The Fat-Pyramid: A Robust Network for Parallel Computation

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

Abstract

This paper shows that a *fat-pyramid* of area $\Theta(A)$ built from processors of size $\lg A$ requires only $O(\lg^2 A)$ slowdown in *bit-times* to simulate any network of area A under very general conditions. Specifically, there is no restriction on processor size (amount of attached memory) or number of processors in the competing network, nor is the assumption of unit wire delay required. This paper also derives upper bounds on the slowdown required by a fat-pyramid to simulate a network of larger area in the case of unit wire delay.

1 Introduction

This paper introduces the *fat-pyramid* network and shows that it is a good candidate as the basis for a general-purpose parallel computer. The flexibility of this network stems from its ability to efficiently simulate any other network of comparable physical size under general conditions.

Especially notable is the capability of the fat-pyramid to contend with the issue of long wires. Previous work on universal networks has generally assumed that unit time suffices to traverse a wire of any length. But this paper shows that such an assumption is unnecessary since competing networks may be simulated on a fat-pyramid in such a way that the distance traversed by any message is not much greater in the fat-pyramid than in the competing network.

Most previous work on universal networks has actually ignored not only wire length issues but also the broader issue of wiring difficulty and area consumption. That is, network cost has usually been measured in terms of processor count and perhaps the degree of the nodes. This paper seeks to better measure real-world costs by focusing on area consumption under standard VLSI modeling assumptions as in some other recent works [6, 8, 9, 10]. 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 [6] using the ideas in [7].

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 [13]. 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. (Other models are easily accommodated; for example, a "dance-hall" model which places processors and memory modules in different locations on a network can be viewed as a special case of the model considered here.)

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 [10]. The fat-pyramid may be viewed as a fat-tree in the style introduced in [8, Sec. 7] (the *butterfly fat-tree*) augmented by hierarchical mesh connections as illustrated in Figure 1. Ignoring the mesh connections, shown with thick lines, the network may be viewed as 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, denoted by $\mathrm{cap}(c)$. By using two types of constant-size switches, it is possible to build fat-pyramids with essentially arbitrary channel capacities as illustrated in Figure 2 borrowed from Leiserson [12]. (The capacities of the mesh connections in the fat-pyramid match those of the adjacent channels.)[1]

The choice of channel capacities is of key importance in specifying the design of fat-pyramids or its fat-tree precursors. As the channel capacities are increased, the communication bandwidth increases, and it is easier for the network to perform complex computations, but the area of the network also increases. Judicious choice of the channel capacities yields a network of modest area which is still able to efficiently route all the messages generated in a competing network of comparable area.

Major issues and organization of this paper

Intuitively, the fat-pyramid combines the strengths of the fat-tree and the mesh. A fat-tree with appropriate channel capacities can efficiently simulate any network of comparable area under the unit wire delay assumption. (Some details of this result are given in Section 2.) This is perfectly acceptable if the wires are not too long, and the time to send messages along a path in the network is dominated by the number of switches on the path. But maximum wire length in the fat-pyramid grows rapidly with network size; straightforward layouts of the fat-pyramid have maximum wire length

[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 Uhr) [15, 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 [5, 14].

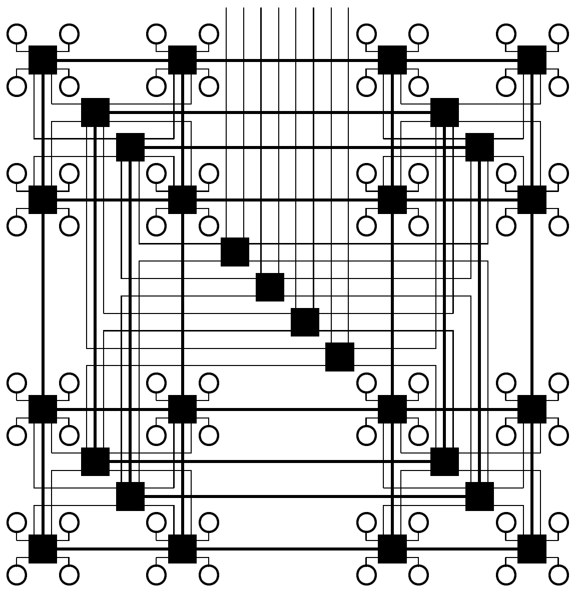

Figure 1: A fat-pyramid. This network is obtained by superposing hierarchical mesh connections on a butterfly fat-tree. The original fat-tree connections are represented by thin lines and the mesh connections by thick lines. (A different layout of the fat-pyramid is used to obtain results independent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

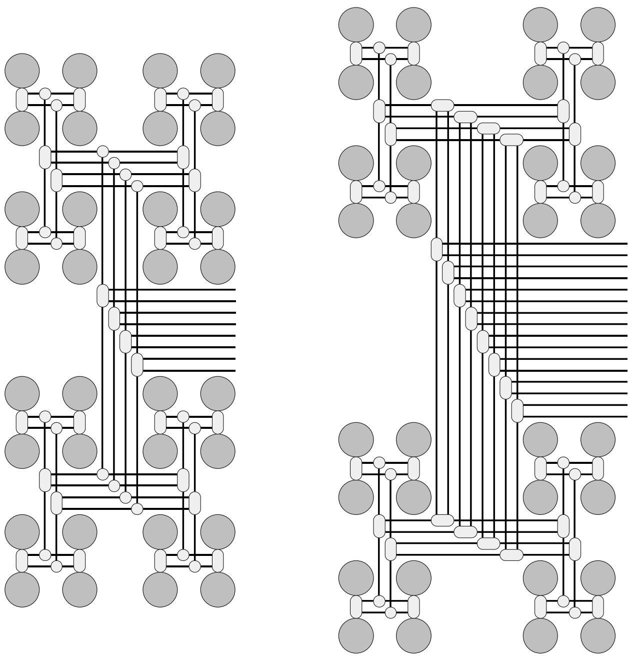

Figure 2: A scalable fat-tree. Essentially arbitrary channel capacities can be obtained by using two types of constant-size switches.

$\Theta(\sqrt{A})$ for a network of area A. 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 bit is a linear function of wire length, the fat-tree will require $\Omega(\sqrt{A})$ time to route messages between such a 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) 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 $O(\lg A)$ 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.

In addition to coping with nonunit wire delay, this paper provides some other extensions to the universality results of [8]. These issues are introduced in the context of universality results for fat-trees, but some of the analyses readily carry over to the fat-pyramid and the case of nonunit wire delay. In particular, this paper explicitly considers the size (amount of attached memory) of processors comprising the networks being compared. One reason processor size is important is that processors in a fat-tree of area A must be of size $\Omega(\lg A)$ in order to address all the other processors.[2] This paper indicates the outcome of comparison between networks of larger and/or different processors and considers the question of best processor size for a universal network. This paper also uses fat-tree channel capacities that yield a denser packing of processors than in [8], so that artificial restrictions on the number of processors in a competing network are removed.[3]

The foregoing results, found in Sections 2 and 3 are applicable in the context of nonunit wire delay as discussed in Section 4. Section 5, on the other hand, considers only the case of unit wire delay; it analyzes the ability of a universal network to simulate other networks that use more total area but the same processor area.

Section 6 contains concluding remarks.

[2]The requirement is $\Omega(\lg n)$, where n is the number of processors, implying $\Omega(\lg A)$ unless the number of processors in the fat-tree is too small to achieve any reasonable simulation.

[3]Leighton, Maggs, and Rao [9] also eliminate this restriction by a different means.

Additional background and terminology

In this paper, universality results are obtained by invoking known results for routing messages on fat-trees and extending these routing results for use on the fat-pyramid. The routing results are expressed in terms of simple characteristics of the set of messages to be routed. Within the fat-tree, the difficulty of routing a set M of messages between pairs of processors is well summarized by a measure we refer to as the *load factor*, $\lambda(M)$. First, let us define the *load* $\mathrm{load}(M, c)$ of M on a channel c of a fat-tree to be the number of messages in M which must pass through c. Then the *load factor* of M on c is

$$\lambda(M, c) = \frac{\mathrm{load}(M, c)}{\mathrm{cap}(c)} ,$$

and the *load factor* of M on the entire fat-tree is

$$\lambda(M) = \max_c \lambda(M, c) .$$

The load factor λ is a lower bound on the time required to deliver a set of messages on a fat-tree, and the routing algorithms which have been demonstrated for fat-trees obtain (high probability) upper bounds on the delivery time δ in the form

$$\delta(\lambda, n) = f(n)\lambda + g(n) ,$$

where f and g are small polylogarithmic functions[4] of the number of processors, n. (Note that this formulation ensures that $\delta(O(\lambda), n) = O(\delta(\lambda, n))$; this fact will be used several times in the following sections.)

Of particular interest is the case where n does not exceed 2^λ, implying that the delivery time is just a small polynomial function of λ. We will use δ with one argument to represent this case, i.e.,

$$\delta(\lambda) = \delta(\lambda, 2^\lambda) .$$

The known routing schemes for fat-trees obtain $\delta(\lg n)$ in the range of $\lg^2 n$ to $\lg^4 n$ in bit-times. The packet routing scheme of Leighton, Maggs, and Rao [9] achieves $\delta(\lg n) = \lg^2 n$ bit-times on a fat-tree.[5] This scheme involves selecting random keys which are used to prioritize packets as they pass through the switches; only constant-size queues are required. Included in [8] are circuit-switched algorithms which randomize in the choice of messages to send in each of a series of *delivery cycles* and work with switches that fill their output lines from an arbitrary subset of the input messages. Once an appropriate probability is used for the sending of each message from its source processor, the internal switches need not perform any comparison

[4] That is, these functions are upper bounded by a polynomial in the logarithm of the argument. Thus, functions such as $\lg^2 n$ or $\lg^3 n \lg\lg n$ qualify.

[5] This result has been translated from $\lg n$ time in the word model.

of priorities or other message characteristics. (Some of the algorithms in [8] apply not to the butterfly fat-tree but to fat-trees with large concentrator switches as introduced by Leiserson [10]. Many of the results in this paper can be applied to fat-trees of either variety, but the results independent of wire delay require a fat-pyramid based upon the butterfly fat-tree.)

The focus in this paper is on the choice of universal network and the mapping of other networks to it in order to achieve a modest communications burden. We obtain results about how good a network can be at facilitating this mapping and generally leave open the possibility of applying various routing approaches, but the results are occasionally specified concretely using the $\delta(\lg n) = \lg^2 n$ approach of Leighton, Maggs, and Rao [9]. Thus, it is possible to see the effect which would result from an improved routing algorithm as well as the current best known results.

In this paper, we make a few important but reasonable assumptions relating to time and space. First of all, the technology being used determines a minimum feature size, which is our unit of space. (Sometimes processor size will be taken as the measure of unit space for simplicity, but ultimately we will seek to explicitly account for processor size in terms of the more fundamental unit of space.) Second there is a fundamental lower bound on the time to switch a wire of unit length, which will be used as our basic unit of time. In VLSI technologies, such a bound is determined by the capacitance and resistance of a minimum-size transistor and the necessity of increasing capacitance if resistance is to be reduced. In fact, we will initially assume that this is the only source of delay and that transmission along any wire can be accomplished in unit time, but later we will relax this assumption. In any case, we assume that the number of bits which can leave an area of a chip in unit time is proportional to the perimeter of the area.

It is also convenient to assume that operation of any competing network is divided into separate 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 should be valid even if the competing network interleaves computation and communication in a more complicated fashion. The validity of the simplification is established rigorously in [6, Sec. 4.5] in the case of unit wire delay; the nonunit delay case requires a more involved analysis and routing results that allow the pipelining of message sets injected into the universal network at different times.

For convenience, we will use the following terminology throughout this paper:

> **Definition:** We say that network B can Δ-simulate network A if, for any t, the operations performed by network A in time t can be performed by network B in time $t\Delta$.

2 A linear-size fat-tree

This section considers comparisons between networks built out of the same processors. It begins by constructing a fat-tree on unit-size processors which occupies area linear in the number of processors. With processors packed so densely, a very simple one-to-one mapping of a competing network's processors to those of the fat-tree ensures that message sets delivered in unit time by a competing network of area A have $O(\lg A)$ load factor in the fat-tree. Fat-tree routing mechanisms require processors of size at least logarithmic in the number of processors (in order to address all other processors), but by simply scaling the measure of area everywhere, the load factor result for unit-size processors leads to a valid simulation result for networks built out of the same processors of size $\Omega(\lg A)$. Nonetheless, we will explicitly introduce processor area since it yields an improved bound for very large processors and prepares the way for comparisons between networks built from different processors.

Specifying the universal fat-tree requires making an appropriate choice of the channel capacity function $c(p)$, which denotes the capacity of a fat-tree channel on top of a subtree of p processors. This section will show that the best construction may not be modular, i.e., the channel capacity on top of a subtree of the network may depend on the total size to which the network is to be built rather than depending just on the size of the subtree. It will be shown that modularity can be obtained using processors of size $\lg^2 A$ to build a fat-tree of area A, but we will see in later sections that this may increase the cost of simulating networks with processors of a different size if fat-tree or fat-pyramid routing strategies are developed which come closer to the lower bound on routing time.

Observe first that if we let $c(p) = \lceil \sqrt{p}/\lg A \rceil$, we can build a fat tree of A unit-size processors in area $\Theta(A)$. The area can be derived by solving a recurrence relation for the side length $S(n)$ of a fat-tree on n processors in the H-tree layout style of Figure 1. We have

$$S(n) \quad = \quad 2S(n/4) + O(c(n)) \ ,$$

with solution

$$S(n) = \sqrt{n}S(1) + \sum_{i=0}^{\log_4 n - 1} 2^i O(c(n/4^i)) \ . \qquad (1)$$

Substituting for $S(1) = 1$ and $c(n/4^i) \leq 1 + \sqrt{n/4^i}/\lg A$, we obtain

$$S(n) \quad \leq \quad O(2\sqrt{n} + \sqrt{n}(\log_4 n)/\lg A) \ ,$$

and $S(A) = \Theta(\sqrt{A})$. (We have assumed constant-size switches, but larger switches can be accomodated as required in [9] as long as we use larger processors and channel capacities as discussed below.)

Now we are prepared to present a result which strengthens the main universality result in [8] (translated to two dimensions) in that the restriction on the number of processors in the competing network is removed, but the proof here is actually more direct. Rather than using the ideas for balancing decomposition trees developed by Bhatt and Leighton [3] and Leiserson [10], we can consider a straightforward geometric mapping of competing networks to fat-trees. (This idea was actually introduced in [8] but only in the case of a fat-tree simulating networks of slightly smaller area.)

Lemma 2.1 *Consider networks with unit-sized processors, and let $\mathcal{R}$ be the set of all networks of area A. Then, there exists a fat-tree F of area $\Theta(A)$ such that any message set delivered in unit time by a network in $\mathcal{R}$ induces a load factor of $O(\lg A)$ on F.*

Proof. We use the channel capacities $c(p) = \lceil \sqrt{p}/\lg A \rceil$ to build a fat-tree of $n = A$ processors, which we have shown requires area $\Theta(A)$. Then we can just recursively bisect any $R \in \mathcal{R}$ in the straightforward geometric fashion, cutting nonsquare pieces in the shorter direction, until we have A pieces. There can be at most one processor in each of the pieces of the recursive bisection, and these processors can be mapped to F so that the recursive bisection of R matches the obvious recursive bisection of F. Then the perimeter of a piece of R corresponding to a subtree of $n/2^l$ processors in F is $O(\sqrt{A}/2^{l/2})$. Thus, for a channel at level l of the decomposition, the load factor of messages generated in unit time is

$$\begin{aligned} \lambda &= O((\sqrt{A}/2^{l/2})/c(n/2^l)) \qquad (2) \\ &= O((\sqrt{A}/2^{l/2})/(\sqrt{A/2^l}/\lg A)) \\ &= O(\lg A)\ . \end{aligned}$$

□

It is actually quite easy to generalize the above results to processors of size α. For simplicity, we assume that processors are square, i.e., $\sqrt{\alpha}$ by $\sqrt{\alpha}$. By building a fat-tree on $n = A/\alpha$ processors with channel capacity function $c(p) = \lceil \sqrt{p\alpha}/\lg(A/\alpha) \rceil$, we obtain area A and load factor $O(\lg(A/\alpha))$. This can be seen by simply making the correct substitutions into Equations 1 and 2.

We can also make the channel capacities slightly larger at the lower levels of the tree than was just indicated. (Such a change is required by some of the routing algorithms discussed in [8].) Specifically, if we increase channel capacities to $\lceil \sqrt{\alpha} + \sqrt{p\alpha}/\lg(A/\alpha) \rceil$, the asymptotic area of the fat-tree does not increase, as can be seen by substituting into Equation 1. The load factor bound remains valid, of course, when the channel capacities are increased.

Using our bound on the load factor, we can now state a simulation result in terms of the running time $\delta(\cdot)$ of the message delivery algorithm.

Theorem 2.2 *Consider networks with processors of size $\alpha = \Omega(\lg A)$, and let $\mathcal{R}$ be the set of all networks of area A. Then, there exists a fat-tree F of area $\Theta(A)$ which can $O(\delta(\lg(A/\alpha)))$-simulate any network in $\mathcal{R}$.*

Proof. We have seen that processors can be mapped one-to-one, and the load factor for each message set of a competing network in $\mathcal{R}$ is $O(\lg(A/\alpha))$. Thus, the delivery time for each message set is $O(\delta(\lg(A/\alpha)))$. □

Using the routing scheme of Leighton, Maggs, and Rao [9], which has $\delta(\lg n) = \lg^2 n$, yields the following corollary:

Corollary 2.3 *Consider networks with processors of size $\alpha = \Omega(\lg A)$, and let $\mathcal{R}$ be the set of all networks of area A. Then, there exists a fat-tree F of area $\Theta(A)$ which can $O(\lg^2(A/\alpha))$-simulate any network in $\mathcal{R}$.* □

This result is similar to a result given in [9] using a fat-tree with the lower levels replaced by meshes. By using the construction here, it is possible to instead retain a uniform routing strategy throughout all levels of the tree. The idea of adding mesh connections to a fat-tree is, however, a useful one when wire delay concerns are addressed, as we shall see in Section 4.

One final note is in order. When $\alpha = \Theta(\lg^2 A)$, we can achieve a modular design; we can use $c(p) = \lceil \sqrt{p} \rceil$, or rounding slightly differently, capacities which double at every other level as in Figure 1. These channel capacities are not adequate to justify the use of one of the routing algorithms presented in [8], the one which required channels of capacity $\Omega(\lg n)$, but other routing algorithms remain valid.

3 Different sized processors

In this section, we consider the possibility of comparing a universal network with processors of one size to other networks with processors of different size. In doing so, we must establish correspondences of single processors in one network with multiple processors in another network. We consider both many-to-one and one-to-many mappings of a competing networks' processors to those of a universal network. The combined observations show that a universal network designed to simulate networks of arbitrary processor size is best built with processors of size α in the range $\lg A \le \alpha \le \delta(\lg A)$.

In order to compare parallel machines within the framework of this paper, we must place some limitations on how we compare machines built from different processors. Rather than get involved in such issues as RISC vs. CISC or other detailed questions of best design for an individual processor, we assume that the 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. The one difference in processors which remains under consideration is the amount of memory attached. For simplicity, we assume that the size of the memory does not affect the instruction execution time; incorporating a small cost for increasing memory size would cause little change to the results given here.

We consider in turn the two cases of the universal network having processors which are larger or smaller than the processors of the simulated network. Then we consider more general statements independent of the sizes of processors in the competing network. In any case, let α_X represent the area of a processor in routing network X.

First, let us consider a competing network with processors smaller than those from which the universal network is to be built. In this case, we can map the competing network R to the universal fat-tree F as before, except that the decomposition of R stops before we get down to individual processors. Instead, we map α_F/α_R processors of R to each processor of F. The computations performed by this block of processors in time t, excluding any communication with processors outside the block, can be realized in time $O(t\alpha_F/\alpha_R)$ on the processor of F. Meanwhile, the communication between blocks can be accomplished with overhead $O(\delta(\lg(A/\alpha_F)))$, yielding the following generalization of Theorem 2.2:

Theorem 3.1 *For any α_R, let $\mathcal{R}$ be the set of all networks of area A built out of processors of area α_R. Then, for any $\alpha_F \geq \max\{\alpha_R, \lg A\}$, there exists a fat-tree of area $\Theta(A)$ built out of processors of area α_F which can $O(\max\{\delta(\lg(A/\alpha_F)), \alpha_F/\alpha_R\})$-simulate any network in $\mathcal{R}$.* □

Now, let us consider a competing network with processors larger than those from which the universal network is to be built. In this case, we decompose the competing network $R \in \mathcal{R}$ down to individual processors as before, but we assign α_R/α_F processors of the fat-tree F to simulate each processor of R. We divide the memory of a processor of R among the corresponding α_R/α_F processors of F. Then as long as processors of F are large enough to address the α_R/α_F regions of memory, we can treat the flow of data to the different pieces of memory just as we would the communication among a set of processors being simulated. Thus, we can view the operation of R as proceeding on the larger number A/α_F of subdivided processors, yielding an additional generalization to Theorem 2.2:

Theorem 3.2 *For any $\alpha_F \geq \lg A$ and any $\alpha_R \geq \alpha_F$, there exists a fat-tree of area $\Theta(A)$ built out of processors of area α_F which can $O(\delta(\lg(A/\alpha_F)))$-simulate any network of area A and processors of area α_R.* □

It is straightforward to generalize Theorems 3.1 and 3.2 to situations in which a single simulated network has processors of many different sizes. In that case, we can simply let α_R represent the minimum size of processors in R, and both theorems will still hold, though the proofs change slightly. In fact, we can combine Theorems 3.1 and 3.2, since the α_F/α_R term in Theorem 3.1 becomes irrelevant for $\alpha_R \geq \alpha_F$:

Theorem 3.3 *For any α_R, let $\mathcal{R}$ be the set of all networks of area A with each processor having area at least α_R. Then, for any $\alpha_F \geq \lg A$, there exists a fat-tree of area $\Theta(A)$ built out of processors of area α_F which can $O(\max\{\delta(\lg(A/\alpha_F)), \alpha_F/\alpha_R\})$-simulate any network in $\mathcal{R}$.* □

Since the overhead in Theorem 3.3, will never exceed $O(\delta(\lg A))$ for $\alpha_R \geq \alpha_F$, we see that the overhead for a fat-tree simulating a network with larger processors is largely insensitive to the relative size of processors. Thus, if we wish to build a universal fat-tree to simulate networks of unknown processor size, it seems that we do best by making the fat-tree processors small enough that $\delta(\lg A)$ will always dominate α_F/α_R in comparisons against networks with smaller processors. Recalling that fat-tree routing mechanisms require that processors of F should be large enough to address $\Omega(A)$ processors, we are led to choose the processor size to satisfy $\lg A \leq \alpha_F \leq \delta(\lg A)$. Then we have the following corollaries to Theorem 3.3.

Corollary 3.4 *There is a fat-tree of area $\Theta(A)$ that can $O(\delta(\lg A))$-simulate any network of area A and processors of arbitrary area.* □

Corollary 3.5 *There is a fat-tree of area $\Theta(A)$ that can $O(\lg^2 A)$-simulate any network of area A and processors of arbitrary area.* □

4 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 [3] 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, the bounds on simulation time shown in earlier sections can almost always be obtained by using an appropriate layout of the fat-pyramid network. A routing path of length d in a competing network of area A corresponds to a path of length $O(d + \lg A)$ in the fat-pyramid, which generally implies that asymptotic simulation overhead is no worse than in the unit wire delay case. Some of the ideas in this section were suggested by Charles Leiserson and Tom Cormen of MIT.

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., [2, 4, 11],[1, p. 280]) in order to obtain results about large classes of functions.) Specifically, let $w(d)$ denote the time required to transmit a bit along a wire of length d; 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
> $$\frac{w(d+x)}{w(d)} \leq \frac{\lg d + x}{\lg d}$$
> for all $x \geq 0$ and $d \geq c$.

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 of the form $cn^q \lg^k n$ 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
> $$\frac{w(d+x)}{w(d)} \leq \frac{d+x}{d}$$
> for all $x \geq 0$ and $d \geq c$.

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

(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 conditions are satisfied by any function w satisfying $c_1 n^q \lg^k n \leq w(n) \leq c_2 n^q \lg^k n$ for sufficiently large n, 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-tree, that is, one in which the components 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 [3, pp. 325–326] and Thompson [16, pp. 36–38]. It may be easiest to think about a fold and squash transformation of a butterfly fat-tree and then a superposition of the hierarchical mesh connections, requiring only a constant factor expansion in area. The result of the fold and squash transformation is illustrated in Figure 3 with inter-level connections omitted but switches

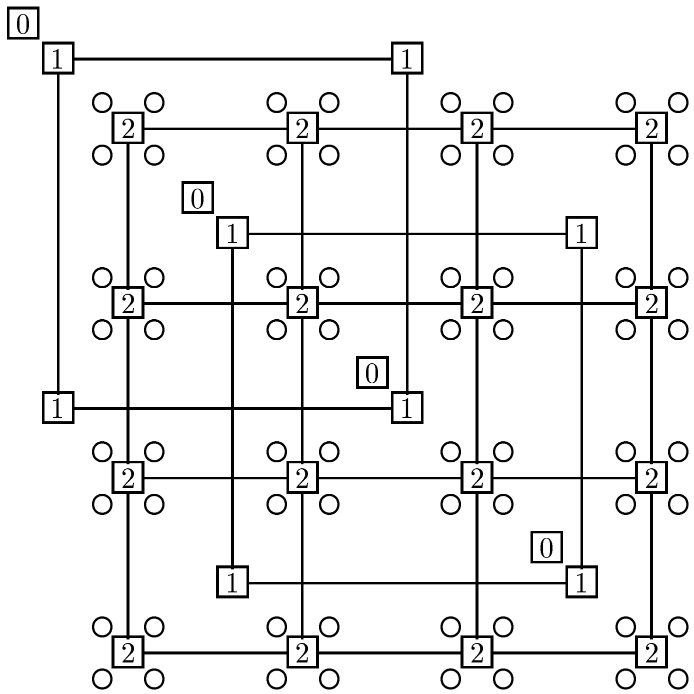

Figure 3: A regular layout of the fat-pyramid, but with the original fat-tree connections removed for ease of illustration. Each switch is numbered with its level from the top of the tree so that this figure can be compared with Figure 1. The network shown here allows good results to be obtained without the unit wire delay assumption.

labeled according to the number of levels from the top of the tree. (Visualization of the transformation of the inter-level connections may be aided by viewing the regular layout of the two-dimensional tree of meshes illustrated in [7].) In general, we must stop folding when we reach a level at which the channel capacities stop decreasing sufficiently (by a factor of 2 at every level in the underlying 4-ary tree), so we may be left with H-trees near the leaves. In any case, we use a fat-tree with processors of area at most $\lg^2 A$ in accordance with the results of Section 3, and the area of the H-tree blocks is $\lg^2 A$.

Messages in the 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 mesh edges. More precisely the routing path is formed by going up tree edges 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 tree edges. As long as the mesh edges are of sufficiently large constant capacity, all of the existing routing algorithms work as well as before; the shortcuts do not cause any extra messages to go through any of the tree edges, and the messages which go through any mesh edge are only those which would have gone up the tree from a constant number of nearby switches.

A key property of the layout in Figure 3 is that the wires connected to a switch l levels up from the H-tree blocks are of length $O(2^l \lg A)$, where A is the area of the fat-tree. To see this, observe that the H-tree blocks near the leaves have side length $\lg A$, and the upper levels can be embedded in a tree of meshes graph of $O(\lg A)$ levels in such a way that each edge connected to a switch l levels up is mapped to at most $O(2^l)$ edges in the tree of meshes. The fold and squash layout of the tree of meshes given in [3] has maximum edge length $O(\lg A)$, and there is at most $\lg A$ extra length caused by the H-tree blocks in our layout. Thus, the length of an edge connected to a switch l levels up in the folded and squashed fat-pyramid is $O(2^l \lg A)$.

Now we can show that the mapping of competing networks to a universal fat-pyramid (having channel capacities as in a universal fat-tree) does not stretch any wires by very much.

Theorem 4.1 *Let R be a network occupying a square of area A. Then, R can be mapped to a fat-pyramid F of area $\Theta(A)$ so that any message following a path of length d in R travels only $O(d + \lg A)$ distance in F.*

Proof. Observe first that if we use the straightforward mapping of R to F, processors separated by distance d in R are at most $\lceil d/\lg A \rceil$ H-tree blocks apart when mapped to F. Since four adjacent subtrees on $(\lceil d/\lg A \rceil)^2$ H-tree blocks must suffice to cover such a pair of processors, the routing path connecting these processors needs only to go up $\lg(\lceil d/\lg A \rceil)$ levels and use two mesh edges. Since any wire connected to a switch l levels up is of length $O(2^l \lg A)$, the length of the routing path connecting processors at distance d in R is $O(d + \lg A)$. $\square$

The above result can be used to almost always insure that asymptotic simulation time does not degrade in the case of nonunit wire delay. Clearly, simulation time does not degrade if the competing network operates in separate phases of communication and computation and frequently produces messages which must travel over distance at least $\Omega(\lg A)$. But we can obtain more general results by using the regularity conditions assumed for the wire delay function w and introducing some modifications to known routing algorithms and analyses.

Extension of the circuit-switched routing approach of [8] to the nonunit wire delay case is discussed in [6]; here we concentrate on the packet routing approach of Leighton, Maggs, and Rao [9]. Also, a nonconstructive result is given here, some additional technical details being necessary to provide an on-line routing algorithm. Leighton, Maggs, and Rao show that for unit wire delay in the word model, there exists a schedule for routing any set of packets in time proportional to the sum of the maximum congestion (largest number of messages that must traverse a single wire) and the maximum distance (largest number of switches in a message path). They also show that the maximum congestion on a fat-tree is $O(\lambda + \lg A)$, where λ is the load factor of the set of messages. This result carries over to the fat-pyramid since the use of short cuts of appropriate capacity does not increase congestion in any part of the network.

To apply the results of Leighton, Maggs, and Rao in the case of nonunit wire delay, 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 messages 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 d. Let T be the time required to deliver the set of messages in the competing network, and note that $T \geq w(d)$. Also, the load factor of this message set is $O(T \lg A)$. Furthermore, the maximum number of fat-pyramid edges which a message must traverse is $2 \lg d$, each containing at most $w(d+\lg A)$ real and imaginary switches. Thus the simulation overhead Δ can be bounded as follows:

$$
\begin{aligned}
\Delta &\leq \frac{w(d+\lg A)O(\lg d) + O(T \lg A)}{T} \\
&\leq \frac{w(d+\lg A)O(\lg d)}{w(d)} + \frac{O(T \lg A)}{T} \\
&\leq \frac{(\lg d + \lg A)O(\lg d)}{\lg d} + O(\lg A) \\
&\leq O(\lg A)\ ,
\end{aligned}
$$

where the third line follows from regularity condition C1.

5 Simulating larger networks

This section obtains upper bounds on the time required by a universal 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, 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 network. Thus, the results given so far are worst-case results for simulating networks of essentially the same total area. In this sense, these networks 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 of larger area but with no greater processor area.

In what follows, we let A_X represent the area of network X. Of necessity, however, we consider only competing networks in which the processor area does not exceed that of our universal fat-tree F. 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. For simplicity, we assume unit-size processors; the other variations discussed in the previous subsection are easily incorporated.

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 competing processors (or pieces of processor area as in Section 3). Fortunately, we can invoke the general theory developed by Bhatt and Leighton [3] and, in a fashion that is cleaner for our purposes, by Leiserson [10]. (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 4.) 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 5.1 *Let $\mathcal{R}$ be the set of networks of total area A_R and processor area A_F. A universal fat-tree of area $O(A_F)$ can $O(\delta(\sqrt{A_R/A_F}\lg A_F, A_F))$-simulate any network in $\mathcal{R}$.*

Proof. Using a balanced decomposition for $R \in \mathcal{R}$ as described above, we find that the load factor of a set of messages delivered by R in unit time is

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

at level l from the root of the fat-tree. The result follows. □

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 5.1. 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, at level l in the balanced decomposition of R, only $O(A_F/2^l)$ messages can leave a corresponding piece of R in unit time. Dividing by fat-tree channel capacity to determine the load factor, we obtain the following result:

Theorem 5.2 *A universal fat-tree of area $O(A_F)$ can $O(\delta(\sqrt{A_F}\lg A_F, A_F))$-simulate any network having processor area A_F.* □

6 Conclusion

This paper has shown that a fat-pyramid network can efficiently simulate any other network built in the same amount of area, without significant additional restrictions. 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.

The results given in this paper suggest that when building a universal network of area A, the best processor size to use is $\Theta(\lg A)$, or $\Theta(\lg^2 A)$ if modularity is more important than the ability to simulate networks of very small processors. The apparent lesson for the design of general-purpose parallel machines is that as total memory demands increase, most of the effort should go into expanding the number of processors rather than individual processor size.

A key open question is whether or not the routing algorithms for universal networks can be improved. It is possible that a better fat-tree routing algorithm could yield only $O(\lg A)$ slowdown in bit-times. Furthermore, it might be possible to take greater advantage of the hierarchical mesh connections in the fat-pyramid. If the notion of load factor is generalized to arbitrary networks by considering arbitrary cuts of the network as opposed to fat-tree channels, then network mappings onto the fat-pyramid yield constant load factor. It is possible that a routing scheme more clever than one relying almost exclusively on the tree connections could yield better message delivery times.

Acknowledgements

Thanks to Charles Leiserson of MIT for suggesting many of the areas of research covered in this paper and for providing extensive comments on earlier drafts. Thanks also to Tom Cormen and Bruce Maggs of MIT for helpful discussions.

This research was initiated at the Massachusetts Institute of Technology and was supported in part by the Defense Advanced Research Projects Agency under Contract N00014–87–K–0825, by the Office of Naval Research under Contract N00014–86–K–0593, and by a Fannie and John Hertz Foundation Fellowship.

References

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

[2] Jon Louis Bentley, Dorothea Haken, and James B. Saxe. A general method for solving divide-and-conquer recurrences. Technical Report CMU-CS-78-154, Department of Computer Science, Carnegie-Mellon University, December 1978.

[3] Sandeep N. Bhatt and Frank Thomson Leighton. A framework for solving VLSI graph layout problems. *Journal of Computer and System Sciences*, 28(2):300–343, April 1984.

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

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

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

[7] Ronald I. Greenberg and Charles E. Leiserson. A compact layout for the three-dimensional tree of meshes. *Applied Mathematics Letters*, 1(2):171–176, 1988.

[8] Ronald I. Greenberg and Charles E. Leiserson. Randomized routing on fat-trees. In Silvio Micali, editor, *Randomness and Computation.* Volume 5 of*Advances in Computing Research.* JAI Press, 1989. To appear. Earlier versions available in MIT/LCS/TM-307 and *26th Annual Symposium on Foundations of Computer Science*, 1985, pages 241–249.

[9] Tom Leighton, Bruce Maggs, and Satish Rao. Universal packet routing algorithms. In *29th Annual Symposium on Foundations of Computer Science*, pages 256–269. IEEE Computer Society Press, 1988.

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

[11] Charles 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.

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

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

[14] 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.

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

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