# Effect of the Prefabricated Routing Track Distribution on FPGA Area-Efficiency

Vaughn Betz and Jonathan Rose
Department of Electrical and Computer Engineering, University of Toronto
Toronto, Ontario, Canada M5S 3G4
{vaughn, jayar}@eecg.utoronto.ca

**Abstract**

*In most commercial Field-Programmable Gate Arrays (FPGAs) the number of wiring tracks in each channel is the same across the entire chip. A long-standing open question for both FPGAs and channelled gate arrays is whether or not some uneven distribution of routing tracks across the chip would lead to an area benefit. For example, many circuit designers intuitively believe that most congestion occurs near the center of a chip, and hence expect that having wider routing channels near the chip center would be beneficial.*

*In this paper we determine the relative area-efficiency of several different routing track distributions. We first investigate FPGAs in which horizontal and vertical channels contain different numbers of tracks in order to determine if such a directional bias provides a density advantage. Secondly, we examine routing track distributions in which the track capacities vary from channel to channel. We compare the area-efficiency of these non-uniform routing architectures to that of an FPGA with uniform channel capacities across the entire chip.*

*The main result is that the most area-efficient global routing architecture is one with uniform (or very nearly uniform) channel capacities across the entire chip in both the horizontal and vertical directions. This paper shows why this result, which is contrary to the intuition of many FPGA architects, is true. While a uniform routing architecture is the most area-efficient, several non-uniform and directionally-biased architectures are fairly area-efficient provided that appropriate choices are made for the pin positions on the logic blocks and the logic block array aspect ratio.*

## 1 Introduction

Field-Programmable Gate Arrays (FPGAs) have seen explosive market growth because they offer instant manufacturing and much lower non-recurring engineering costs than Mask-Programmed Gate Arrays. FPGAs enable fast manufacturing and low development costs because their logic and routing resources are prefabricated and the user implements a circuit by programming the appropriate connections between these prefabricated resources [1].

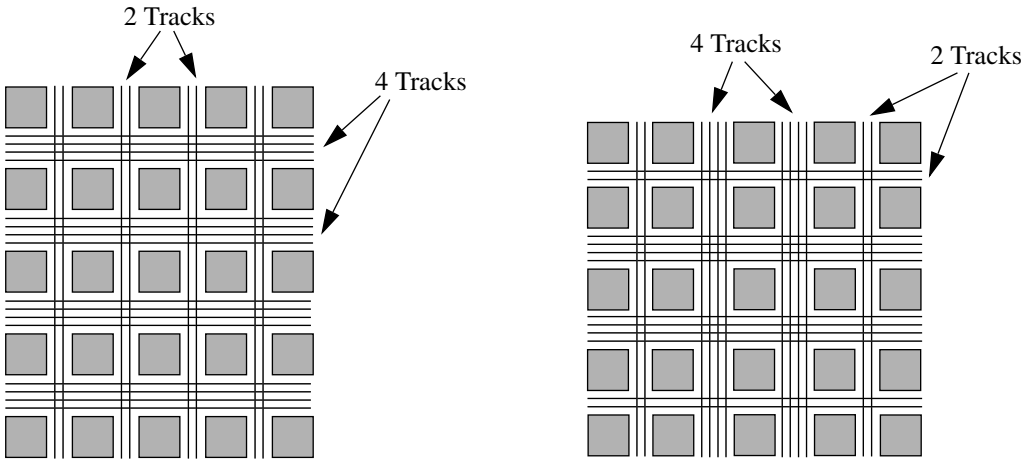
The prefabrication of routing resources implies that the number of routing tracks in each channel is set by the FPGA manufacturer. These routing resources should be distributed in a manner that permits their efficient utilization by the largest class of circuits. If there are too few tracks in some area of the chip then many circuits will be unroutable, while if there are too many tracks, they may be wasted. The relatively low density of FPGAs makes an area-efficient distribution of routing tracks essential.

This paper addresses two fundamental questions concerning the distribution of routing tracks across an

FPGA. First, should the number of tracks in the horizontal channels be different from the number in the vertical channels? Figure 1(a) depicts an FPGA with such a *directional bias*. In essence, we are investigating if there is an intrinsic property of circuits that makes a directional bias more area-efficient. If so, what amount of bias is best? Commercial FPGAs with both unbiased routing [2, 3] and biased routing [4, 5] exist, so this question has commercial relevance.

Second, should all routing channels in the same direction in an FPGA contain the same number of tracks or is a *non-uniform* routing architecture, in which some channels are wider than others, preferable? An example FPGA with a non-uniform routing architecture is shown in Figure 1(b).[1] Intuitively, such an architecture may facilitate routing in congested regions. However, if the wider channels cannot be used efficiently they will waste area. Many in the FPGA community believe that most routing congestion occurs near the center of an FPGA, and therefore expect that wider channels in this region would be beneficial. The Lucent Technologies ORCA 2C FPGAs employ a non-uniform routing architecture in which the center channel is wider than the others [6]. On the other hand, board-level constraints often force designers to fix the position of an FPGA's I/Os, and some believe that this increases congestion near the chip edges, and therefore the channel between the pads and the logic block array should be widened. The Xilinx 4000 and 5000 series FPGAs have a wide channel between the pads and logic, at least partially to improve routability when the I/O locations are fixed [7, 8]. In this paper, we determine the best distribution of tracks across an FPGA both when the I/O assignment to pads is unconstrained and when it is fixed in a poor configuration.

We evaluate FPGA architectures experimentally; benchmark circuits are placed and routed into




(a) Directionally-Biased (b) Non-Uniform

**Figure 1:** Types of Global Routing Architectures.

1. Note that any given channel will always have the same number of tracks along its entire length. We did not consider varying the channel capacity along its length as this makes it very difficult, and likely impractical, to lay out the FPGA.

FPGAs with different global routing architectures to determine the relative area consumed by the circuit in each architecture. To obtain meaningful results, the CAD tools used to place and route these circuits must take advantage of the biased and non-uniform nature of these architectures. Accordingly, we have created a new placement and routing tool which aggressively seeks to minimize congestion and fully utilize the channels of the specified architecture during both placement and global routing.

The organization of this paper is as follows. Section 2 outlines the experimental framework used to evaluate the different FPGA architectures. Section 3 describes the custom placement and routing CAD tools. We evaluate the area-efficiency of FPGAs with differing amounts of directional routing bias in Section 4. In Section 5 we address the uniform vs. non-uniform channel thickness question. Finally, we summarize our results and conclusions. Note that a less detailed examination of many of the issues discussed in this paper appeared in [9].

## 2 Experimental Methodology

To compare the area-efficiency of the different global routing architectures we technology-map, place and route 26 of the largest MCNC benchmark circuits [10] into each architecture. In this section we describe the CAD flow, the area-efficiency metric used to compare architectures, and several important architectural details.

### 2.1 CAD Flow

Figure 2 summarizes the CAD flow. First, SIS [11] is used to perform technology-independent logic

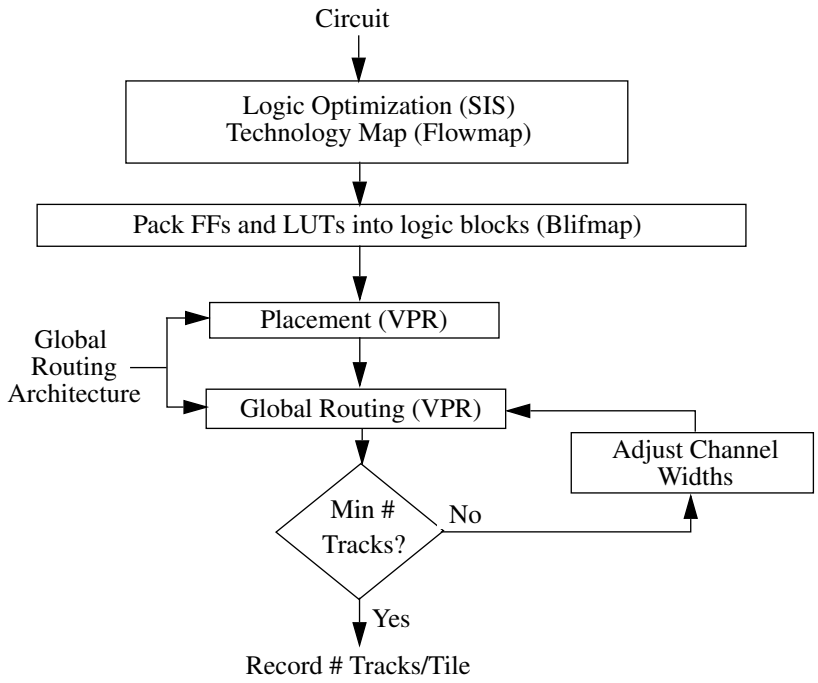



**Figure 2:** Architecture Evaluation Flow.

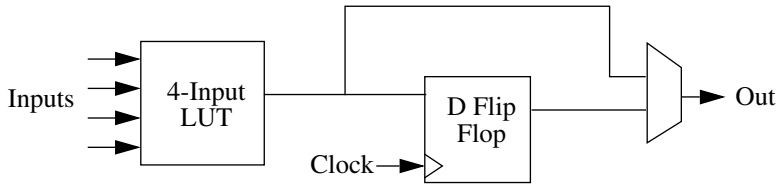



**Figure 3:** Logic Block Structure.

optimization on a circuit. Next this circuit is technology-mapped by Flowmap [12] into four-input look-up tables (4-LUTs) and flip flops. The logic block used in these experiments contains a 4-LUT and a flip-flop, as illustrated in Figure 3. A custom-built program (blifmap) packs the 4-LUTs and flip flops together into these logic blocks.

The netlist of logic blocks and a description of the FPGA global routing architecture are then read into the placement and global routing tool, VPR. This program places the circuit, and then repeatedly routes (or attempts to route) the circuit with different numbers of tracks in each channel (*channel capacities)*. VPR performs a binary search on the channel capacities, increasing them after a failed routing and reducing them after a successful one, until it finds the minimum number of tracks required for the circuit to route successfully on a given global routing architecture. Note that while the absolute number of tracks per channel is adjusted upwards or downwards after each attempted routing, the *relative* numbers of tracks in the various channels across the FPGA are always kept at the values specified by the FPGA architecture. For example, VPR's first attempt at routing a circuit in an architecture with a two-to-one directional bias might assume horizontal channel capacities of twelve tracks and vertical channel capacities of six tracks. If this routing was successful, VPR would next attempt to route the circuit in an FPGA with horizontal channel capacities of six tracks and vertical channel capacities of three tracks, and so on until the minimum number of tracks required for routing is determined.

The benchmark circuits used in this study consist of 14 combinational and 12 sequential MCNC benchmark circuits [8], which vary in size from 222 to 1878 of our logic blocks.

## 2.2 Area-Efficiency Metric

Our goal is to measure the area-efficiency of different global routing architectures without reference to the detailed routing architecture (e.g. segmentation and switch block topology). At this level, it is the amount of "global wiring" that changes as we vary the architecture. A simple track count will not accurately represent the wiring area of rectangular FPGAs, as the tracks in one direction are longer than those in the other. Accordingly, we define a *track segment* to be a prefabricated wire that spans one logic block; a channel of width W tracks that spans L logic blocks contains WL track segments. The total number of track segments an FPGA must contain to globally route a circuit is a representative metric of the "global wiring" area. In order to average the results from circuits of differing sizes we use the average number of

track segments per tile (i.e. per logic block) as our area measure. For example, in a square N x N uniform FPGA with W tracks in each channel, the total number of track segments is $2WN^2$, and the number of tracks per tile is 2W. Note that the routing area is given by the total number of track segments in the entire FPGA, and not the number of track segments which are actually used by a circuit.

### 2.3 Significant FPGA Architectural Details

Several architectural parameters other than the global routing architecture must be specified in order to define an FPGA. We set these parameters to be as close to those of commercial FPGAs as possible.

First, the size of the FPGA array used for a given circuit (i.e. the number of logic blocks) is set to be the *smallest* FPGA with the desired aspect ratio (number of columns / number of rows) with sufficient logic blocks to accommodate the circuit. This situation, in which there is minimal "spare room" in the FPGA, presents the greatest challenge to routing completion and is normally the case manufacturers wish to optimize.

In this study the number of I/O pads that can fit into the height or width of a logic block is set to two. This number is commensurate with the relative sizes of I/O pads and 4-LUTs in current FPGAs [2, 3, 5] and ensures that none of the 26 benchmarks is pad-limited.

Finally, we do not route the clock net (all the MCNC benchmarks use only a single clock) in sequential circuits, since this net is normally distributed through a special clocking network in commercial FPGAs.

## 3 Tuned Placement and Routing Algorithms

In FPGA architecture explorations of this kind [1] one must ensure that the CAD tools used are responsive to the architectural parameters being varied. To ensure a fair comparison between different global routing architectures, we created a new placement and global routing tool which directly exploits biased and non-uniform routing architectures. As this CAD tool is capable of mapping to a wide variety of FPGA architectures, we named it VPR, short for Versatile Place and Route. VPR is publicly available from http://www.eecg.toronto.edu/~jayar/software.html and is capable of mapping to a wider variety of FPGA architectures than are discussed in this paper.

### 3.1 Global Routing Resource-Aware Placement

We employ the simulated annealing algorithm [13] for placement. The annealing schedule is based on feedback control of the accepted move rate, which was found to be crucial to obtaining excellent placements in [14, 15]. The key to a routing-resource-aware placement tool is ensuring that the cost function correctly models the relative difficulty of routing connections in regions with different channel widths. After considerable experimentation with many alternatives [19], we have developed a *linear congestion* cost function which provides the best results in reasonable computation time. Its functional form is

$$Cost_{linear} = \sum_{n=1}^{M} q(n)\left[\frac{bb_x(n)}{C_{av,x}(n)^{\alpha}} + \frac{bb_y(n)}{C_{av,y}(n)^{\alpha}}\right]$$

where the summation is over the M nets in the circuit. For each net, $bb_x$ and $bb_y$ denote the horizontal and vertical spans of its bounding box, respectively. The q(n) factor compensates for the fact that the bounding box wire length model underestimates the wiring necessary to connect nets with more than three terminals, as suggested in [16]. Its value depends on the number of terminals of net n; q is 1 for nets with 3 or fewer terminals, and slowly increases to 2.79 for nets with 50 terminals. $C_{av,x}(n)$ and $C_{av,y}(n)$ are the average channel capacities (in tracks) in the x and y directions, respectively, over the bounding box of net n.

This cost function penalizes placements which require more routing in areas of the FPGA that have narrower channels. The exponent, $\alpha$, in the cost function allows the relative cost of using narrow and wide channels to be adjusted. When $\alpha$ is zero the linear congestion cost function reverts to the standard bounding box cost function. The larger the value of $\alpha$, the more wiring in narrow channels is penalized relative to wiring in wider channels; we have experimentally found that setting $\alpha$ to 1 results in the highest quality placements.

Since $C_{av}$ depends only on the channel capacities, which do not change during a placement, and on the maximum and minimum coordinates of the bounding box, we precompute all possible $C_{av,x}$ and $C_{av,y}$ values. Consequently, recomputing this cost function is essentially as fast as recomputing the traditional bounding box cost function.

In an FPGA where all channels have the same capacity, $C_{av}$ is also a constant and hence the linear congestion cost function reduces to a bounding box cost function. In non-uniform and directionally-biased FPGAs, however, this cost function results in higher quality placements than a bounding box cost function. The exact amount of routability improvement depends on the precise global routing architecture used; as one would expect, those in which there is a large difference between the widths of channels in different regions show the largest improvement. For the architectures studied in this paper, placements produced with the linear congestion cost function typically require 5 to 10% fewer tracks to route than placements produced with a bounding box cost function.

We also implemented the cost function of [16], which we call a *non-linear congestion* cost function. This cost function divides the FPGA into an array of N x N regions and attempts to model the routing resource demand and supply in each of these regions. When a placement causes the routing resource demand to exceed the supply in some regions, the placement is heavily penalized. We found that this non-linear congestion cost function, when computed on a 4 x 4 grid (16 regions), generally produces placements which require 2 to 4% fewer tracks to route than those produced by the linear congestion cost func-

tion. However, keeping track of the routing resource demand in the various chip regions is computationally expensive, and placement with this cost function requires five times greater CPU time than the linear congestion function. Dividing the FPGA into smaller subregions to make localized congestion more visible did not work well; a non-linear congestion cost function computed on a 16 x 16 grid (256 regions) performs only marginally better than a cost function computed on a 4 x 4 grid, yet consumes sixteen times the CPU time.

We considered the reductions in track count achieved by the non-linear congestion cost function too small to warrant the additional CPU time, so the results presented in this study all use the linear congestion cost function. Nonetheless, we did rerun a few of our experiments with the non-linear congestion cost function and found that its use did not change any of the architectural conclusions presented below.

### 3.2 Congestion-Driven Global Routing

It is crucial for the global router to leverage the differences in the capacities of the various FPGA channels. The global router developed for this study employs a variant of the PathFinder negotiated congestion algorithm [17]. This algorithm consists of routing each net with a maze router [18], then ripping up and rerouting each net in sequence several times. In each of these subsequent routing iterations, the cost of using a node (which is *either* a channel segment or a logic block input pin) is modified, based on the competition for that node in both the current iteration and all previous iterations. A channel segment is the length of channel that spans one logic block; in an FPGA composed of an N x N array of logic blocks each channel contains N segments. We define the cost of a routing node somewhat differently than [17]; the cost of using routing node *n* is

$$c_n = (1 + h_n \cdot h_{fac}) \times (1 + p_n \cdot p_{fac}) + b_{n,n-1}.$$

The $p_n$ term is a measure of the present congestion at this node. It is updated *every time any net* is ripped-up and rerouted. The value of $p_n$ is equal to the overuse of this node that would occur if one more route were to use it, since the decision we are making during routing is whether another net should go through this node or not. For example, consider a channel with a capacity of six tracks and a segment of this channel in which all six tracks are currently used. The $p_n$ value of this channel segment is one, since routing one more net through this channel will result in an overuse of one.

The $h_n$ term accounts for the historical, or past, congestion at this node. It is updated *only after an entire routing iteration* is completed; i.e. after every net in the circuit has been ripped up and rerouted. Initially $h_n$ is 0; at the end of each routing iteration $h_n$ is increased by the amount by which demand for this node outstrips its capacity.

The $b_{n,n-1}$ term penalizes bends, since global routes with many bends in them present a more difficult

detailed routing problem in FPGAs with segmented routing, and will generally lead to detailed routes that are both slower and require more tracks. The value of $b_{n,n-1}$ is one if making the connection from node n-1 to node n implies a bend (i.e. node n-1 is a horizontal channel segment and node n is a vertical channel segment or vice versa), and is zero otherwise. Including this bend cost in the total cost of using a node produces routes with very few unnecessary bends and increases the global routing track count very little.

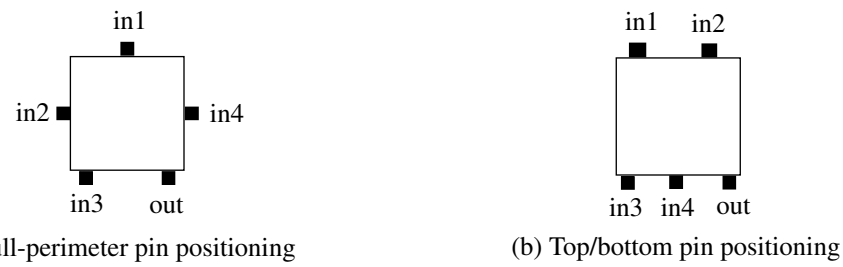
The key idea of Pathfinder is that the $p_{fac}$ term is 0 for the first routing iteration, and is gradually increased in successive iterations. Hence, each net is initially routed by the shortest path found. In successive iterations, the $p_{fac}$ term is gradually made larger so that congestion becomes more expensive and those nets which have alternate routes move out of the congested areas. The history term, $h_{fac}$, allows information from previous routing iterations to affect the current routing, further improving the router's ability to find and avoid congestion. By treating both channel segments and input pins as routing nodes, this algorithm makes use of the functional equivalence of LUT input pins in a very natural way. Initially, each connection uses the logic block input pin which leads to the shortest route. As the cost of congestion increases, nets are gradually forced to use unique input pins.

In our implementation each channel can have a different capacity. Since the cost of a channel segment is based on the amount by which routing demand exceeds its capacity, this router will automatically act to relieve pressure on narrow channels by rerouting nets through wider channels whenever necessary.

Considerable effort was spent tuning the *routing schedule* (the values of $p_{fac}$ and $h_{fac}$ over the course of the iterations). The best routing schedule we found set $p_{fac}$ to 0 for the first iteration, 0.5 for the second iteration, and 1.5 times the previous $p_{fac}$ value for all subsequent iterations. The value of $h_{fac}$ was set to 0.2 for all iterations; the fact that $h_n$ can only increase from iteration to iteration provides sufficient increase in the historical congestion penalty. This routing schedule increases the cost of congestion slowly enough that the net ordering is not very important -- nets with the most alternate routes move out of congested areas first. Increasing the cost of congestion more slowly than this reduced the number of tracks required only by 1 - 2% while increasing the CPU time by a factor of 2 to 3. Setting $h_n$ to 0 so that the router has no information about past congestion increased the number of tracks required by 15%.

# 4 Experimental Results for FPGAs with Directionally-Biased Routing Resources

The experimental framework and tools described above were employed to answer the questions posed in the introduction to this paper: first, is there an area-efficiency advantage to a directionally-biased architecture? Recall that in a directionally-biased FPGA the number of routing tracks in the horizontal and vertical directions are different. In essence, we are investigating if there is an exploitable directional bias in the basic nature of circuits. We characterize directionally-biased FPGAs by the ratio of the width of a hor-




(a) Full-perimeter pin positioning (b) Top/bottom pin positioning

**Figure 4:** Logic Block Pin Position Alternatives.

izontal channel to the width of a vertical channel, denoted as $R_h$. For example, Figure 1(a) depicts an FPGA with a 2:1 directional bias, i.e. $R_h = 2$.

We need to define an additional architectural feature which markedly affects our results: the positioning of the pins on the logic block. The two main cases of interest are illustrated in Figure 4. In Figure 4(a), the logic block input and output pins are distributed evenly around the entire perimeter of each logic block. We call this the *full-perimeter* pin positioning, and it is similar to the pin positioning used in Xilinx and ORCA FPGAs [2, 3]. Figure 4(b) illustrates the *top/bottom* pin positioning, which restricts the logic block input pin locations to lie only on the top and bottom of the logic block; it is similar to the pin positioning used in Actel FPGAs [4]. In all the results we show in this paper, each logic block pin appears (physically) on only one side of a logic block. We have found that for the channel connectivity values ($F_c$ [1]) found in today's commercial FPGAs this leads to the most area-efficient FPGAs [19].

We have also found that the ratio of the number of columns to the number of rows in an FPGA, which we call the aspect ratio, significantly affects area efficiency. Since most FPGAs have the same number of rows and columns, we first present the results for square (aspect ratio 1) FPGAs, before discussing the more general case of rectangular FPGAs in Section 4.2.

## 4.1 Results for Square FPGAs

Twenty-six large MCNC benchmarks were passed through the experimental flow of Figure 2 for values of $R_h$ ranging from 1 to 4. As discussed in Section 2, the result for each circuit is the number of track segments *per tile* needed to successfully global route the circuit in an FPGA[2] with the specified value of $R_h$. Figure 5 is a plot of area-efficiency versus the degree of routing direction bias, $R_h$, for both types of pin positioning. The vertical axis is the average number of tracks per tile required to successfully route the 26 benchmarks. Tables 1 and 2 summarize how well each of the seven largest circuits maps into FPGAs with differing amounts of directional bias; the results for individual circuits closely parallel the overall average shown in Figure 5.

2. Track segments are counted whether or not they are actually used, so this is a true representation of the area that must be devoted to routing in the layout.

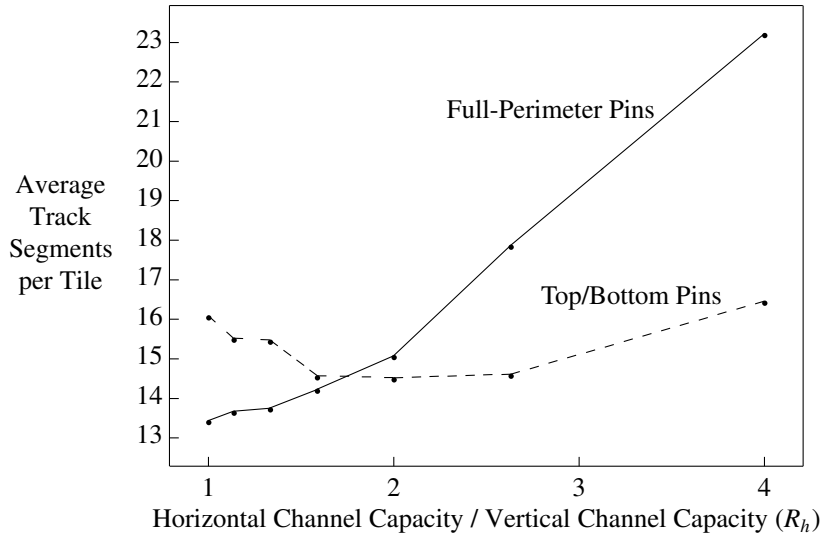


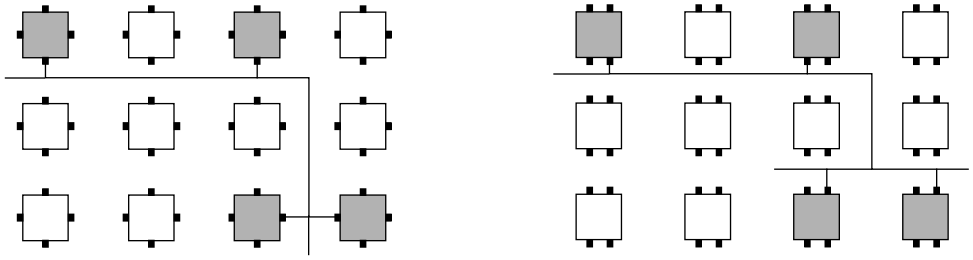

**Figure 5:** Area-Efficiency vs. Directional Bias for Square FPGAs.

**Table 1:** Routing Requirements of Largest Benchmarks when Mapped to FPGAs with Distributed Pins.

| Circuit | Number of Logic Blocks | Tracks per Tile Required for Global Routing | | | |
|---|---|---|---|---|---|
| | | $R_h = 1$ | $R_h = 1.33$ | $R_h = 2$ | $R_h = 4$ |
| alu4 | 1522 | 16 | 16 | 17 | 28 |
| apex2 | 1878 | 18 | 18 | 20 | 28 |
| apex4 | 1262 | 18 | 18 | 20 | 28 |
| diffeq | 1497 | 12 | 11 | 12 | 19 |
| ex5p | 1064 | 20 | 21 | 23 | 35 |
| misex3 | 1397 | 16 | 16 | 20 | 31 |
| seq | 1750 | 16 | 18 | 20 | 33 |

**Table 2:** Routing Requirements of Largest Benchmarks when Mapped to FPGAs with Top/Bottom Pins.

| Circuit | Number of Logic Blocks | Tracks per Tile Required for Global Routing | | | |
|---|---|---|---|---|---|
| | | $R_h = 1$ | $R_h = 1.33$ | $R_h = 2$ | $R_h = 4$ |
| alu4 | 1522 | 16 | 18 | 17 | 18 |
| apex2 | 1878 | 20 | 19 | 18 | 20 |
| apex4 | 1262 | 22 | 21 | 20 | 23 |
| diffeq | 1497 | 14 | 14 | 12 | 13 |
| ex5p | 1064 | 22 | 21 | 20 | 28 |
| misex3 | 1397 | 18 | 16 | 17 | 20 |
| seq | 1750 | 20 | 18 | 18 | 23 |



**(a)** Full-Perimeter Pins. **(b)** Top/Bottom Pins.

**Figure 6:** Example Multi-Terminal Net Routing.

Figure 5 shows that for the full-perimeter logic block pin positioning the best architecture has no directional bias. However, when the pins are restricted to the top and bottom of the logic block, the most efficient architecture has horizontal channels which are roughly twice as thick as the vertical channels. Another important conclusion is that the best full-perimeter architecture is about 8% more area-efficient than the best top/bottom pin architecture.

The full-perimeter architecture is more area-efficient because there is a greater chance that the block input pins are closer to their desired connections when they are in this configuration than when they are in the top/bottom configuration. For example, consider the two routings of a multi-terminal net shown in Figure 6. The top/bottom pin configuration needs six track segments to route this net, while the full-perimeter configuration requires only five. By making use of the functional equivalence of LUT input pins during routing, the router can often connect to a logic block pin adjoining a track segment it needs to use for other connections, essentially making the connection to this logic block for free. Since the top/bottom pin configuration has input pins bordering on only the horizontal channels, such "free" connections into logic blocks are less frequent, reducing area-efficiency.

The full-perimeter pins configuration achieves the highest area-efficiency when there is no directional bias to the routing because this makes the difficulty of routing to each of a logic block's nearest neighbors roughly equal. Consequently, the placement software can use all the nearby logic block locations equally to cluster the fanout of a net around its driver. Essentially, this allows one to cluster tightly coupled portions of logic in the smallest possible area. The top/bottom pins configuration, on the other hand, prefers a 2:1 directional bias because every connection to a logic block pin must come from a horizontal channel. This extra pressure on the horizontal routing resources is significant, since the average distance routed between pins is only about 3 track segments.

## 4.2 Rectangular FPGAs

In order to increase the IO-to-logic ratio, FPGA manufacturers may want to build rectangular FPGAs, as this increases the die perimeter and hence the number of pads. In this case the channels in one direction are longer and have more blocks connected to them than the orthogonal channel, so the best amount of

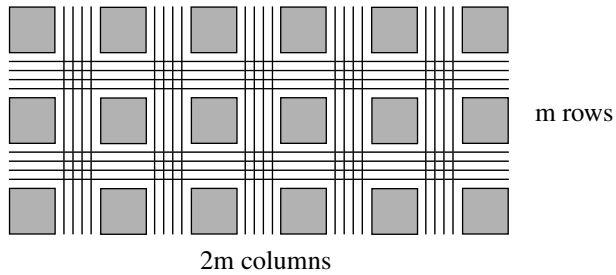



**Figure 7:** An FPGA with an Aspect Ratio of 2.

directional bias may change. We refer to the ratio of the number of columns in an FPGA to the number of rows as its aspect ratio. Figure 7 depicts an FPGA with an aspect ratio of two.

Figure 8 is a plot of the required tracks per tile versus $R_h$ for various chip aspect ratios for an FPGA with the full-perimeter logic block pin positioning. There are two features of interest in Figure 8. First, notice that the minimum of the aspect ratio = 1 curve is the lowest of the three, indicating that a square FPGA is most area-efficient. Secondly, the value of $R_h$ at which the minimum area occurs *increases* as the aspect ratio increases. As the aspect ratio increases, the horizontal channels become longer than the vertical channels and this results in greater demand for horizontal track segments. The best value of $R_h$ increases from 1 for a square FPGA to 1.33 and 1.59 for aspect ratios of 2 and 3, respectively.

The solid curve in Figure 9 shows how area-efficiency varies with aspect ratio when we set $R_h$ to the most appropriate value for each aspect ratio. The dotted curve in Figure 9 keeps $R_h$ fixed at 1, which is the best value for a square FPGA. The routing resource requirements increase moderately with aspect ratio; an FPGA with an aspect ratio of 3 requires 18% more tracks per tile than a square FPGA when $R_h$ is 1. When the most appropriate value of $R_h$ is used for each aspect ratio, however, an FPGA with an aspect ratio of 3 requires only 4% more track segments than a square FPGA. Thus we conclude that, as long as the horizon-

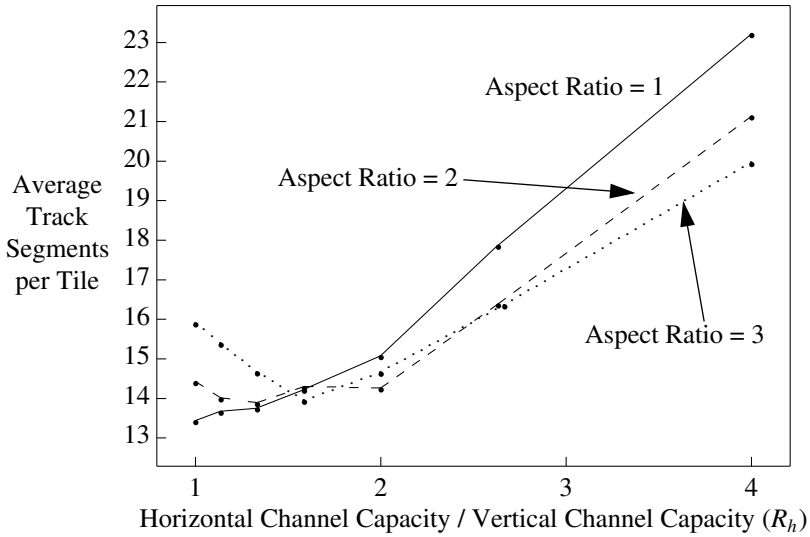



**Figure 8:** Area-Efficiency of Rectangular FPGAs with Full-Perimeter Pins.

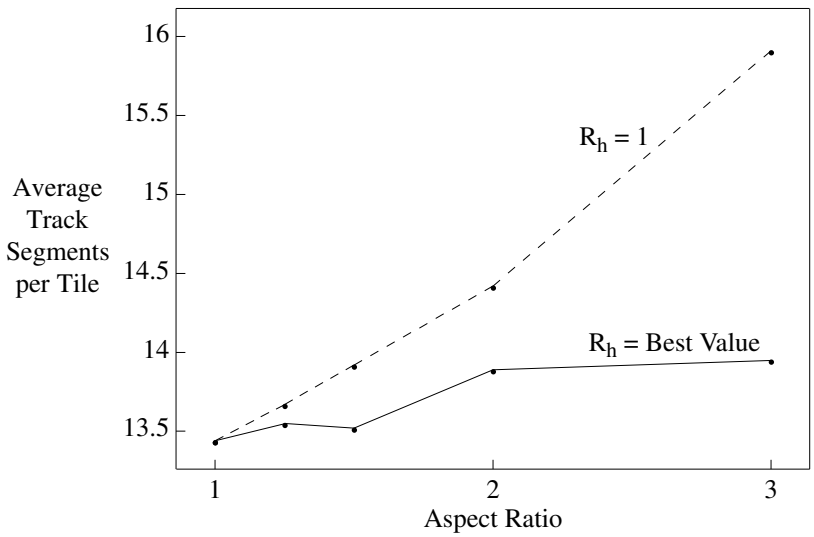



**Figure 9:** Area-Efficiency vs. Aspect Ratio for FPGAs with Full-Perimeter Pins.

tal and vertical channel widths are appropriately balanced, chip aspect ratios, and hence I/O counts, can be increased with little impact on the core area.

The variation of core routing area with aspect ratio is similar for FPGAs that use the top/bottom logic block pin positioning [19]. In this case an FPGA with an aspect ratio of 3 requires only 5% more tracks per tile than a square FPGA. For FPGAs of this type, however, the increase in the best value of $R_h$ as aspect ratio increases is less dramatic. The best square FPGA with top/bottom pins has horizontal channels which are twice as wide as vertical channels; the thicker horizontal channels are better able to cope with the increased pressure for horizontal tracks as aspect ratio increases.

# 5 Experimental Results for FPGAs With Non-Uniform Routing

The second key issue we explore concerns the area-efficiency obtained when the channels in different regions of an FPGA have different capacities. We only investigate FPGAs which use the full-perimeter pin positioning, as Section 4 showed that this pin positioning is best.

We define a non-uniform routing architecture to be one in which the number of tracks per channel changes from channel to channel across an FPGA. For example, Figure 1(b) illustrates a non-uniform FPGA in which the channels near the chip center are wider than those near the periphery. If congested regions of a circuit can be localized and placed in the portions of the FPGA with the widest channels, a non-uniform FPGA could have better area efficiency than a uniform FPGA. We will investigate three types of non-uniform FPGAs in which we vary the center/edge channel capacity ratio, the capacity of only the center channel, and the I/O channel capacity, respectively.

## 5.1 Center/Edge Capacity Ratio

There is a widespread belief that most congestion occurs in the center of FPGAs, and hence having wider channels near the FPGA center and narrower channels near the edges is expected to improve area-efficiency. To keep the layout problem tractable, we restrict ourselves to FPGAs which use channels of only two different widths. We can describe global routing architectures of this form with two parameters. Let $R_w$ be the ratio of the widths of the channels near the center of the FPGA to the widths of the channels near the FPGA edges, i.e. $W_{center}$ / $W_{edge}$. Let $R_c$ be the ratio of the number of channels with width $W_{center}$ to the total number of channels. For example, the FPGA of Figure 1(b) has $R_w = 2$ and $R_c = 0.5$.

Using the flow of Section 2, we again implemented 26 benchmark circuits in several architectures to determine their area-efficiency. We examined FPGAs with $R_w$ equal to 0.75, 1.18, 1.33, and 2, and with $R_c$ values varying from 0 to 1. The relative density of FPGAs with $R_w = 1.33$ and $R_w = 2$ is summarized in Figure 10, which plots the average number of tracks per tile required by the 26 benchmarks in each architecture. Note that the points at which $R_c$ equals 0 or 1 correspond to a uniform FPGA. The area required by each of the seven largest benchmarks in several representative architectures is listed in Table 3; again the results for individual circuits generally match the overall average, although there is some circuit-dependent behaviour. Note that the number of tracks per tile required for routing individual circuits in a nonuniform architecture is generally not an integer; since some portions of the FPGA have wider channels than others, the number of tracks per tile is an average over the various tiles such an FPGA must contain.

The results generally show that the less uniform the channel widths, the worse the FPGA area-efficiency. The worst area-efficiency with $R_w = 2$ occurs when $R_c$ is 0.5, meaning that half the FPGA channels are twice as wide as the other half. In fact, only two non-uniform FPGAs show even marginal area-efficiency improvements over the uniform case and both these FPGAs are very close to a uniform architecture.

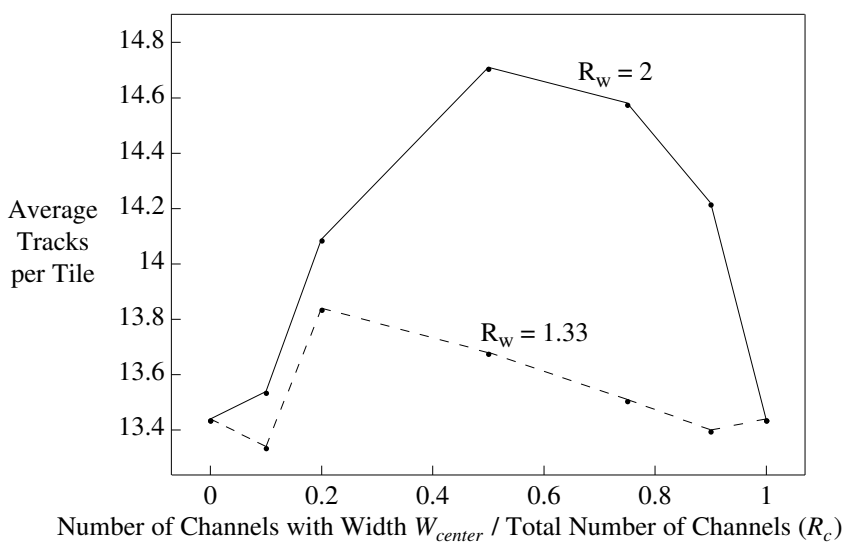



**Figure 10:** Area-Efficiency vs. Routing Architecture.

**Table 3:** Tracks/Tile Required for Global Routing by the Largest Benchmarks.

| Circuit | Uniform ($R_w = 1$) | $R_w = 2$ | | | $R_w = 1.33$ | | |
|---|---|---|---|---|---|---|---|
| | | $R_c = 0.2$ | $R_c = 0.5$ | $R_c = 0.75$ | $R_c = 0.2$ | $R_c = 0.5$ | $R_c = 0.75$ |
| alu4 | 16 | 16.0 | 16.6 | 15.6 | 14.7 | 15.9 | 16.8 |
| apex2 | 18 | 19.2 | 23.5 | 18.9 | 16.8 | 17.9 | 18.8 |
| apex4 | 18 | 18.6 | 22.4 | 20.1 | 17.1 | 17.8 | 18.7 |
| diffeq | 12 | 14.4 | 13.6 | 15.6 | 12.8 | 13.8 | 11.4 |
| ex5p | 20 | 20.8 | 19.6 | 20.5 | 21.1 | 20.8 | 20.2 |
| misex3 | 16 | 16.5 | 16.9 | 16.9 | 16.7 | 17.9 | 14.8 |
| seq | 16 | 18.6 | 19.9 | 17.2 | 16.7 | 18.0 | 16.9 |

In one, the 10% of channels nearest the center are 33% wider than the other channels, while in the other the 90% of channels closest to the center are 33% wider than the channels nearest the edges. The reduction in tracks per tile over a uniform FPGA is less than 1% for both of these FPGAs, so the improvement is not very significant, and is certainly not enough to justify the extra layout effort required in the physical design of such an FPGA.

These results are significant because there is a common belief among FPGA architects that there would be considerable benefit to these kinds of non-uniform architectures. The fundamental reason they do not show any benefit is that there is not much more congestion in the center of an FPGA than there is near its edges. In order to determine the "natural" routing demand distribution of circuits, we placed and routed the 26 benchmark circuits with all congestion avoidance features disabled, so that placement minimized wirelength and the router connected each net by the shortest path. Figure 11 plots the maximum and average number of tracks required by the horizontal channels as a function of the channel position within the

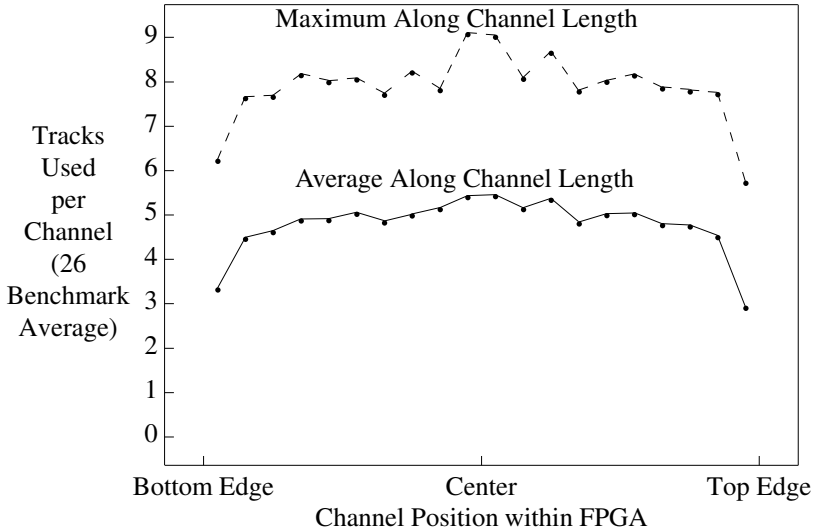



**Figure 11:** Average over Benchmarks of Track Demand vs. Position for Horizontal Channels.

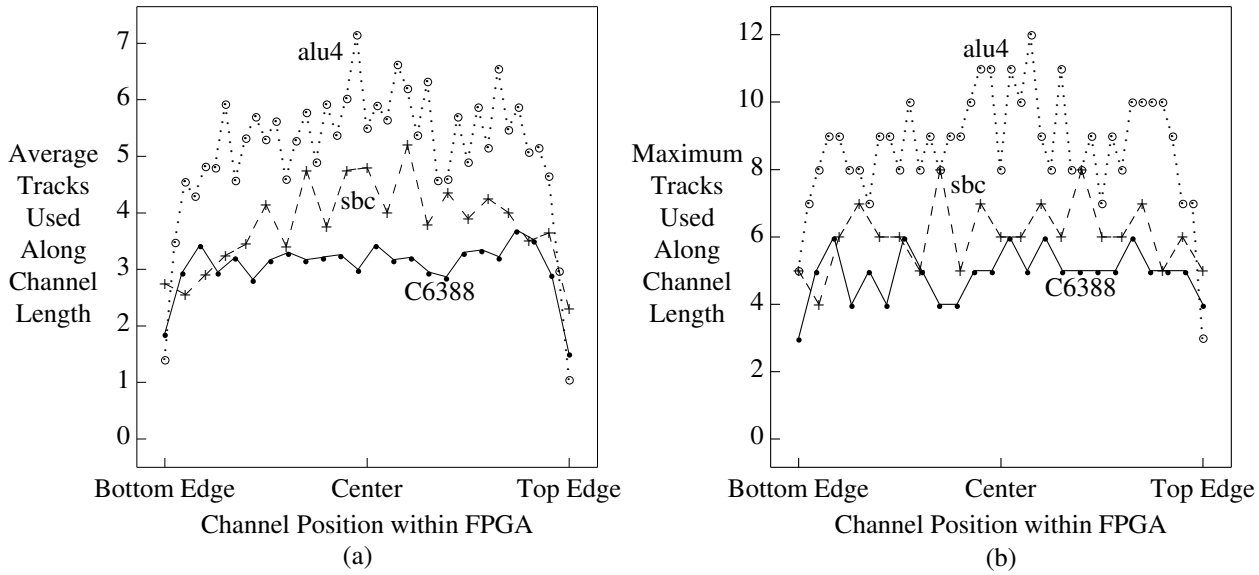



**Figure 12:** Average **(a)** and Maximum **(b)** Track Demand vs. Channel Position for Three Circuits.

FPGA, averaged over the 26 benchmark circuits. Demand for routing tracks is relatively constant over the middle 90% of the FPGA, and there is only a moderate decrease as one gets very close to the chip edges. Figure 12 plots the average and maximum tracks required by each horizontal channel versus the channel position for three representative benchmark circuits. There is some high-frequency variation from channel to channel, since the router is, in this case, not making any effort to route nets around congestion. Nevertheless, it is clear that these circuits closely mirror the behavior of the overall averages of Figure 11.

An additional reason for the poor area-efficiency of FPGAs with extra routing near their center is that typical circuits contain numerous local congestion "hotspots" (small regions where all the channels are full) and some of these hotspots occur quite close to the FPGA edge. Consequently, in order for an FPGA with thicker channels near its center to use fewer routing resources, the placement software must move all of these hotspots into the FPGA center. As discussed in Section 3.1, we spent considerable time investigating placement cost functions that modelled congestion well. The more advanced, and computationally expensive, cost functions, however, improved the performance of the uniform FPGA more than they did the non-uniform FPGA. We believe it is therefore more effective for CAD tools to attempt to spread out congestion as much as possible, rather than to try to localize it to a designated portion of a chip.

## 5.2 Single Center Channel

In an effort to improve routability, Lucent Technologies (formerly AT&T), has introduced an FPGA in which the center channel in each direction is extra-wide [6]. We define $R_m$ to be the ratio of the width of these center channels to the width of the other channels. Figure 13 depicts an FPGA with this extra routing in its center.

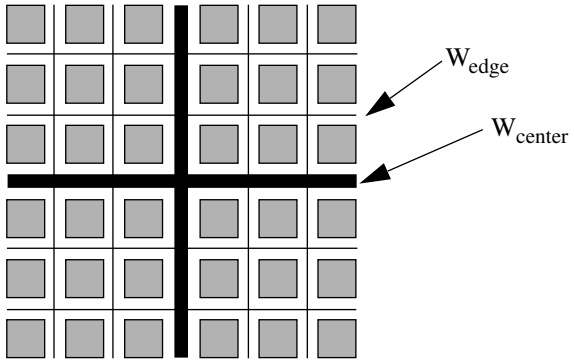



**Figure 13:** An FPGA with an Extra-Wide Center Channel.

The solid and dashed lines in Figure 14 show how area-efficiency varies with $R_m$ for this type of FPGA when a linear congestion cost function and when a bounding box cost function are used during placement, respectively. The data show that the most area-efficient FPGA is one without an extra wide channels in the middle -- i.e. $R_m = 1$. There is a sharp dip in the number of tracks/tile required at $R_m = 2$ when the linear congestion cost function is used, indicating an FPGA with area-efficiency almost as good as one with $R_m = 1$. This dip occurs at the first point at which the linear congestion cost function considers the cost of routing through narrow channels to connect two adjacent blocks to have the same cost as connecting two blocks separated by one intervening block through the extra wide channel. Consequently, the placer is able to make better use of the extra-wide channel at this point. The bounding box cost function leads to worse area-efficiency than the linear congestion cost function and the dip in required area near $R_m = 2$ does not occur. As with the non-uniform FPGAs of Figure 10 then, the best results are obtained by spreading extra routing resources over the entire FPGA rather than by adding them to only one region.

Aside from trying to create a more area-efficient FPGA, there is an alternate reason for making the

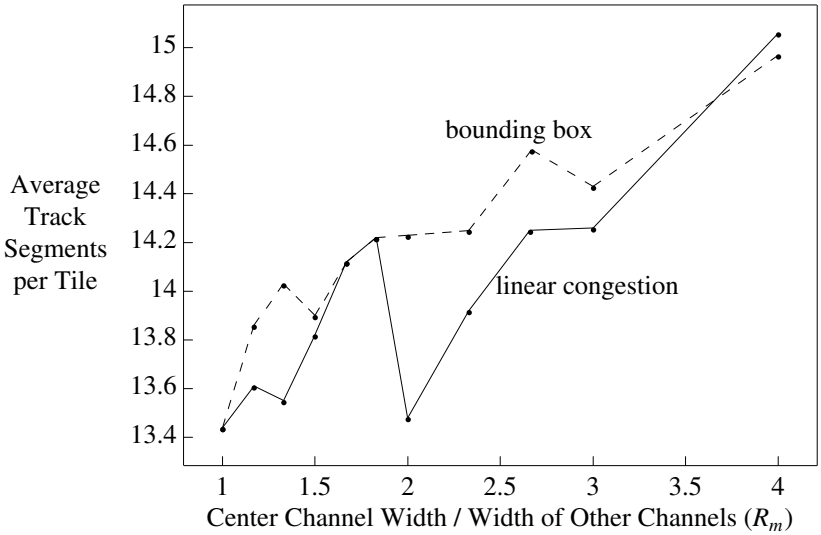



**Figure 14:** Effectiveness of an Extra-Wide Center Channel

center channel of an FPGA wider. FPGA manufacturers tend to create several FPGAs with the same basic architecture but with different numbers of logic blocks in order to appeal to customers with different capacity needs. As the number of logic blocks in a circuit increases, the demand for routing also increases [20], so at some point the channels should be widened. However, widening the channels for FPGAs with more logic blocks requires redoing the layout of the basic tile, which involves considerable time and expense. Since the center of an FPGA typically contains the programming logic, this area already requires its own custom layout, so extra routing tracks can be added to it with relatively little effort.

We found, however, that in most cases adding tracks to the center channel did not result in any significant reduction in routing pressure on the other channels. The only exceptions were FPGAs with $R_m$ near two. When circuits were mapped into such FPGAs by a placer using the linear congestion cost function the routing pressure on the other channels was reduced. It is relatively difficult to improve the routability of an FPGA simply by adding tracks to the center channel because the placement tool must now move all portions of the circuit that require more routing than the "normal" channels can provide close to the center channel so that they can make use of the extra routing available there.

### 5.3 I/O Channel

Another major FPGA vendor has added routing resources to the "I/O-channel" that runs between the I/O pads and the logic blocks, at least in part to ensure that fixed I/O pad placement does not impact routability and speed [8]. We define $R_{IO}$ to be the ratio of the width of this outermost channel to the width of the other channels. For example, Figure 15 depicts an FPGA with $R_{io}$ equal to 0.5. In these experiments all the channels running between logic blocks have the same width, $W_{Logic}$, so $R_{IO}$ completely describes the global routing architecture.

Figure 16 is a plot of the average track segments per tile required for the 26 benchmarks circuits versus $R_{io}$. The solid line in Figure 16 shows the trend when the I/O locations are chosen by the placement tool, while the dashed line is found when the I/O pads are "fixed" in a random location, to model the effect of

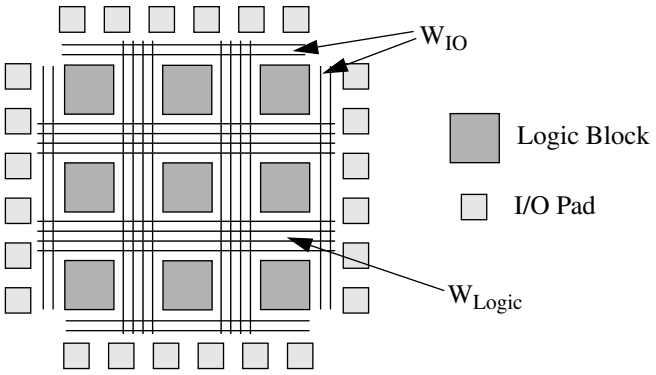



**Figure 15:** An FPGA with $R_{IO}$ = 0.5.

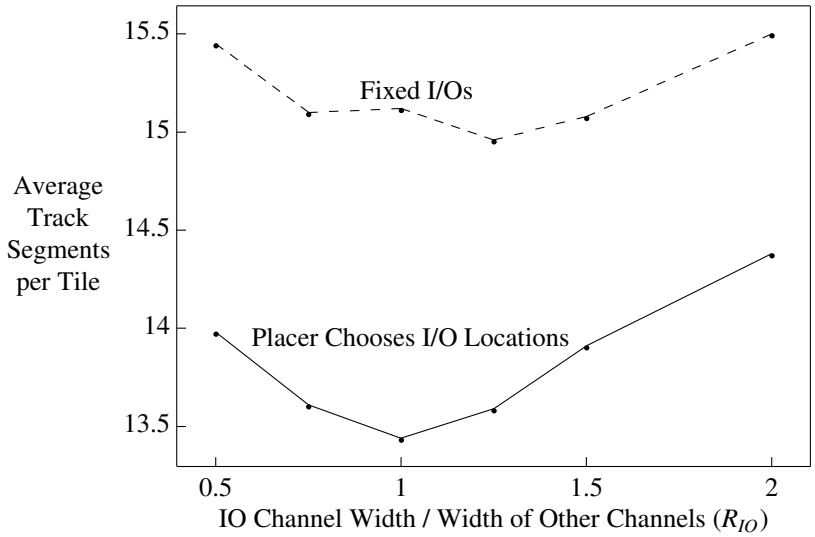



**Figure 16:** Routability vs. I/O Channel Width.

poor (from the FPGA's point of view) pin constraints.

There are several features of interest in Figure 16. First notice that fixing the I/O locations increases the number of routing tracks required by 12% on average. Architects must take this into account when designing FPGAs. Secondly, the curve where the I/O locations are chosen by the placement tool has its minimum value when $R_{IO} = 1$, again showing that it is best to spread routing resources evenly across the chip. Fixing the I/O pins shifts the minimum in the tracks per tile curve slightly so that it now occurs when $R_{IO} = 1.25$. While fixing the I/O pins leads to a significant increase in the number of routing tracks required, this increase is, for the most part, spread over the FPGA and not confined to the channels connecting to the I/O pads. Consequently, one should not make very wide channels adjoining the pads in order to improve routability with pin constraints, although a small increase in the I/O channel capacity is a net benefit.

In interpreting the best values of $R_{IO}$ we found, one should remember that in the FPGAs we consider the number of pins connecting to an I/O channel is roughly the same as the number of pins connecting to a logic channel. In architectures with different ratios of I/O pad to 4-LUT area, the best $R_{IO}$ values will change as the ratio of the number of pins bordering a logic channel to the number bordering an I/O channel changes.

In order to determine how the "natural" demand for tracks is altered when the I/O locations of a circuit are fixed in a poor configuration, we repeated the congestion-oblivious placement and routing experiments described in Section 5.1with the I/O locations fixed in a random configuration. Figure 17 plots the maximum and average number of tracks required by the horizontal channels as a function of the channel position within the FPGA, averaged over the 26 benchmark circuits. Comparing with the corresponding curve obtained with movable I/Os (Figure 11), one sees that the curves have shifted up by approximately half a

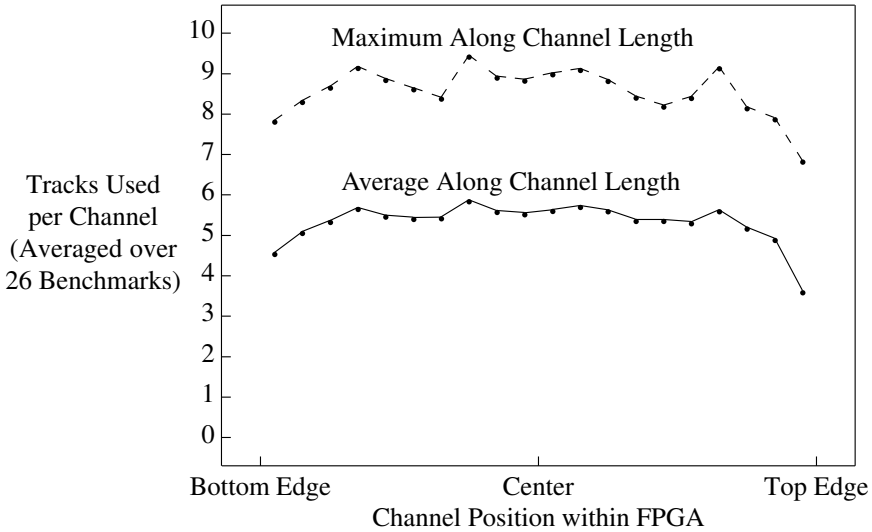



**Figure 17:** 26 Benchmark Average of Horizontal Track Demand when I/O Locations are Fixed.

track, and that the drop off in track demand near the chip edges is less pronounced. Figure 18 shows how the "natural" track demand of three typical circuits vary with channel position. By comparing with Figure 12, one sees that the curve for alu4 has changed little, while the sbc and C6388 curves have each shifted up by approximately a track and show significantly more demand for routing tracks near the chip edges than they did when the I/Os were movable. This is due to the different IO to logic ratios of these three circuits. Alu4 has very few I/Os; it uses only 7% of the I/O pads available in the FPGA to which it is mapped. C6388 and sbc, on the other hand, have considerably more I/O, and use 35% and 61% of the I/O locations available to them, respectively. As one would expect, then, fixing I/O locations has little effect on circuits with few I/Os. On the other hand, circuits with larger I/O requirements show an increase in routing track

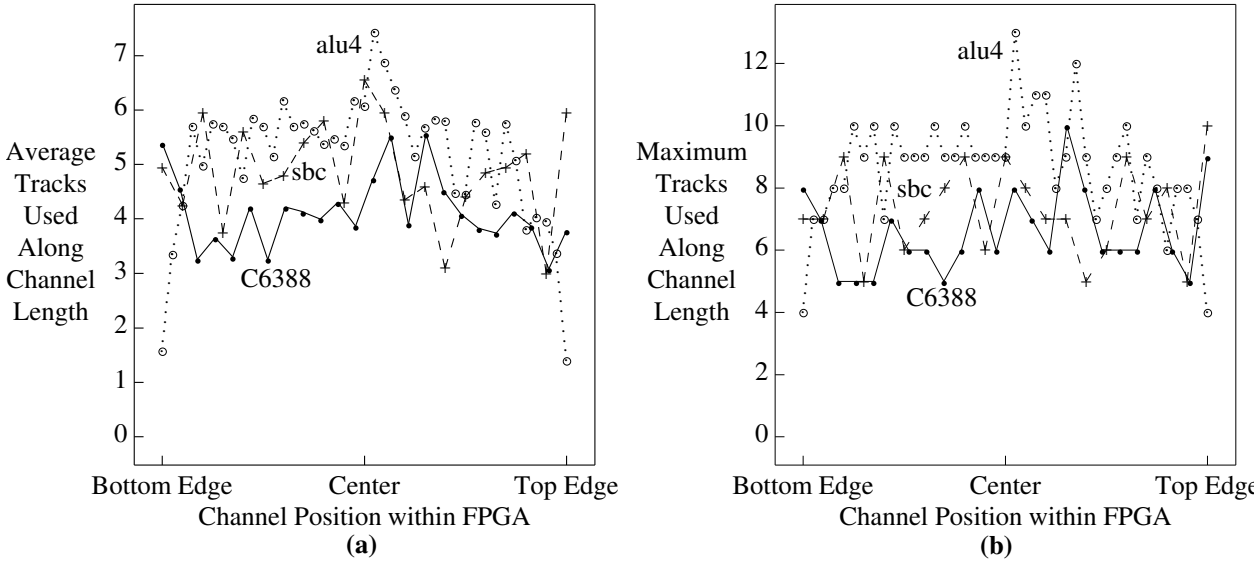



**Figure 18:** Average **(a)** and Maximum **(b)** Track Demand vs. Channel Position for Three Circuits when I/O Locations are Fixed.

demand across the entire FPGA, with the greatest increase near the chip edges.

## 6 Conclusions

The most interesting (and unexpected) conclusion of this work is that the most area-efficient global routing structure is one with completely uniform channel capacities across the entire chip and in both horizontal and vertical directions. The basic reason is that most circuits "naturally" tend to have routing demands which are evenly spread across an FPGA. The only (slight) exception we found to this "uniform is better" rule occurred when the I/O locations of circuits were fixed by board-level constraints. In this case making the I/O channel 25% wider than the other channels was a net benefit.

Of almost equal note, the area-efficiency is decreased only slightly by some non-uniform or directionally-biased architectures, provided the pin placement on the logic blocks is well-matched to the channel capacity distribution. Hence if such architectures are desirable for other reasons the impact on core area doesn't preclude their use.

More specifically, of the FPGA architectures studied, a full-perimeter pin position FPGA with no directional routing bias and uniform channel widths is most area-efficient. Employing a logic block with the top/bottom pin position requires approximately 8% more routing resources than full-perimeter FPGAs, and the most area-efficient top/bottom FPGA has twice as many horizontal routing tracks as vertical ones. We also found that one can construct rectangular FPGAs which are only slightly less dense than square FPGAs provided one adjusts the degree of directional bias in the routing resources to best match the chip aspect ratio.

Our experimental results in this paper were gathered with the linear congestion cost function in the placement tool because we felt the non-linear cost function was too slow to be commercially viable. However, it is interesting to note that while the non-linear function improved the routability of circuits for all FPGA architectures, it improved routability the most for uniform routing architectures. Apparently it is easier for advanced CAD tools to spread out congested regions than it is to localize them to designated portions of a chip that have extra routing resources. Consequently, we expect that future advances in CAD tools will tend to slightly increase the advantages of uniform routing architectures over their non-uniform counterparts.

## References


[1] S. D. Brown, R. J. Francis, J. Rose, and Z. G. Vranesic, *Field-Programmable Gate Arrays*, Kluwer Academic Publishers, 1992.

[2] Xilinx Inc., *The Programmable Logic Data Book*, 1994.

[3] AT & T Inc., *ORCA Datasheet*, 1994.

[4] Actel Inc., *FPGA Data Book and Design Guide*, 1994.

[5] Altera Inc., *Data Book*, 1993.

[6] B. K. Britton, et al., "Second Generation ORCA Architecture Utilizing 0.5µ Process Enhances the Speed and Usable Gate Capacity of FPGAs," *IEEE Int. ASIC Conf.*, Sept. 1994, pp. 474-478.

[7] M. Khalid and J. Rose, "The Effect of Fixed I/O Positioning on the Routability and Speed of FPGAs," *Proc. Canadian Workshop on Field-Programmable Devices*, 1995, pp. 94 - 102.

[8] D. Tavana, W. Yee, S. Young, and B. Fawcett, "Logic Block and Routing Considerations for a New SRAM-Based FPGA Architecture," *CICC*, 1995, pp. 24.6.1 - 24.6.4.

[9] V. Betz and J. Rose, "Directional Bias and Non-Uniformity in FPGA Global Routing Architectures," *to appear in ICCAD 1996*.

[10] S. Yang, "Logic Synthesis and Optimization Benchmarks, Version 3.0, *Tech. Report*, Microelectronics Centre of North Carolina, 1991.

[11] E. M. Sentovich et al, "SIS: A System for Sequential Circuit Analysis," *Tech. Report No. UCB/ERL M92/41*, University of California, Berkeley, 1992.

[12] J. Cong and Y. Ding, "FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs," *IEEE Trans. Computer-Aided Design*, Jan. 1994, pp. 1-12.

[13] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, May 13, 1983, pp. 671 - 680.

[14] J. Lam and J. Delosme, "Performance of a New Annealing Schedule," *ACM Design Automation Conference*, 1988, pp. 306 - 311.

[15] W. Swartz and C. Sechen, "New Algorithms for the Placement and Routing of Macros Cells," *ICCAD*, 1990, pp. 336 - 339.

[16] C. E. Cheng, "RISA: Accurate and Efficient Placement Routability Modeling," *ACM Design Automation Conf.*, 1994, pp. 690 - 695.

[17] C. Ebeling, L. McMurchie, S. A. Hauck and S. Burns, "Placement and Routing Tools for the Triptych FPGA," *IEEE Trans. on VLSI*, Dec. 1995, pp. 473 - 482.

[18] C. Y. Lee, "An Algorithm for Path Connections and its Applications," *IRE Trans. Electron. Comput.*, Vol. EC-10, 1961, pp. 346 - 365.

[19] V. Betz and J. Rose, "On Biased and Non-uniform Global Routing Architectures and CAD Tools for FPGAs," Technical Report, University of Toronto, 1996.

[20] M. Feuer, "Connectivity of Random Logic," *IEEE Trans. on Computers*, Jan. 1982, pp. 29 - 33.