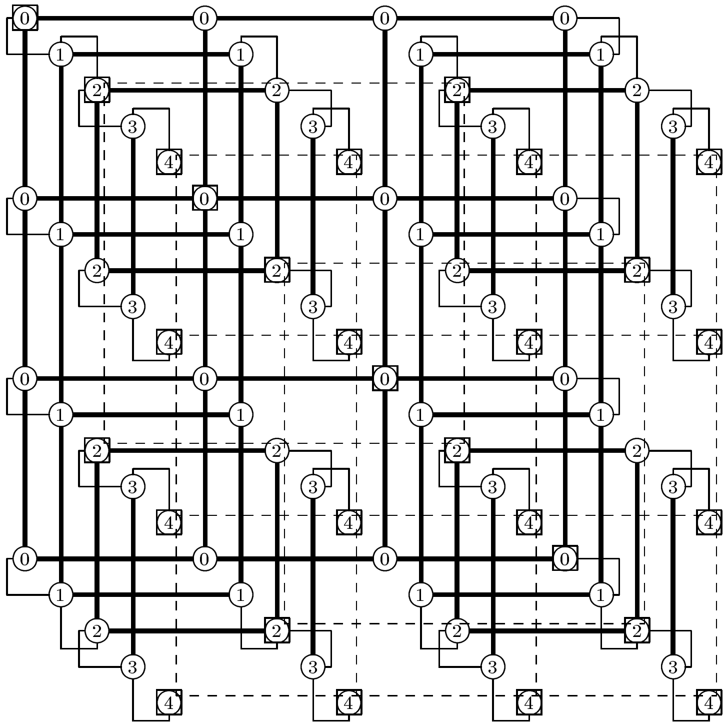

Figure 3: Layout of the fat-pyramid after folding and squashing. As before, the squares represent fat-pyramid switches, and fat-tree edges are routed through the tree of mesh edges shown with thin and thick solid lines. (The connections between the layers obtained through folding, the thin lines, are routed by using an auxiliary row or column located in the direction of the corresponding fold and lying parallel to it.) The insertion of the fat-pyramid's hierarchical mesh connections is illustrated with dashed lines.

edges does not increase, because when a message takes a shortcut, it does not traverse any tree edge it was not already destined to traverse; in each mesh edge, the messages that pass may arrive from only from a constant number of tree edges.

Now we can show that the mapping of competing networks to a fat-pyramid does not stretch any wires by very much.

Lemma 3 *Any network R of area A can be mapped to a fat-pyramid F of area $\Theta(A)$ so that any message following a path of length δ in R travels only $O(\delta + \log A)$ distance in F.*

Proof. First note that we can assume R is square. (If R is not square, it can be converted to a square layout of at most 1.8 times as much area with each wire at most 3 times as long as in the original layout [2].) Now we can map the processors of a square network R to F in a straightforward geometric fashion as in Section 2. Then two processors separated by distance δ in R are at most $\lceil \delta / \log_2 A \rceil$ H-tree blocks apart horizontally or vertically in F. Since two subtrees on $(\lceil \delta / \log_2 A \rceil)^2$ H-tree blocks in the underlying 4-ary tree that are physically adjacent (horizontally, vertically, or diagonally) must suffice to cover such a pair of processors, the routing path connecting these processors needs only to go up $\log_2(\lceil \delta / \log_2 A \rceil)$ levels above the H-trees and use two mesh edges. Since any wire connected to a switch h levels up is of length $O(2^h \log A)$, and the distance traveled within H-tree blocks is $O(\log A)$, the length of the routing path connecting processors at distance δ in R is $O(\delta + \log A)$. ■

We can now state our main result for nonunit wire delay, relying only on our regularity condition for wire delay. As before, we focus on off-line scheduling in the packet model; extensions are discussed afterwards. It is also important to note that the availability of transmission lines is assumed, so that wires can contain a number of packets equal to the delay of the wire at any given time. Though it is natural to think of a transmission line as a device for pipelining bits, it is only more conservative to think of pipelining packets and to continue measuring delay in terms of packet steps. (Use of transmission lines is occurring in real parallel machines, e.g., see the references in [22].)

Theorem 4 *Using transmission lines, a fat-pyramid F of area $\Theta(A)$ can simulate any network of area A with $O(\log A)$ overhead.*

Proof. To apply the packet routing results of Leighton, Maggs, and Rao [15, 16], we can imagine additional switches on each wire of the fat-pyramid in number equal to the delay for that wire. With the inclusion of these imaginary (and trivial) switches, we can view the routing problem as fitting into the unit wire delay framework; we have simply increased the maximum distance (in terms of switches) that packets must travel.

Now consider any set of messages generated by the competing network in which the maximum *physical* distance that a message travels in the competing network is δ. Let T be the time required to deliver the set of messages in the competing network, and note that $T \geq w(\delta)$. Also, the congestion created by this message set is $O(T \log A)$. Furthermore, the maximum number of fat-pyramid edges which a message must traverse is $2 \log_2 \delta$, each containing at most $w(\delta + \log_2 A)$ real and imaginary switches. (Actually, the number of switches should be $w(O(\delta + \log A))$, but the results below remain valid because w is at most linear.) Thus the communication overhead μ can be bounded as follows, based on Leighton, Maggs, and Rao's result that routing can be performed in time proportional to congestion plus dilation:

$$
\begin{aligned}
\mu &\leq O\left(\frac{T \log A + w(\delta + \log_2 A) \log \delta}{T}\right) \\
&\leq O\left(\frac{T \log A}{T}\right) + O\left(\frac{w(\delta + \log_2 A) \log \delta}{w(\delta)}\right) \\
&\leq O(\log A) + O\left(\frac{(\log \delta + \log A) \log \delta}{\log \delta}\right) \\
&\leq O(\log A) \ ,
\end{aligned}
$$

where the third line follows from regularity condition C1. The computation overhead is also $O(\log A)$ as in Section 2. ■

The simulation can also be performed on-line, albeit with some loss of efficiency in the case of nonunit wire delay. In the unit delay case, there is no loss of efficiency, due to analysis of packet routing on a "leveled" network [14, 16]. With nonunit wire delay, however, it is not apparent how to make use of the framework of leveled networks. But the algorithm of Leighton, Maggs, and Rao for routing on-line in $O(c + d\log(Nd))$ steps (with high probability) in general networks, where c is congestion, d is dilation, and N ($\leq A$ here) is the number of packets [15], can be applied to yield a simulation overhead of $O(\log^2 A)$.[5] In fact, the overhead can be improved to $O(\log^2 A/\log\log A)$ using a variation on their technique appearing in [13, 24]. (The question of routing in networks without transmission lines is also considered in [13], and the results there may prove useful to the construction of universal networks without transmission lines.)

Finally, it is desirable to consider the overhead in *bit-times* for on-line routing, since on-line routing schemes may need to tack $\Omega(\log A)$ address bits onto messages that are of constant size in the competing network. The overhead in bit-times for nonunit wire delay is $O(\log^2 A)$, which can be shown as in the proof of Theorem 4 by using the following proof that for unit wire delay, $O((c + d + \log N)\log(Nd))$ bit steps suffice to route packets of $\log_2 N$ bits with high probability $(1 - O(\frac{1}{Nd}))$. We begin by assigning each packet an initial (integral) delay chosen randomly and uniformly from the interval $[1, \alpha c]$, for a constant α to be specified later, and consider the "schedule" in which there is no other waiting except that the bits of any given message are sent one after another. This yields a "schedule" of length $O(c + d + \log N)$ bit-steps in which there may be bits from several different messages traversing an edge at a given time. But the probability that more than $\log_2(Nd)$ packets have a bit traversing a given edge at a given bit-step is at most

$$\binom{c}{\log_2(Nd)}\left(\frac{\log_2 N}{\alpha c}\right)^{\log_2(Nd)} \leq \left(\frac{e}{\alpha}\right)^{\log_2(Nd)} .$$

This probability multiplied by the number of choices for an edge and a bit step is less than $\frac{1}{Nd}$ for a sufficiently large constant α. So with high probability, we can obtain a schedule of length $O((c+d+\log N)\log(Nd))$ in which only one bit traverses an edge at a time, by having switches cycle through incoming packets forwarding one bit at a time.

4 Simulating larger networks

This section obtains upper bounds on the time required by a fat-tree to simulate networks that occupy more area but have the same amount of area devoted to processors. The reason for the latter restriction is that for any significant difference in memory, there are computations which can be performed in the larger amount of memory space but not in the smaller amount of memory space. Rather than placing restrictions on the type of computation, it is probably more meaningful to look at restrictions on the way that space is allocated. That is, if the larger network uses the same amount of processor area (including memory) and simply uses more interconnect area, then we can make meaningful comparisons between the networks. As would be expected, simulation difficulty increases as the area of the competing network does, but only up to a certain threshold beyond which extra area does not help the competition.

In this section we return to a reliance on the unit wire delay assumption due to a change in the means of mapping competing networks to the universal network. The results are, of course, applicable to the fat-pyramid as well as the fat-tree, but it is an open problem to show that unit wire delay is unnecessary when a universal network simulates a larger network.

As we open up the issue of restricting the processor area used by competing networks, it may seem natural to ask about situations in which the competing network has less processor area than is allowed for the universal network. Indeed, we could have considered this question earlier when comparing networks of the same total area. But when the processor area of competing networks is so restricted, the best results are obtained by tailoring the universal network to the particular mix of processor and interconnect area, with the most difficult case occurring when the competing network has no less processor area than the universal

[5] Actually, the universal network should also be modified to have processors that are larger and fewer (both by a factor of $\Theta(\log A)$) in order to accommodate the necessary queues of $O(\log(Nd))$ packets.

network. Thus, the results given so far are worst-case results for simulating networks of essentially the same total area. In this sense, the universal networks described above are the best known to build in a given area. Rather than digress to networks tailored to particular mixes of processors and interconnect, we now ask how well the networks discussed so far can do when they are actually matched against networks with larger area but no greater processor area.

In what follows, we let A_X represent the area of network X. We are, of course, interested in the case where the competing network R has at least as much area as F, i.e., $A_R \geq A_F$; when $A_R \leq A_F$, our earlier results apply.

We use the same basic strategy as before for demonstrating universality results; that is, we recursively bisect the competing network and map appropriate pieces to the fat-tree processors. But when the competing network may have greater area than processor area, extra care is required to ensure that the decomposition is balanced; that is, when we bisect the area of the competing network, we must also bisect the set of processors of the competing network. Fortunately, we can invoke the general theory developed by Bhatt and Leighton [6] and, in a fashion that is cleaner for our purposes, by Leiserson [18]. (It is not desirable to use this approach when unnecessary due to a "loss of locality" in the mapping, which destroys the results on nonunit wire delay in Section 3.) These results tell us that since the competing network of area A_R has a decomposition using cuts of size $\sqrt{A_R}/2^{l/2}$ at level l, it has a balanced decomposition using cuts of the same size (up to a constant factor). Keeping this fact in mind, we can prove the following theorem:

Theorem 5 *A universal fat-tree of area $O(A_F)$ can simulate any network of total area $A_R \geq A_F$ and processor area at most A_F with $O(\sqrt{A_R/A_F}\log A_F)$ overhead.*

Proof. Using a balanced decomposition as described above for the competing network R, we find that the ratio of messages to channel capacity for a set of messages delivered by R in unit time is

$$O\left(\frac{\sqrt{A_R}/2^{l/2}}{\sqrt{A_F/\log^2 A_F}/2^{l/2}}\right)$$

at level l from the root of the fat-tree. Thus, the communication overhead, as determined by congestion plus dilation, is $O(\sqrt{A_R/A_F}\log A_F)$, which dominates the $O(\log A_F)$ computation overhead. ■

When the area of the competing network is much larger than the area of the universal fat-tree, we can actually do better than is suggested by Theorem 5. When A_R is $\Omega(A_F^2)$, the competing network is limited more by the restriction on processor area than by its total area. This is true because communication out of a piece of network R is limited not only by the perimeter of that piece but also by the perimeter of the processors in the piece. Thus, only $O(A_F/2^l)$ messages can leave a piece of R at level l in a balanced decomposition. Dividing by fat-tree channel capacity to determine the congestion, we obtain the following result:

Theorem 6 *A universal fat-tree of area $O(A_F)$ can simulate any network having processor area at most A_F with $O(\sqrt{A_F}\log A_F)$ overhead.* ■

5 Conclusion

This paper has shown that a fat-pyramid network can efficiently simulate any other network built in the same amount of area. The results allow an essentially arbitrary relationship of delay to wire length and allow arbitrary processor size and density in competing networks.

This paper has also obtained bounds on the time required by a universal network to simulate larger networks of the same total processor area. Unfortunately the latter result is not readily extended to the case of nonunit wire delay, due to the use of decomposition trees that are balanced. It is an open question whether or not this extension can be achieved. Perhaps it could be shown that there is a balanced decomposition tree which will not force nearby processors to be mapped too far from each other in the universal fat-pyramid.

Another open question is whether on-line simulation by a universal network can be performed with overhead better than $O(\log^2 A)$ bit times. Of the known on-line results, only the overhead in the word model for unit wire delay ($O(\log A)$) is known to be optimal (by an AT^2 lower bound of Bay and Bilardi [3, 4]).

Finally, it would be desirable to find networks that have good universality properties without using transmission lines.

Acknowledgements

Thanks to Charles Leiserson of MIT, Tom Cormen of Dartmouth University, Bruce Maggs of Carnegie-Mellon University, and Paul Bay of Thinking Machines Corporation for helpful discussions and for reviewing drafts of this paper.

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, MA, 1974.

[2] R. Aleliunas and A. L. Rosenberg. On embedding rectangular grids in square grids. *IEEE Trans. Computers*, C-31(9):907–913, Sept. 1982.

[3] P. Bay and G. Bilardi. Deterministic on-line routing on area-universal networks. In *31st Annual Symposium on Foundations of Computer Science*, pages 297–306. IEEE Computer Society Press, 1990.

[4] P. E. Bay. *Area-Universal Interconnection Networks for VLSI Parallel Computers*. PhD thesis, Department of Computer Science, Cornell University, May 1992.

[5] J. L. Bentley, D. Haken, and J. B. Saxe. A general method for solving divide-and-conquer recurrences. Technical Report CMU-CS-78-154, Department of Computer Science, Carnegie-Mellon University, Dec. 1978.

[6] S. N. Bhatt and F. T. Leighton. A framework for solving VLSI graph layout problems. *Journal of Computer and System Sciences*, 28(2):300–343, Apr. 1984.

[7] R. P. Brent and H. T. Kung. Fast algorithms for manipulating formal power series. *Journal of the ACM*, 25(4):581–595, Oct. 1978.

[8] A. M. Despain and D. A. Patterson. X-Tree: A tree structured multi-processor computer architecture. In *Proceedings of the 5th Annual International Symposium on Computer Architecture*, pages 144–151. ACM/IEEE, 1978.

[9] R. I. Greenberg. *Efficient Interconnection Schemes for VLSI and Parallel Computation*. PhD thesis, Department of Electrical Engineering & Computer Science, Massachusetts Institute of Technology, Aug. 1989. MIT/LCS/TR-456.

[10] R. I. Greenberg. The fat-pyramid: A robust network for parallel computation. In W. J. Dally, editor, *Advanced Research in VLSI: Proceedings of the Sixth MIT Conference*, pages 195–213. MIT Press, 1990.

[11] R. I. Greenberg and C. E. Leiserson. A compact layout for the three-dimensional tree of meshes. *Applied Mathematics Letters*, 1(2):171–176, 1988. Also see erratum in vol. 1, no. 3, p. 315.

[12] R. I. Greenberg and C. E. Leiserson. Randomized routing on fat-trees. In S. Micali, editor, *Randomness and Computation*. Volume 5 of *Advances in Computing Research*, pages 345–374. JAI Press, 1989.

[13] R. I. Greenberg and H.-C. Oh. Packet routing in networks with long wires. In *Proceedings of the 30th Annual Allerton Conference on Communication, Control, and Computing*, pages 664–673, 1992. Revised version in *Journal of Parallel and Distributed Computing*, 31(2):153–158, December 1995.

[14] F. T. Leighton, B. M. Maggs, A. G. Ranade, and S. B. Rao. Randomized routing and sorting on fixed-connection networks. *Journal of Algorithms*, 17(1):157–205, July 1994.

[15] F. T. Leighton, B. M. Maggs, and S. B. Rao. Packet routing and job-shop scheduling in O(congestion + dilation) steps. *Combinatorica*, 14(2):167–180, 1994.

[16] T. Leighton, B. Maggs, and S. Rao. Universal packet routing algorithms. In *29th Annual Symposium on Foundations of Computer Science*, pages 256–269. IEEE Computer Society Press, 1988.

[17] C. E. Leiserson. Area-efficient graph layouts (for VLSI). In *21st Annual Symposium on Foundations of Computer Science*, pages 270–281. IEEE Computer Society Press, 1980.

[18] C. E. Leiserson. Fat-trees: Universal networks for hardware-efficient supercomputing. *IEEE Trans. Computers*, C-34(10):892–901, Oct. 1985.

[19] C. E. Leiserson. VLSI theory and parallel supercomputing. In C. L. Seitz, editor, *Advanced Research in VLSI: Proceedings of the Decennial Caltech Conference on VLSI*, pages 5–16. MIT Press, 1989.

[20] C. E. Leiserson, Z. S. Abuhamdeh, D. C. Douglas, C. R. Feynman, M. N. Ganmukhi, J. V. Hill, W. D. Hillis, B. C. Kuszmaul, M. A. S. Pierre, D. S. Wells, M. C. Wong, S.-W. Yang, and R. Zak. The network architecture of the connection machine CM-5. In *Proceedings of the 4th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 272–285. Association for Computing Machinery, 1992.

[21] C. E. Leiserson and B. M. Maggs. Communication-efficient parallel algorithms for distributed random-access machines. *Algorithmica*, 3:53–77, 1988.

[22] S. L. Scott and J. R. Goodman. The impact of pipelined channels on k-ary n-cube networks. *IEEE Trans. Parallel and Distributed Systems*, 5(1):2–16, Jan. 1994.

[23] C. H. Séquin, A. M. Despain, and D. A. Patterson. Communication in X-TREE, a modular multiprocessor system. In *ACM 78: Proceedings 1978 Annual Conference*, pages 194–203, 1978.

[24] D. B. Shmoys, C. Stein, and J. Wein. Improved approximation algorithms for shop scheduling problems. In *Proceedings of the 2nd Annual SIAM Symposium on Discrete Algorithms*, pages 148–157, 1991.

[25] S. L. Tanimoto. Towards hierarchical cellular logic: Design considerations for pyramid machines. Technical Report 81-02-01, Department of Computer Science, University of Washington, Feb. 1981.

[26] C. D. Thompson. *A Complexity Theory for VLSI.* PhD thesis, Department of Computer Science, Carnegie-Mellon University, 1980.