Design of FPGA Interconnect for Multilevel Metalization

Raphael Rubin
Dept. of CS, 256-80
California Institute of Technology
Pasadena, CA 91125
rafi@ugcs.caltech.edu

André DeHon
Dept. of CS, 256-80
California Institute of Technology
Pasadena, CA 91125
andre@acm.org

ABSTRACT

How does multilevel metalization impact the design of FPGA interconnect? The availability of a growing number of metal layers presents the opportunity to use wiring in the third-dimension to reduce switch requirements. Unfortunately, traditional FPGA wiring schemes are not designed to exploit these additional metal layers. We introduce an alternate topology, based on Leighton's Mesh-of-Trees, which carefully exploits hierarchy to allow additional metal layers to support arbitrary device scaling. When wiring layers grow sufficiently fast with aggregate network size (N), our network requires only O(N) area; this is in stark contrast to traditional, Manhattan FPGA routing schemes where switching requirements alone grow superlinearly in N. In practice, we show that, even for the admittedly small designs in the Toronto "FPGA Place and Route Challenge," the Mesh-of-Trees networks require 10% less switches than the standard, Manhattan FPGA routing scheme.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Network Topology*; B.7.1 [**Integrated Circuits**]: Types and Design Styles—*VLSI*

General Terms

Design, Experimentation, Theory

Keywords

Mesh-of-Trees, Hierarchical, Multi-level Metalization, FPGA, Interconnect

1. INTRODUCTION

How should FPGA interconnect be designed to exploit multilevel metalization?

VLSI technology has advanced considerably since the first FPGAs [7]. Feature sizes have shrunk, die sizes and raw capacities have grown, and the number of metal layers available for interconnect has grown. The most advanced VLSI processes now sport 7–9 metal layers, and metal layers have grown roughly logarithmically in device capacity [5].

How should this shift in available resources affect the way we design FPGAs?

One can view multi-level metalization, and particularly the current rate of scaling, as an answer to the quandary that interconnect requirements for typical designs (Rent's Rule [13] $p > 0.5$) grows faster than linearly with gate count [10] [9]. If we can accommodate the growing wire requirements in the third dimension using multiple wire layers, then we may be able to maintain constant density for our devices. Alternately, if we cannot do this, the (2D) density of our devices necessarily decreases as we go to larger device capacities.

The existence of additional metal layers is not sufficient, by itself, to stave off this problem. We must further guarantee that we can contain the active silicon area to a bounded area per device (*e.g.* an asymptotically bounded number of switches per gate) and that we can topologically arrange to use the additional metalization.

We show that the dominant, traditional, Manhattan style, interconnect scheme is not directly suited to exploiting multilevel metalization (Section 2). Its superlinear switch requirements preclude it from taking full advantage of additional metal layers. The density of these architectures ultimately decreases with increasing gate count.

We introduce an alternative topology, based on Leighton's Mesh-of-Trees [15] [14] which exploits hierarchy more strictly while retaining the two-dimensional interconnect style of the Manhattan interconnect (Section 3). We show that this topology has an asymptotically constant number of switches per endpoint and that it can be arranged to fully exploit additional metal layers. As a result, given a sufficient interconnect layer growth rate, the gate density remains constant across increasing gate counts.

In Section 4, we summarize a set of empirical comparisons which place our Mesh-of-Trees design relative to standard Manhattan routing topologies and explore a few of the important design parameters available to this topology.

2. MANHATTAN INTERCONNECT

2.1 Base Model

Figure 1 shows the standard model of a Manhattan (Symmetric [6], Island-style [4]) interconnect scheme. Each compute block (LUT or island of LUTs) is connected to the adjacent channels by a C-box. At each channel intersection is an S-box. In the C-box, each compute block IO pin is con-

FPGA'03, February 23–25, 2003, Monterey, California, USA.

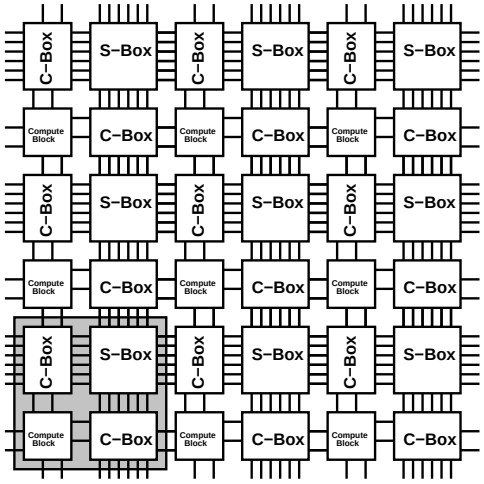


Figure 1: Manhattan Interconnect Model ($W = 6$, $I = 2$ shown)

nected to a fraction of the wires in a channel. At the S-box, each channel on each of the 4 sides of the S-box connects to one or more channels on the other sides of the S-box.

Early experiments [6] considered the number of sides of the compute block on which each input or output of a gate appeared (T), the fraction of wires in each channel each of these signals connected to (F_c), and the number of switches connected to each wire entering an S-box (F_s). Regardless of the detail choices for these numbers, they have generally been considered constants, and the asymptotic characteristics are independent of the particular constants chosen.

To keep this general, let's simply assume each side of the compute block has I inputs or outputs to the channel. If we are thinking about a single-output k-LUT as our compute block, then $I = \frac{T \times (k+1)}{4}$. The number of switches in a C-box is:

$$C_{sw} = 2 \cdot F_c \cdot I \cdot W \tag{1}$$

W is the width of the channel. Each S-box requires:

$$S_{sw} = \left(\frac{4}{2}\right) \cdot F_s \cdot W = 2 \cdot F_s \cdot W \tag{2}$$

On average, each compute block adds two connection boxes and one S-box (as shown highlighted in Figure 1). So, the total number of switches per compute block is:

$$B_{sw} = 2 \cdot C_{sw} + S_{sw} = 2W\left(2 \cdot F_c \cdot I + F_s\right) \tag{3}$$

Dropping the constants we get:

$$B_{sw} = O(W) \tag{4}$$

That is, we see that the number of switches required per compute block is linear in W, the channel width.

We can get a loose bound on channel width simply by looking at the bisection width of the design. If a design has a minimum bisection width BW, then we have a lower bound on the channel width:

$$BW \leq \sqrt{N} \cdot W \tag{5}$$

that is, we must provide at least BW bandwidth across the $\sqrt{N}$ row (or column) channels which cross the middle of the chip. This allows us to solve for a lower bound on W:

$$W \geq \frac{BW}{\sqrt{N}} \tag{6}$$

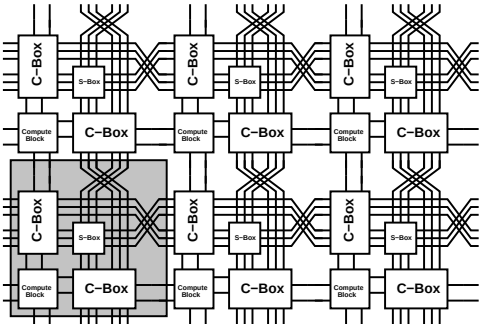


Figure 2: Segmentation in Manhattan Interconnect Model (Example shows $L_{seg} = 2$)

Empirically, we find that the bisection width of a design can often be characterized by the Rent's Rule relation [13]:

$$BW = IO = cN^p \tag{7}$$

This now allows us to define a correspondence between W and N:

$$W \geq \frac{cN^p}{\sqrt{N}} = O\left(N^{(p-0.5)}\right) \tag{8}$$

This is the same correspondence which one gets by combining the results of Donath [11] and El Gamal [12] for $p > 0.5$. This means:

$$B_{sw} = O(W(N)) = O\left(N^{(p-0.5)}\right) \tag{9}$$

All together, this says that as we build larger designs, if the interconnect richness is greater than $p = 0.5$, the switch requirements per compute block is growing for the Manhattan topology; this means the aggregate switching requirements grow superlinearly with the number of compute blocks supported. Regardless of the metalization offered, our designs will decrease in density with increasing gate count.

2.2 Segmentation

Modern designs, both in practice and in academic studies use segments which span more than one switchbox (See Figure 2). For example, a recent result from Betz suggests that length 4–8 buffered segments require less area than alternatives [2]. The important thing to notice is that any **fixed** segmentation scheme only changes the constants and not the asymptotic growth factor in Equation 9. In particular, using a single segmentation scheme of length L_{seg} will change Equation 2 to:

$$S_{sw} = \left(\frac{1}{L_{seg}}\right)(2)\, F_s \cdot W = \left(\frac{2}{L_{seg}}\right) F_s \cdot W \tag{10}$$

In practice the W will be different between the segmented and non-segmented cases, with the segmented cases requiring larger W's, but the asymptotic lower bound relationship on W derived above still holds. Similarly, a mixed segmentation scheme will also change the constants, but not the asymptotic requirements.

2.3 Hierarchical

A strictly hierarchical segmentation scheme might allow us to reduce the switchbox switches. Consider, that we have a base number of wire channels W_b, and populate the channel with W_b single length segments, W_b length 2 segments, W_b length 4 segments, and so forth. Using Equation 10

with W_{lb} in for W and summing across the geometric wire lengths, we see the total number of switches needed per switchbox is:

$$\begin{aligned} S_{sw} &= \left(\sum_{L_{seg}=1}^{N_{level}} \left(\frac{1}{L_{seg}} \right) \right) (2)\, F_s \cdot W_b \\ &= \left(\frac{1}{1} + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots \right) \cdot 2 \cdot F_s \cdot W_b \\ &\leq 4 \cdot F_s \cdot W_b \end{aligned} \quad (11)$$

The total wire width of a channel is now:

$$W = N_{level} \cdot W_b \quad (12)$$

For sufficiently large N_{level}, we can raise W to the required bisection width. Since S_{sw} in this hierarchical case does not, asymptotically, depend on N_{level}, the number of switches converges to a constant.

However, we should note this still does not change the asymptotic switch requirements, since the switch requirements depend on both the C-box switches and the S-box switches. As long as the C-box switches continue to connect to a constant fraction of W and not W_b, the C-box contribution to the total number of switches per compute block (Equation 1) continues to make the total number of switches linear in W and hence growing with N.

From this we see clearly that it is the flat connection of block IOs to the channel which ultimately impedes scalability.

2.4 Switch Dominated

Conventional experience implementing this style of interconnect has led people to observe that switch requirements tend to be limiting rather than wire requirements (*e.g.* [2]). Asymptotically, we see that an N-node FPGA will need:

$$N_{switch}(N) = B_{sw} \cdot N = O\left(N^{(p+0.5)}\right) \quad (13)$$

With BW wires in the bisection, we will require at least

$$A_{wire}(N) \geq \left(\frac{BW}{L/2} \right)^2 = O\left(\frac{N^{2p}}{L^2} \right) \quad (14)$$

For a fixed number of wire layers (L), this says wiring requirements grow slightly faster than switches (*i.e.*, when $p > 0.5$, $2p > p + 0.5$). Asymptotically, this suggests that if the number of layers, L grows as fast as $O(N^{\left(\frac{2p-1}{4}\right)})$, then we will remain switch dominated. Since switches have a much larger constant contribution than wires, it is not surprising that designs require a large N for these asymptotic effects to become apparent.

3. MESH OF TREES

The asymptotic analysis in the preceding section says that it is necessary to bound the compute block connections to a constant if we hope to contain the total switches per compute block to a constant independent of design size. Leighton's Mesh-of-Trees (MoT) network [15] [14] is a topology which does just that. Simply containing the switches to a constant is necessary but not sufficient to exploit additional metal layers. Later in this section, we also show that the MoT topology can be wired within a constant layout area per compute block.

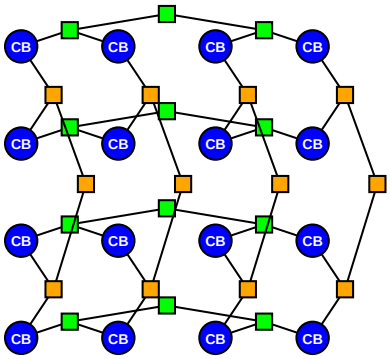


Figure 3: Basic Mesh-of-Trees (MoT) Topology

3.1 Basic Arrangement

In the MoT arrangement we build a tree along each row and column of the grid of compute elements (See Figure 3). For now, we will assume the tree is binary, but we can certainly vary the arity of the tree as one of the design parameters. The compute blocks connect only to the lowest level of the tree. Connection can then climb the tree in order to get to longer segments. We can place multiple such trees along each row or column to increase the routing capacity of the network. Each compute block is simply connected to the leaves of the set of horizontal and vertical trees which land at its site. We can parameterize the way the compute block connects to the leaf channels in a manner similar to the Manhattan C-box connections above.

We will use the parameter C to denote the number of trees which we use in each row and column. The C-box connections at each "channel" in this topology are made only to the C wires which exist at the leaf of the tree.

$$C_{sw} = 2F_c \cdot I \cdot C \quad (15)$$

In the simplest sense, we do not have switch boxes in this topology. At the leaf level, we allow connections between horizontal and vertical trees. Typically, we consider allowing each horizontal channel to connect to a single vertical channel in a domain style similar to that used in typical Manhattan switchboxes. This gives:

$$HV_{sw} = C \quad (16)$$

It would also be possible to fully populate this corner turn, allowing any horizontal tree to connect to any vertical tree at points of leaf intersection without changing the asymptotic switch requirements.

$$HV_{sw} = C^2 \quad (17)$$

Within each row or column tree, we need a switch to connect each lower channel to its parent channel. This can be as simple as a single pass transistor and associated memory cell. Amortizing across the compute blocks which share a single tree, per compute block we need a total of:

$$T_{sw} = 1 + \frac{1}{2} + \frac{1}{4} + ... < 2 \quad (18)$$

The horizontal channel holds C such trees, as does the vertical channel. Thus, each compute block needs:

$$\begin{aligned} B_{sw} &= C_{sw} + HV_{sw} + 2 \cdot C \cdot T_{sw} \\ &< C_{sw} + HV_{sw} + 4 \cdot C \end{aligned} \quad (19)$$

Leaf Span	Wires	Switches per Wire	Switches per Endpoint
1	1	2	2
2	2	1	1
4	2	2	1
8	4	1	$\frac{1}{2}$
16	4	2	$\frac{1}{2}$
32	8	1	$\frac{1}{4}$
64	8	2	$\frac{1}{4}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 1: Switches/Node for $p = 0.75$ MoT

Using the linear corner turn population (Eq. 16):

$$\begin{aligned} B_{sw} &< 2 \cdot F_c \cdot I \cdot C + 5 \cdot C \\ &< (2 \cdot F_c \cdot I + 5) \cdot C \end{aligned} \tag{20}$$

Assuming we can hold C bounded with increasing design size, this leaves us with a constant number of switches per compute block.

Tree Growth The strict binary tree we have shown correspondents to $p = 0.5$. To accommodate larger p values, it is necessary to grow the number of parents in the tree. Returning to Equation 8, we need $W = CN^{(p-0.5)}$. We can arrange to support a larger p with the mesh of trees by increasing the stage-to-stage growth rate.

For example, if alternate tree levels double the number of parent segments, we can achieve $p = 0.75$ (See Figure 7). The number of tree levels is $\log_2$ of the length of each row or column, which is $\sqrt{N}$. The number of channels composing the root level of each tree will thus be:

$$N_{ch}(N) = 2^{\left(\log_2\left(\sqrt{N}\right)/2\right)} = 2^{\left(\log_2\left(\sqrt[4]{N}\right)\right)} = \sqrt[4]{N} \tag{21}$$

The total bisection width at this level is the aggregate channel capacity across all $\sqrt{N}$ channels across the chip:

$$W_{bisect} = \sqrt{N} \cdot N_{ch}(N) \tag{22}$$

In this case that becomes:

$$W_{bisect} = \sqrt{N} \cdot \sqrt[4]{N} = N^{0.75} \tag{23}$$

That is, this growth is equivalent to providing: $p = 0.75$.

Note, however, that even though we increased the rate of wire growth, the total number of switches per node remain asymptotically constant (See Table 1):

$$T_{sw} = 2 + 1 + 1 + \frac{1}{2} + \frac{1}{2} + \frac{1}{4} + \frac{1}{4} ... < 6 \tag{24}$$

Which makes:

$$B_{sw}(p = 0.75) < (2 \cdot F_c \cdot I + 13) \cdot C \tag{25}$$

This property holds for any $p < 1.0$. That is, given sufficiently large N, we can approximate any p by programming the stage-to-stage growth rate, and the number of switches per compute block remains asymptotically constant. The particular constant grows with p as this example suggests. For arbitrary design bisection width, we can pick a p that is equal or greater to the design p, and a network with constant switches per endpoint can provide that much bisection bandwidth.

We are thus able to satisfy the lower bound relationship (Equation 8) introduced in the previous section with constant switches per compute block. However, the lower bound relationship only guarantees that we have sufficient wires in the bisection, *if we can use them*. The population scheme will determine whether or not enough of the wires can be used to keep C bound to a constant. At this point, we have no proof of the sufficiency of the population, so we employ empirical experiments, reported in Section 4, to assess the sufficiency of this population scheme.

3.2 Basic Layout

Constant switches per endpoint was necessary to show that we could layout the network in area linear in the number of compute blocks. However, it is not sufficient to show that we can use additional wire layers to achieve a compact layout. For unconstrained logic, it is not clear that more wire layers will always be usable. For example, [17] argues that wiring on an upper layer metal plane will occupy 12-15% of all the layers below it. Integrating this result across wire planes, this argues a useful limit of 6–7 wiring levels. The MoT wiring topology, however, is quite stylized with geometrically increasing wire lengths. Consequently, it does not exhibit the same saturation effect which we would get with unconstrained netlists. In fact, we can show that a design which needs $O(f(N))$ bisection bandwidth can be layed out with only $O(\max(f(N)/\sqrt{N}, 1))$ wiring layers.

Binary Tree ($p = 0.5$) To build intuition, let us focus initially on the binary tree case ($p = 0.5$). The key observation is that we can layout each binary tree along its row (or column) using $O(\log(l_{row}))$ wiring layers in a strip which is $O(1)$ wide and runs the length of the row ($l_{row} = \sqrt{N}$).

Figure 4 shows how the row (column) tree is mapped into a one-dimensional layout with $O(\log(N))$ wiring layers. It is important to notice that each subtree layout leaves one free switch location for an upper level switch. When we combine two subtrees, we can place the switch connecting them in one of the two free slots, leaving a single slot free in the resulting subtree. In this manner, the recursive composition of subtrees can continue indefinitely; the geometrically increasing via spacing allows it to avoid ever running out of via area on the lower levels of metalization. As shown, each new tree level simply adds one additional wire run above the existing wires. This $p = 0.5$ case requires $\Theta(\log(N))$ metal layers, which is asymptotically optimal to accommodate the $\log(N)$ wires which each tree contributes to each row or column. Note that if we make the width of the column as wide as a via and a wire, we can bring all the wires up to the appropriate metal layer without interfering with the column wire runs (See the "Top View" in Figure 4).

In practice, the width of a switch is likely to be several wire pitches wide, consequently, we can place several tree levels in a single metal layer and run them within the width of the switch row; this means that the number of wire layers we need for each row (or column) layout in practice is $\log_2(\sqrt{N})/r$ where r is the ratio of the switch width to wire pitch (strictly speaking one less than that to accommodate the via row). For example if the switch width is 50λ and the wire pitch is 8λ, we can put 6 wires within the width of the switch. If we use one track for vias, this means we can place 5 tree levels on each wire layer, so the number of layers needed to accommodate the row (column) tree is $\left\lceil \log_2(\sqrt{N})/5 \right\rceil$.

The full MoT structures requires both row and column trees. We must space out the row and column switches to

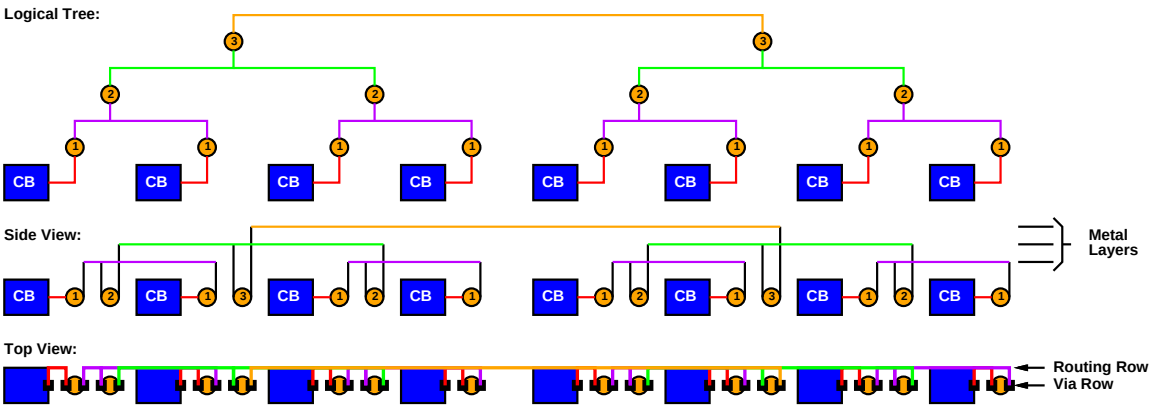


Figure 4: One-Dimensional Binary Tree Layout

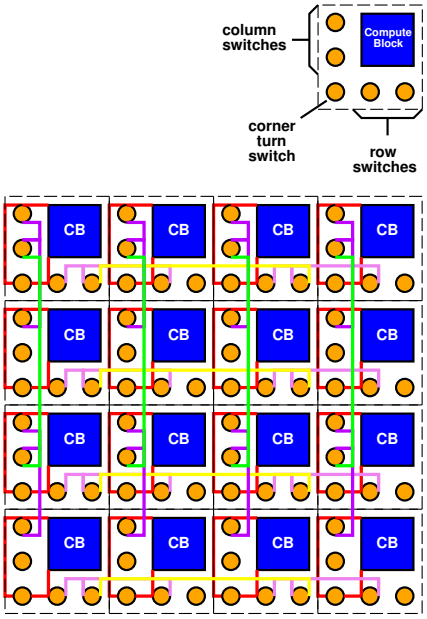


Figure 5: Minimal MoT Layout

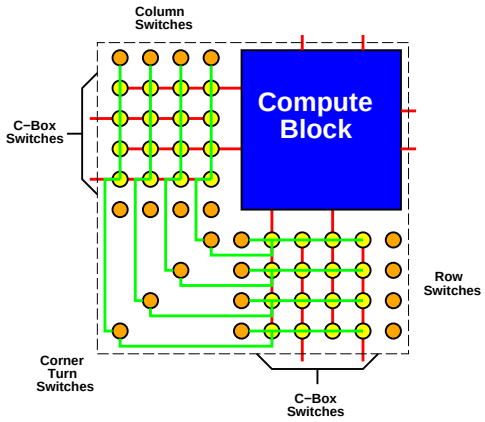


Figure 6: MoT Tile with $C = 4$ and $I = 2$

accommodate the cross switches. Further, we must assign separate wire layers for the rows and columns. Together, this means we will need $\lceil 2 \cdot \log_2(\sqrt{N})/r \rceil$ layers for wiring. In practice, additional wiring layers will be needed for power, ground, and clock routing.

Figure 5 shows a minimal layout with a single tree in each row and column channel. In practice, we will typically use several trees ($C > 1$) in each row and column and require C-box switches. Figure 6 shows the base tile for a larger network configuration.

Fatter Trees ($p > 0.5$) This same basic layout scheme works for the case where $0.5 \leq p < 1.0$. We will not always have exactly half as many switches on each immediately successive tree level. However, as long as $p < 1.0$, there are a number of tree stages over which the number of switches will be half the number of switches in the preceding group of tree stages. By grouping the switches into these groups, we can use the same strategy shown for the binary tree case.

Figure 7 shows the switch arrangement for the aforementioned $p = 0.75$ case. It should be clear from the layout tree diagrams that the switches can be shuffled to the base layer as in Figure 4. Here, we will, asymptotically, end up with 6 switches between every pair of compute blocks (Equation 24). Up to a span of 16 endpoints, we need 5 switches (Table 1). Beyond that, each pair of stages contributes half as many switches as the previous pair of stage, resulting in a total of one more switch per endpoint. You can begin to see that as we compose each additional pair of stages we end up leaving half of the remaining slots in each span with space for switches from the next span. This filling can continue indefinitely just as the $p = 0.5$ filling we have already seen. Further notice that the total number of metal layers is asymptotically optimal. That is, for $p > 0.5$, the number of layers required is $\Theta\left(N^{(p-0.5)}\right)$.

3.3 Variations

Upper Level Corner Turns We can add some corner turns at higher levels of the tree hierarchy, but we must be careful to maintain the property that each compute block tile contains a constant number of switches independent of the design size. If we allow every level to connect at every switch box, we will clearly ended up with too many switches ($O\left(N^{2p}\right)$ per compute block when $p > 0.5$).

We can, however, afford to place corner turns between the wire segments whose switch connection are associated with the same endpoint node. That is, we have already guaranteed that we can distribute the switches in each row and column such that there are a constant number of switches associated with each leaf node. Now, if we simply connect among those segments which switch at the same node, we, at most, increase the constant switch count at each node (See

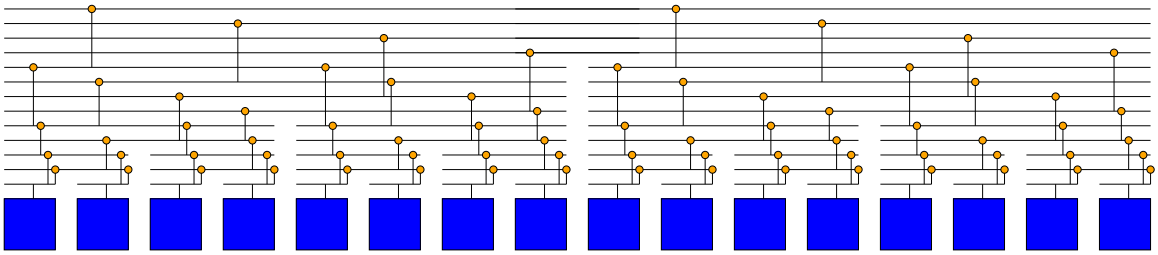


Figure 7: Row/Column Tree Growth to Achieve $p = 0.75$

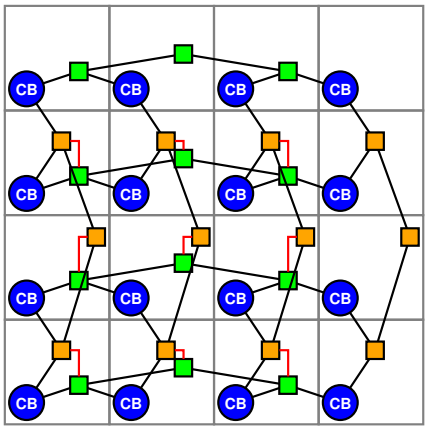


Figure 8: MoT with Non-Leaf Corner Turns

Figure 8). Likely, we would simply place a single switch between the horizontal and vertical segments in the same tree domain making up links at this stage; this way we have only three switches where we had two before. This makes our switch equation now:

$$B_{sw} = C_{sw} + HV_{sw} + 3 \cdot C \cdot T_{sw} \quad (26)$$

Shortcuts The breaks between tree segments create discontinuities in the array where leaves are physically close but logically in different subtrees. It also leads to bandwidth discontinuities along each row and column. For $p > 0.5$, these discontinuities do not affect the asymptotic wiring requirements, but may affect the practical wiring requirements by a constant factor. For example, returning to our $p = 0.75$ example, the root bandwidth for a row or column tree grows as $\sqrt[4]{N}$ (Equation 21). Now, if we consider all the channels at all levels, we simply have:

$$\begin{aligned} N_{total}(N) &= N_{ch}(N) + N_{ch}\left(\frac{N}{4}\right) + N_{ch}\left(\frac{N}{16}\right) + \cdots \\ &= \sqrt[4]{N} + \sqrt[4]{\frac{N}{4}} + \sqrt[4]{\frac{N}{16}} + \cdots \\ &= \sqrt[4]{N}\left(1 + \frac{1}{\sqrt{2}} + \frac{1}{2} + \frac{1}{2\sqrt{2}} + \cdots\right) \\ &\leq N_{ch}(N)\left(\frac{1}{1 - \frac{1}{\sqrt{2}}}\right) \\ &< 3.5 N_{ch}(N) \end{aligned}$$

As this shows, it can be a non-trivial constant factor. Shortcuts can also shorten wire runs.

We can add a single switchpoint between each pair of adjacent segments in the same tree at the same level of the hierarchy without changing the asymptotic switch requirements (See Figure 9). Note that we simply add one shortcut switch next to each tree switch, so our established layout scheme serves to accommodate these switches as well. From this it is clear that the shortcut switches simply add another T_{sw} horizontal and vertical switches to each compute block. Once added, all things which are physically close are also logically close and there are no bandwidth discontinuities in the array. It seems unlikely, however, that these shortcuts are needed on all trees in a row and column and on all tree levels. Staggering the trees within the same row or column may also reduce the need for shortcut connections.

Islands We can group multiple LUTs into each leaf compute block in the Island Style [4]. This does not change the asymptotic switching and wiring requirements for either the Manhattan or MoT wiring topology, but it may change the switching constants in important ways.

Buffered/Registered Switchpoints It is also possible to use buffered or registered switch points with this topology without harming the asymptotic switching and wiring requirements established above. We can group together children, shortcut, and parent connections into a single, local switching block. We can thus drive point-to-point signals between blocks. The switching blocks can be placed so we have a single such block per compute block and the number of wiring layers remains the same as in the unbuffered case.

3.4 Switch Dominated

The asymptotic reduction in switching requirements compared to the Manhattan topology makes wiring requirements more likely to be a limiting factor. At the same time, however, this topology allows us to maximally use additional metal layers. As a consequence, the MoT designs will always be switch area dominated when given sufficient layers of interconnect.

3.5 Delay

Note that switch delay is asymptotically logarithmic in distance between the source and the destination. A route simply needs to climb the tree to the appropriate tree level to link to the destination row or column, then descend the tree. It is also worthwhile to note that the stub capacitance associated with each level of the tree is constant. That is, there are a constant number of switches (drivers or receivers) attached to each wire segment, regarless of its length. This is an important contrast with the flat, Manhattan connection scheme where the number of switches attached to a long wire is proportional to the length of the wire. An added benefit of the strict hierarchy is that it manages to minimize the switch capacitance associated with long wire runs.

Buffered switches are needed to achieve minimum delay and to isolate each wire segment from the fanout that may occur on a multipoint net.

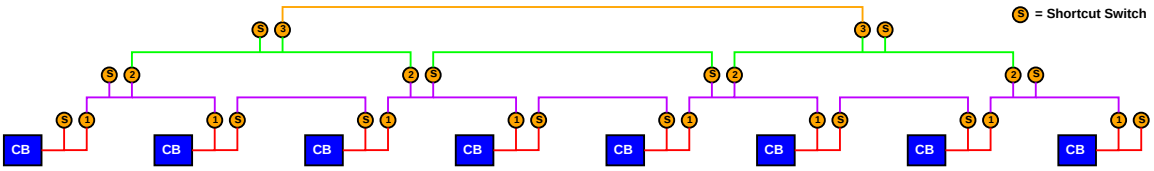


Figure 9: One-Dimensional Binary Tree with Shortcuts

3.6 Long Wire Runs

Ultimately, we will need to buffer the long wire runs in order to achieve linear delay with interconnect length and minimize the delay travelling long distances. This will end up forcing us to insert buffers at fixed distances which can reduce the benefits of the convenient geometric switching property identified. Technological advances that provided linear delay with distance without requiring repeaters (*e.g.* optical, superconducing wires) would obviate this problem.

3.7 Relation to Tree-of-Meshes

Both Agarwal [1] and Tsu [18] have previously described hierarchical FPGA interconnect architectures. DeHon showed that the Butterfly Fat-Tree style interconnect of the HSRA could also be layed out in constant area given sufficient wire layers for the $p = 0.5$ case [9]. These networks all build a single, unified hierarchy and are closely related to the Tree-of-Meshes topology [14]. In constract, the Mesh-of-Trees used here is directly a two-dimensional structure building hierarchical routing along each row and column. As such, the MoT can be viewed as a hybrid between the strict, single hierarchy of the Tree-of-Meshes and the non-hierarchical Manhattan array. Fully understanding the implications of the differences between the Tree-of-Meshes and the Mesh-of-Trees remains a matter for future work.

4. EMPIRICAL EXPERIMENTS

In the previous section we demonstrated the favorable asymptotic switching requirements for the MoT design assuming we can contain the number of required base-channels to a suitably small constant. In this section we show empirically that the base channel requirements are uniformly small. Further, we show that even for the small sizes of conventional FPGA benchmarks, the MoT scheme already shows some practical advantages in reducing aggregate switch requirements. We explore many of the design variations introduces in Section 3.

4.1 Base Comparison

For a base level comparison, we use the benchmarks from Toronto's "FPGA Place and Route Challenge" [3] to compare the channel, domain, and switch requirements between the traditional Manhattan routing topology and our MoT topology. We used the `vpr422_challenge_arch` architecture as the baseline mesh; this has single-length segments and a single LUT per Island. We substitute a universal switch [8] for the subset switch used in the vpr422 challenge because the routed mesh designs using universal switches uniformly require less switches than the subset-switch-based designs. Each of the 4 LUT inputs appears on a single side of the logic block ($T_{in} = 1$), and the output appears on two sides ($T_{out} = 2$); both are fully populated ($F_c = 1$) (See Figure 10). We use VPR 4.3 to produce the placed designs for **both** the Manhattan and MoT routing. We use the channel minimizing VPR 4.3 router to route the Manhattan

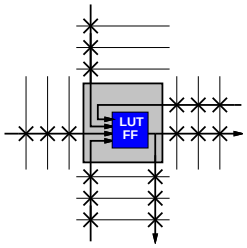


Figure 10: Logic Block IO Structure from VPR422 Challenge (Shown $W = 3$)

designs. Since prior work suggested the superiority of longer segments [2] [4], we also routed a uniform, length-4 segment Manhattan case for comparison; all other parameters are identical to the base length-1 Manhattan case.

For our overall comparison, we assembled a MoT design with $T = 1$ (see Figure 11), upper-level corner turns, and no shortcuts. We developed our own, Pathfinder-based [16] router to route the MoT designs. To match the VPR-style results, we let the number of base channels, C, float and report the minimum number of channels required to route the design for various p values.

Table 2 summarizes these basic results. For almost all designs, the MoT routes with sufficiently small C as to require fewer total switches than the Manhattan designs.

Small C's The C's are uniformly small, many being as low as 3 for $p = 0.75$. Increasing IO population (Table 3), shortcuts (Table 5), and staggering (Table 6) reduce most of the remaining cases to 3 or 4 as well. The C required for the design is driven by three things:

1. Bisection bandwidth
2. Number of distinct signals which must enter a channel
3. Domain coloring limitations

A sufficiently large p value can generally accommodate bisection needs (See Figure 12). For channel entrance, note that a fully used k-LUT with a single output needs to have $k+1$ potentially distinct signals enter one of the four channels which surrounds it. Further note that it shares each of those channels with 2 other k-LUTs which have similar requirements. Consequently, the channel entrance lower bound is:

$$C_{lb} \geq \left\lceil \frac{2 \cdot (k+1)}{4} \right\rceil \tag{27}$$

For $k = 4$, $C_{lb} = 3$. Finally, since the Mesh-of-Tree design described here maintains the domain topology typical of Manhattan FPGA interconnect, it could have colorability limitations [19]. The routed results suggest that the colorability issues are not a major issue in practice as we achieve within one channel of the channel entrance lower bound on all designs.

4.2 Variations

IO Population and Distribution We considered population schemes from fully connecting the IOs to each chan-

Design		Manhattan (universal)				Mesh of Trees								
		$L_{seg}=1$		$L_{seg}=4$		$p=0.67$			$p=0.67$ stagger			$p=0.75$		
Circuit	**#LB**	W	B_{sw}	W	B_{sw}	C	B_{sw}	$\Delta\%$	C	B_{sw}	$\Delta\%$	C	B_{sw}	$\Delta\%$
alu4	1522	9	121	13	121	4	96	-21	4	102	-16	4	112	-7
apex2	1878	10	133	14	129	5	116	-9	5	125	-2	4	107	-17
apex4	1262	11	150	15	142	5	123	-13	5	134	-5	5	145	+1
bigkey	1707	6	79	10	91	3	67	-14	3	71	-9	3	77	-1
clma	8382	10	126	14	122	5	108	-11	4	89	-26	4	101	-17
des	1591	7	91	10	89	3	70	-22	3	71	-20	3	84	-6
diffeq	1497	6	81	9	84	4	97	+20	3	77	-5	3	86	+5
dsip	1370	6	79	10	91	3	67	-14	3	71	-9	3	77	-1
elliptic	3604	9	117	12	108	4	87	-18	4	92	-14	4	100	-7
ex1010	4598	10	129	13	115	5	114	-1	4	96	-17	4	108	-6
ex5p	1064	12	165	15	144	5	128	-10	5	140	-2	4	120	-16
frisc	3556	11	143	14	126	5	110	-12	4	92	-26	4	100	-20
misex3	1397	10	135	13	122	5	120	-1	4	104	-15	4	111	-8
pdc	4575	14	180	18	160	6	136	-14	5	121	-24	5	136	-14
s298	1931	6	80	10	92	4	92	+14	4	99	+23	3	80	0
s38417	6406	7	89	10	88	4	88	0	4	93	+5	3	78	-10
s38584.1	6446	7	89	10	88	4	88	0	4	93	+5	4	105	+19
seq	1750	10	134	13	121	5	118	-2	5	128	+6	4	108	-10
spla	3690	12	155	16	143	6	131	-8	5	116	-18	5	125	-12
tseng	1047	6	83	9	86	4	102	+22	3	83	0	3	90	+8
sum		179	2359	248	2262	89	2058	-9	81	1997	-11	76	2050	-9

[$\Delta\%$ is relative to the best of the two mesh cases.]

Table 2: Manhattan vs. Mesh-of-Trees

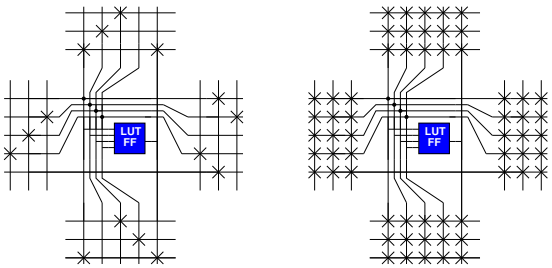


Figure 11: Distributed IO Population with T=1 [left] and T=4 [right] (Shown $W=3$)

nel down to connecting each base channel domain only once. That is, we decided to connect each input or output with $T \times C$ switches, and to balance those switches over both sides and base channel domains (See Figure 11). We also considered the IO configuration used in the vpr422 challenge architecture [3] (See Figure 10). Table 3 summarize the results for the $p=0.67$ case. The $T=1$ case where we rotate the channel connections around the four sides of the block generally achieves the minimum switch count. At the expense of additional switches, the higher T values can be used to reduce the number of base channels required by the design.

Growth Rates As noted previously, larger p's will imply greater bisection bandwidth for a given base channel size and greater switches. Increasing p will tend to decrease C. For a given design the question is whether the decrease in base channels is sufficient to compensate for the increased switch requirements per channel for the larger p value. Tables 4 summarizes these effects for $p=0.50$, $p=0.67$, and $p=0.75$. In general, we expect that exactly matching p for the MoT with the p for the placed design will be the minimum point. Since the designs likely have different placed p's, it is not surprising they are minimized by different p values.

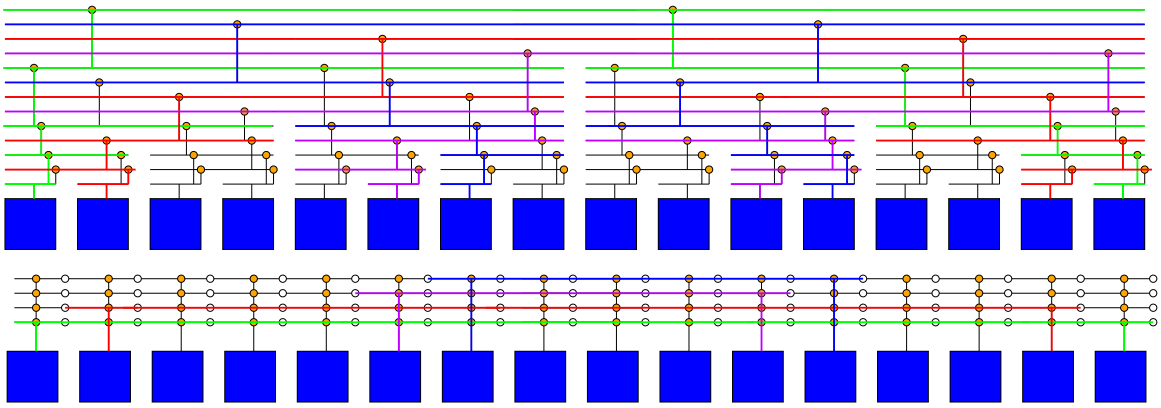
Design	C				Switches			
	VPR	Dist. T			VPR	Distributed T		
Circuit	chlg	1	2	3	chlg	1	2	3
alu4	5	4	4	4	128	96	119	142
apex2	5	5	5	4	124	116	145	138
apex4	6	5	5	5	156	123	152	182
bigkey	3	3	3	3	72	67	84	100
clma	5	5	5	4	115	108	134	128
des	4	3	3	3	99	70	86	103
diffeq	4	4	3	3	104	97	91	108
dsip	3	3	3	3	72	67	84	100
elliptic	5	4	4	4	117	87	109	131
ex1010	5	5	4	4	121	114	112	134
ex5p	6	5	5	5	162	128	158	188
frisc	5	5	4	4	118	110	110	132
misex3	5	5	4	4	128	120	119	142
pdc	7	6	6	6	169	136	169	201
s298	4	4	3	3	99	92	86	103
s38417	4	4	3	3	94	88	82	98
s38584.1	4	4	3	3	94	88	82	98
seq	5	5	5	4	126	118	147	139
spla	6	6	5	5	140	131	137	165
tseng	4	4	3	3	109	102	94	112

Table 3: Effects of LB IO Pop. and Dist. ($p=0.67$)

Nonetheless, as Table 2 shows, both $p=0.67$ and $p=0.75$ are superior to the mesh layout for most designs.

Corner Turns Including upper level corner turns does reduce the number of base channels required. However, the total number of switches required is roughly the same in both cases.

Shortcuts Including shortcuts will reduce the number of base channels (Table 5), but the additional switches per logic block are not sufficiently compensated by the reduction in channels. Consequently fully populated shortcuts result



Shown here are 1D slices of a $p = 0.75$ MoT (top) and a Flat Manhattan (bottom) topology. The MoT accommodates the bisection width of 4 using only a single base domain, while the Manhattan topology requires at least one domain for every wire in the bisection; this demonstrates how the MoT can often get away with a smaller C than the Manhattan channel width (W). Asymptotically, the MoT will require 6 switches per endpoint for this arrangement, while the Manhattan requires 8 to accommodate this channel width of 4. For larger spans, the effect increases. For a span of 32 nodes, the MoT can accommodate a bisection bandwidth of 8 while still using at most 6 switches per endpoint; the mesh with a bisection width of 8 will require 16 switches per endpoint.

Figure 12: Bisection Width Comparison

	C			Switches		
Design	p			p		
Circuit	0.50	0.67	0.75	0.50	0.67	0.75
alu4	9	4	4	130	96	112
apex2	10	5	4	141	116	107
apex4	11	5	5	163	123	145
bigkey	4	3	3	54	67	77
clma	11	5	4	137	108	101
des	5	3	3	68	70	84
diffeq	6	4	3	88	97	86
dsip	5	3	3	68	67	77
elliptic	8	4	4	101	87	100
ex1010	10	5	4	127	114	108
ex5p	11	5	4	167	128	120
frisc	10	5	4	135	110	100
misex3	9	5	4	131	120	111
pdc	15	6	5	202	136	136
s298	6	4	3	84	92	80
s38417	6	4	3	79	88	78
s38584.1	7	4	4	93	88	105
seq	10	5	4	143	118	108
spla	12	6	5	151	131	125
tseng	5	4	3	76	102	90

Table 4: Effects of p for $T = 1$ Distributed Population

in a net increase in switching requirements.

Staggering In Table 6 we show the effects of staggering the base channel domains with respect to each other. As noted in Section 3.3, breaks between trees yield bandwidth discontinuities. Since we have more than one base channel in each row and column, we have the opportunity to offset them from each other to minimize discontinuity effects. For 8 of the designs, staggering saves a base channel, saving us 10-20% in switch count. For other designs, tree alignment issues end up costing us a couple of extra switches per domain. An open question is whether or not it is possible to get the benefits of staggering without paying this additional cost.

Design		No Short		Short			
Circuit	**#LBs**	C	B_{sw}	C	$\Delta\%$	B_{sw}	$\Delta\%$
alu4	1522	4	96	4	0	153	+60
apex2	1878	5	116	4	-20	148	+27
apex4	1262	5	123	4	-20	156	+26
bigkey	1707	3	67	3	0	108	+60
clma	8382	5	108	4	-20	138	+28
des	1591	3	70	3	0	113	+61
diffeq	1497	4	97	3	-25	117	+20
dsip	1370	3	67	3	0	108	+60
elliptic	3604	4	87	4	0	141	+61
ex1010	4598	5	114	4	-20	145	+28
ex5p	1064	5	128	4	-20	162	+26
frisc	3556	5	110	4	-20	132	+19
misex3	1397	5	120	4	-20	152	+26
pdc	4575	6	136	4	-33	145	+6
s298	1931	4	92	3	-25	111	+20
s38417	6406	4	88	3	-25	107	+21
s38584.1	6446	4	88	3	-25	107	+21
seq	1750	5	118	4	-20	149	+26
spla	3690	6	131	4	-33	141	+7
tseng	1047	4	102	3	-25	122	+19

Table 5: Effects of Shortcuts ($p = 0.67$)

5. SUMMARY AND FUTURE WORK

Using the Mesh-of-Trees topology, we can achieve better scalability than a flat, Manhattan topology. Assuming the number of base channels, C, remains constant for increasing design size, the total number of switches per LUT in our MoT converges to a constant $[O(1)]$ independent of design size; this should be contrasted with the $O(N^{p-0.5})$ switches per LUT required for a flat, Manhattan topology. Given sufficient wiring layers, the MoT network layout can maintain a constant area per logic block as the design scales up. Asymptotically, the number of switches in any path in the MoT needs to only grow as $O(\log(N))$. Our initial empirical experiments verify small C values that show no signs of growing with design size, and total switch requirements that are 10% smaller than those of conventional Mesh designs.

Design		No Stagger		Stagger			
Circuit	**#LBs**	C	B_{sw}	C	$\Delta\%$	B_{sw}	$\Delta\%$
alu4	1522	4	96	4	0	102	+6
apex2	1878	5	116	5	0	125	+7
apex4	1262	5	123	5	0	134	+8
bigkey	1707	3	67	3	0	71	+5
clma	8382	5	108	4	-20	89	-17
des	1591	3	70	3	0	71	+1
diffeq	1497	4	97	3	-25	77	-21
dsip	1370	3	67	3	0	71	+5
elliptic	3604	4	87	4	0	92	+5
ex1010	4598	5	114	4	-20	96	-15
ex5p	1064	5	128	5	0	140	+9
frisc	3556	5	110	4	-20	92	-16
misex3	1397	5	120	4	-20	104	-13
pdc	4575	6	136	5	-16	121	-10
s298	1931	4	92	4	0	99	+7
s38417	6406	4	88	4	0	93	+5
s38584.1	6446	4	88	4	0	93	+5
seq	1750	5	118	5	0	128	+8
spla	3690	6	131	5	-16	116	-11
tseng	1047	4	102	3	-25	83	-18

Table 6: Effects of Staggering ($p = 0.67$)

Future In this paper, we have explored many of the parameters associated with designing MoT networks but many more design parameters deserve additional study. We limited ourselves to binary trees here; it will be useful to better understand and quantify the tradeoffs associated with higher arity trees. We limited these studies to logic blocks holding a single LUT; it will be interesting to see how Island-style clustering interacts with this topology. We expect larger benchmarks will better demonstrate the scalability of this architecture. A more careful review of timing effects would also be beneficial.

6. ACKNOWLEDGMENTS

This research was funded in part by the DARPA Moletronics program under grant ONR N00014-01-0651 and by the NSF CAREER program under grant CCR-0133102.

7. REFERENCES

[1] A. A. Agarwal and D. Lewis. Routing Architectures for Hierarchical Field Programmable Gate Arrays. In *Proceedings 1994 IEEE International Conference on Computer Design*, pages 475–478. IEEE, October 1994.

[2] V. Betz and J. Rose. FPGA Routing Architecture: Segmentation and Buffering to Optimize Speed and Density. In *Proceedings of the 1999 International Symposium on Field Programmable Gate Arrays (FPGA'99)*, pages 59–68, February 1999.

[3] V. Betz and J. Rose. FPGA Place-and-Route Challenge. <`http://www.eecg.toronto.edu/~vaughn/challenge/challenge.html`>, 1999.

[4] V. Betz, J. Rose, and A. Marquardt. *Architecture and CAD for Deep-Submicron FPGAs*. Kluwer Academic Publishers, 101 Philip Drive, Assinippi Park, Norwell, Massachusetts, 02061 USA, 1999.

[5] M. Bohr. Interconnect Scaling – The Real Limiter to High Performance ULSI. In *International Electron Devices Meeting 1995 Technical Digest*, pages 241–244. Electron Devices Society of IEEE, December 1995.

[6] S. D. Brown, R. J. Francis, J. Rose, and Z. G. Vranesic. *Field-Programmable Gate Arrays*. Kluwer Academic Publishers, 101 Philip Drive, Assinippi Park, Norwell, Massachusetts, 02061 USA, 1992.

[7] W. S. Carter, K. Duong, R. H. Freeman, H.-C. Hsieh, J. Y. Ja, J. E. Mahoney, L. T. Ngo, and S. L. Sze. A User Programmable Reconfigurable Logic Array. In *IEEE 1986 Custom Integrated Circuits Conference*, pages 233–235. IEEE, May 1986.

[8] Y.-W. Chang, D. F. Wong, and C. K. Wong. Universal Switch-Module Design for Symmetric-Array-Based FPGAs. In *Proceedings of the 1996 International Symposium on Field-Programmable Gate Arrays*, pages 80–86. ACM/SIGDA, February 1996.

[9] A. DeHon. Compact, Multilayer Layout for Butterfly Fat-Tree. In *Proceedings of the Twelfth ACM Symposium on Parallel Algorithms and Architectures (SPAA'2000)*, pages 206–215. ACM, July 2000.

[10] A. DeHon. Rent's Rule Based Switching Requirements. In *Proceedings of the System-Level Interconnect Prediction Workshop (SLIP'2001)*, pages 197–204. ACM, March 2001.

[11] W. E. Donath. Placement and Average Interconnection Lengths of Computer Logic. *IEEE Transactions on Circuits and Systems*, 26(4):272–277, April 1979.

[12] A. E. Gamal. Two-Dimensional Stochastic Model for Interconnections in Master Slice Integrated Circuits. *IEEE Transactions on Circuits and Systems*, 28(2):127–138, February 1981.

[13] B. S. Landman and R. L. Russo. On Pin Versus Block Relationship for Partitions of Logic Circuits. *IEEE Transactions on Computers*, 20:1469–1479, 1971.

[14] F. T. Leighton. New lower bound techniques for VLSI. In *Twethy-Second Annual Symposium on the Foundations of Computer Science*. IEEE, 1981.

[15] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. Morgan Kaufmann Publishers, Inc., 1992.

[16] L. McMurchie and C. Ebling. PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs. In *Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 111–117. ACM, February 1995.

[17] G. A. Sai-Halasz. Performance Trends in High-End Processors. *Proceedings of the IEEE*, 83(1):20–36, January 1995.

[18] W. Tsu, K. Macy, A. Joshi, R. Huang, N. Walker, T. Tung, O. Rowhani, V. George, J. Wawrzynek, and A. DeHon. HSRA: High-Speed, Hierarchical Synchronous Reconfigurable Array. In *Proceedings of the International Symposium on Field Programmable Gate Arrays*, pages 125–134, February 1999.

[19] Y.-L. Wu, S. Tsukiyama, and M. Marek-Sadowska. Graph Based Analysis of 2-D FPGA Routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15(1):33–44, January 1996.

APPENDIX

A. PROGRAMMING GROWTH RATES

In general, the easiest way to think about growth rates for 2-ary trees (as we limit ourselves to in this paper) is to allow each tree stage to have either one parent or two. If the tree stage has one parent, the number of wires in the tree at that stage is the same as the previous stage. If it has two, there are twice as many wires. Section 3.1 gave an example where alternating stages had two parents.

To design a particular growth rate, we choose a basic repeat sequence of 2-parent and 1-parent stages in the tree. For the aforementioned example, the repeat sequence was (2 1). By repeating this as needed [(2 1)*], we can build any size array and provide $p = 0.75$. In general, let's call the repeat sequence length L_R and the number of 2's in the repeat sequence N_2. For the (2 1)* sequence, we have $L_R = 2$ and $N_2 = 1$. For the (2 1 2 1 1)* sequence we have $L_R = 5$ and $N_2 = 2$.

We can generalize Equation 21, to be:

$$\begin{aligned} N_{channels}(N) &= 2^{\left(\log_2\left(\sqrt{N}\right)\left(\frac{N_2}{L_R}\right)\right)} = 2^{\left(\log_2(N)\left(\frac{N_2}{2\cdot L_R}\right)\right)} \\ &= N^{\frac{N_2}{2\cdot L_R}} \end{aligned} \quad (28)$$

This of course, is strictly true only at the boundaries of the repeat sequence; most importantly it captures the asymptotic growth behavior. Combining with Equation 22, we get:

$$W_{bisect} = \sqrt{N} \cdot N^{\frac{N_2}{2\cdot L_R}} \quad (29)$$

Combining with the Rent Relationship (Equation 7) (and dropping the constant we have been omitting), we have:

$$W_{bisect} = N^{\left(0.5+\frac{N_2}{2\cdot L_R}\right)} = N^p \quad (30)$$

From which we can conclude:

$$p = \frac{N_2}{2 \cdot L_R} + \frac{1}{2} \quad (31)$$

We can use this to conclude, our (2 1 2 1 1)* example, asymptotically provides $p = \frac{2}{10} + \frac{1}{2} = 0.7$.

B. UPPER LEVEL CORNER TURNS

Table 7 compares the effects of including or excluding the upper level corner turn.

Design		Upper		Base Only	
Circuit	**#LBs**	C	B_{sw}	C	B_{sw}
alu4	1522	4	96	6	109
apex2	1878	5	116	7	124
apex4	1262	5	123	7	130
bigkey	1707	3	67	4	68
clma	8382	5	108	7	115
des	1591	3	70	4	71
diffeq	1497	4	97	5	92
dsip	1370	3	67	4	68
elliptic	3604	4	87	6	101
ex1010	4598	5	114	6	103
ex5p	1064	5	128	7	135
frisc	3556	5	110	6	94
misex3	1397	5	120	6	109
pdc	4575	6	136	8	138
s298	1931	4	92	5	88
s38417	6406	4	88	5	84
s38584.1	6446	4	88	5	84
seq	1750	5	118	7	125
spla	3690	6	131	7	117
tseng	1047	4	102	4	77

Table 7: Upper-Level Corner Turn Effect ($p = 0.67$)