ON TWO-STEP ROUTING FOR FPGAS

Guy G.F. Lemieux *Stephen D. Brown* *Daniel Vranesic*

Department of Electrical and Computer Engineering, University of Toronto, Canada
lemieux|brown|danv@eecg.utoronto.ca

ABSTRACT

We present results which show that a separate global and detailed routing strategy can be competitive with a combined routing process. Under restricted architectural assumptions, we compute a new lower bound for detailed routing and show that our detailed router typically requires no more than two extra routing tracks above this computed limit. Also, experimental results show that the Mapping Anomaly presented in [20], which suggests that separated routing may yield arbitrarily poor results in certain instances, is a concern only if nets are restricted to a single track domain. Finally, to motivate future work, we show the latest two-step routing results that we have achieved with the VPR global router and SEGA detailed router tools on the largest CBL benchmark circuits.

1. INTRODUCTION

Recent FPGA routing results have suggested that a separate global and detailed routing strategy is inferior to a combined routing process [1, 10, 19, 21]. Similarly, the practise of dividing multipoint nets into multiple two-point nets for routing was thought to negatively impact routability. In fact, recently published results have shown that combined routers have used significantly fewer routing tracks than the best-known two-step routers, CGE [4] and SEGA [11]. However, results obtained with a new global router, VPR, show that distinct global and detailed routing, combined with multipoint net division, can be competitive with the latest published FPGA routing tools. This is encouraging because separate global and detailed routing of two-point nets may have other practical benefits such as reduced memory use or compute time.

There is an additional concern that separate global and detailed routing may suffer from what [20] calls a Mapping Anomaly. This is a condition where the global route forms such a constraint that the channel density greatly under-specifies the minimum number of routing tracks required. After making the architectural assumptions suggested by [20], we sometimes detect the presence of a Mapping Anomaly. Our experimental results indicate that this anomaly is of critical concern if multipoint nets are constrained to a single track domain. However, the anomaly was not found to be present when nets were allowed to be split onto multiple track domains at input and output pins.

Finally, a new lower bound for evaluating the performance of any detailed router is presented. Although the new bound is not completely tight, the SEGA detailed router typically routes benchmarks within two tracks of the bound.

1.1. Paper Overview

The rest of this paper is organized as follows. Section 2 describes the FPGA architecture model used. Section 3 provides an overview of previous FPGA routing algorithms against which comparisons will be made. In Section 4, the Mapping Anomaly, confronting graph, and other graph-theoretic terminology are defined. In Section 5 the empirical methodology and tools used in this paper are presented, and the results and analysis follow in Section 6. The conclusions drawn from this data are summarized in Section 7.

2. FPGA ARCHITECTURE MODEL

The style of FPGA architecture assumed in this paper is similar to the Xilinx XC4000 series, but it is modeled with a set of parameters that represents a range of architectures. As illustrated in Figure 1, the architecture comprises a rectangular array of logic blocks with both horizontal and vertical *routing channels*, and I/O cells around the periphery. The contents of the logic blocks (L) are not of interest for this study. The routing channels comprise the wire segments and switches used to interconnect logic blocks. Wire segments are organized into both vertical and horizontal *tracks*; in the example in Figure 1 there are four tracks per channel and each logic block has two pins on each of its sides. We assume that all routing tracks consist of only short wire segments that span a single logic block. This assumption is made because we wish to compare results achieved by several recently-produced FPGA routing algorithms, and all of these algorithms' published results assume only short wire segments.

A key characteristic of the FPGA model is that the channels comprise two kinds of blocks, called *Switch (S)* and *Connection (C) blocks*, as illustrated in Figure 1. The S blocks hold routing switches that can connect one wire segment to another, and the C blocks house the switches that connect the wire segments to the logic block pins.

An S block is a rectangular switch box that connects wire segments in one segment of a channel to those in another. Depending on the topology, each wire segment on one side of an S block may be switchable to either all or some fraction of the wiring segments on each other side of the S block. The flexibility of the S block is given by the parameter F_s, which defines the number of other wire segments that a wire segment ending at an S block can connect to. An example S block appears in Figure 1a, in which each dashed line represents a programmable routing switch—in this figure, $F_s = 3$. In this study, the S-block topology is assumed to be disjoint. This means that the wiring tracks are isolated into disjoint domains by the switch organization. Consequently, if all S-block switches are turned on, a number of unconnected wiring groups are created, called *track domains*. For example, with the S block in

Figure 1. General Model of an Array-based FPGA.

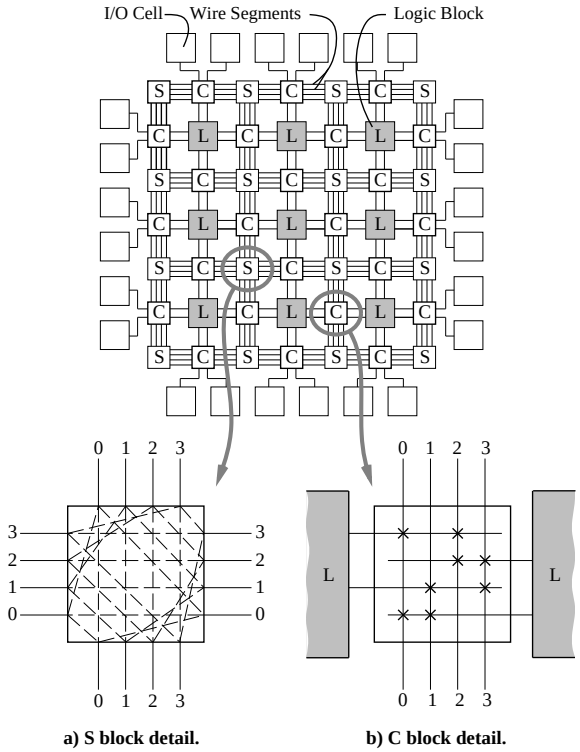


Figure 1a, a signal beginning on track 0 is restricted to wire segments in track 0, no matter which S-block switches it goes through.

Figure 1b illustrates a C block. The tracks are hardwired to pass through it and can be connected to the logic block pins via a set of switches. The flexibility of a C block, F_c, is defined as the number of wire segments in the C block that each logic block pin can connect to. In the figure, a routing option is represented as an $\times$— for this example, each pin can be connected to two vertical tracks; hence $F_c = 2$.

An important architectural feature is how the C block is implemented. If each $\times$ is simply a pass transistor, then two or more switches on the pin may be turned on to permit a routing dogleg, where the pin and connected wires behave as one electrically equivalent wire. However, if the $\times$'s along an input (driver) pin are implemented as a (de)multiplexor, only one connection to the tracks can be made. In these cases, doglegs are not possible. Many previous routing studies have assumed that routing doglegs can be used at both input and driver pins. However, commercial FPGAs such as Xilinx XC4000 [22] and Lucent ORCA FPGAs [12] do not permit input pin doglegs. The study in this paper considers both cases: doglegs at only the driver pin, and doglegs at any pin.

The main advantage provided by the FPGA model described above is its generality, which supports a wide range of routing architectures by changing the number of tracks per channel and the contents of the C and S blocks. Earlier studies have examined the effects of the F_c and F_s parameters [16]. Based on those previous studies, we will use the values $F_s = 3$ and $F_c = W$, where W is the number of tracks per channel, for all of the experiments in this paper. Note that these same assumptions are also used in recent publications on routing algorithms [1, 7, 10, 19, 21], and so are generally accepted as being reasonable.

3. PREVIOUS WORK

This section describes previous work related to FPGA routing that is directly comparable to the study in this paper.

3.1. FPGA Logic Block and I/O Placement

Many FPGA routing studies have used the benchmark netlists originally generated for CGE/SEGA. The placement for these benchmarks was generated by ALTOR [14], a tool originally intended for standard cell placement. ALTOR used a recursive min-cut bipartitioning strategy. By repeatedly partitioning in horizontal and vertical directions, ALTOR creates a final placement.

Recent tools, namely FPR [2], SPLACE [19] and VPR [3], include placement algorithms that are targeted specifically for FPGA use. FPR uses a recursive-partitioning technique that is similar to ALTOR, but each step uses simulated annealing to divide the netlist into an $m \times n$ grid, for some small fixed m and n. Before each recursive step, FPR also performs some global routing. This simultaneous placement and global routing strategy is unique among the FPGA tools considered in this paper. In comparison, SPLACE and VPR use simulated-annealing placement algorithms. VPR provides more efficient treatment of high-fanout nets and can therefore consider more moves than SPLACE in a given amount of CPU time.

3.2. FPGA Global Routers

The global router LocusRoute [15] was originally intended for standard cell applications. It accepts a placement and a multipoint netlist as inputs and breaks the nets into two-point nets. Each two-point net is routed with two or fewer bends with the objective of minimizing channel density. A bend cost can be applied to further discourage bends [18]. The output is a coarse graph for each connection consisting of a series of adjacent channel segments to guide it through the FPGA array. The quality of the LocusRoute channel assignment is measured by the maximum channel density, D_{max}, which is the largest number of distinct signals occupying a single channel segment.

The global routing step of VPR [3] uses a maze router on multipoint nets in a manner similar to [13]. All nets are routed, ripped up, and rerouted several times. After every iteration, it accrues a history cost to channel segments with a density greater than the target density, D_{target}. Subsequent net routings tend to avoid congested channels unless no alternative exists. VPR finds the minimum possible D_{target} that successfully routes a circuit with $D_{max} \leq D_{target}$.

3.3. FPGA Detailed Routers

The Coarse Graph Expansion (CGE) algorithm [4] was specifically developed for FPGA routing research. It expands all two-point nets along their global route into a small number of distinct paths, carefully pruning the search space. Wire resources for the lowest cost path are committed until the circuit is routed. A rip-up strategy is employed if needed, in which less pruning of possible choices is done in hard-to-route areas.

The successor to CGE, SEGment Allocator (SEGA) [11], used a different cost function structure to make use of long wire segments. SEGA also made the assumption that a net could be fully expanded into all possible paths along the global route. Consequently, SEGA does not re-expand a net when its paths are exhausted. Instead, the cost function increases a net's priority as its choices diminish. This approach yielded good results, so CGE-style rip-up was deemed unnecessary to the algorithm.

Since SEGA's original publication date, a number of different cost functions have been explored to investigate routability and speed-performance [5]. The cost function used to produce the results for this paper, called *Area*, has been the most successful so far in using the fewest wiring tracks. The *Area* cost causes SEGA

to first identify the nets which have the fewest number of remaining paths. Among these nets, the path with the lowest *Demand* cost (akin to CGE's cost) is chosen.

3.4. Combined FPGA Global and Detailed Routers

The Greedy Bin Packing (GBP) algorithm [20] combines both global and detailed routing into one step. By making the assumptions that $F_s = 3$, $F_c = W$, and that a disjoint S-block topology is used, an FPGA is routed by treating every track domain as a bin and greedily filling that bin with nets until no more will fit. GBP then proceeds to the next track domain and repeats the process. In this way, GBP is similar to the Best Fit Decreasing bin-packing heuristic. Observations that GBP did not densely pack the last few track domains led to the Orthogonal Greedy Coupling (OGC) algorithm [21]. By switching from one greedy algorithm to another (which has a different optimization goal) after some track domains have been packed, the last few track domains were more densely packed and fewer routing tracks were used.

A series of one-step routing algorithms was presented in [1]. In these algorithms, multipoint nets are routed one net at a time. If a net fails to route, it is moved to the front of the net order and routing is restarted. The FPGA routing resources are represented by a graph which shrinks as nets are routed. The algorithms differ by the way they route a multipoint net through the remaining graph. Five different core algorithms were presented, three of which were further enhanced using iteration. Of these eight algorithms, four minimized wirelength by solving the Network Steiner Tree Problem and four minimized source to sink distance using shortest-paths algorithms. In this paper, we compare our results to those produced by IKMB, one of the iterated Steiner-tree algorithms.

The FPGA Placement and Routing (FPR) algorithm [2] uses the same net routing strategy described above. It also uses the IKMB algorithm to perform detailed routing. However, before each recursive partitioning step, FPR greedily selects a partial global route for each net based on rectilinear Steiner arborescences[1] and assigns nets to specific S blocks. This allows FPR to balance congestion across each cut and fix the signal entrance or exit points on each side before cutting each subpartition.

The TRACER-fpga algorithm [7] also performs combined routing of multipoint nets. It uses a maze router seeded from the source and all sinks to route each net. Initially, all nets are routed by allowing them to share wires. Next, a simulated evolution technique (similar to simulated annealing) chooses nets for rip-up and rerouting; nets sharing resources are more likely to be ripped up. During rerouting, a high cost is used to discourage future sharing. When no more sharing occurs, a solution has been found. The TRACER-fpga_PR algorithm [10] is similar, except that it avoids sharing during initial net routing. Also, it uses slacks to order nets during initial routing and for selection of nets during rip-up. By using slacks, it gives long nets priority for direct connections and allows short nets to route around congestion.

The SROUTE algorithm [19] sequentially maze-routes each multipoint net by searching out the next closest sink. If a path for a net cannot be found, it is moved to the front of the net order and routing is restarted. To reduce the maze-routing search space, it initially follows paths which advance toward the closest sink.

3.5. Summary

A number of routers have been presented which address the routing problem in slightly different ways. None of the algorithms above directly address the issue of speed-performance, but some try to reduce wasted wirelength or take more-direct paths. All of these algorithms emphasize routability, and all try to minimize wirelength. The most recent routers (VPR, SROUTE, FPR, TRACER, IKMB, GBP, and OGC) improve routability by relaxing the minimum wirelength condition.

[1] An arborescence is a construction which contains the shortest path from a distinguished vertex, or source, to all other vertices or sinks.

Figure 2. A sample global routing, G, for three nets (a, b, and c) and the corresponding confronting graph, H. Notice the three connected vertices in H imply that three routing tracks are required to route. If the multipoint net c is broken into two-point nets (c_0c_1 and c_1c_2) and dogleg routes are permitted, the confronting graph H'_{dl} results and can be routed with only two tracks. One possible solution is implicitly shown in G, where a dogleg occurs on pin c_1. The confronting graph H'_{ddl} shows that when doglegs are allowed at the driver pin only, three tracks are still required.

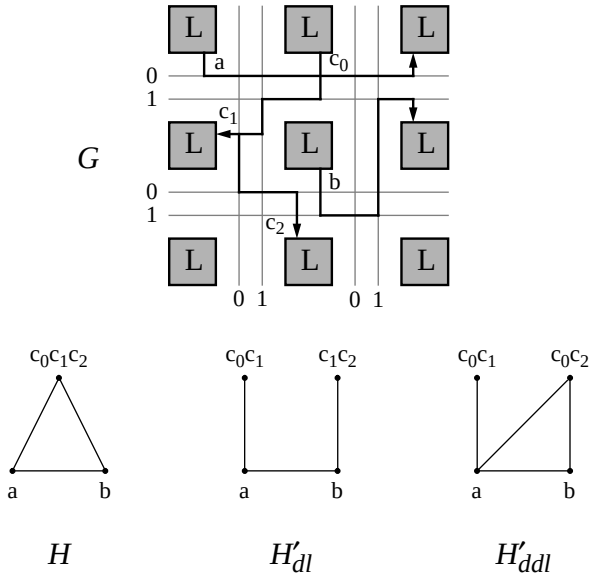


Table 1 gives a general overview of various architectural features and routing techniques that are used by the routers in this paper. Blank entries in the table mean 'not applicable'.

4. TERMINOLOGY

With the architectural assumptions that $F_s = 3$, $F_c = W$, and that S blocks are disjoint, the routing problem can be restated as a graph colouring problem. This observation led to the concepts of the Mapping Anomaly and a confronting graph in [20]. In this section, these terms and the underlying graph theory are defined.

A multipoint netlist which has been global routed can be represented by a forest of trees, $G(V, E)$, or simply G. Each net is a tree, where the driver and sinks form the leaves and intermediate vertices between the leaves are C or S blocks through which the net is routed. All vertices are labeled with the (x, y) co-ordinates of their location in the FPGA model. Additionally, leaf vertices are labeled with their corresponding logic block pin number.

In this paper, a two-point netlist $G'(V, E)$ is constructed from G in two different ways. In the first method, routing doglegs are permitted at input and driver pins of a net, forming $G'_{dl}(V, E)$. The second method permits routing doglegs at the driver pin only, and is called $G'_{ddl}(V, E)$. Each connected component of G'_{dl} or G'_{ddl} is a two-point net; in the latter graph, one endpoint is always a driver pin.

The disjoint S-block topology divides each routing track into a separate domain. This property allows the construction of the *confronting graph*, $H(V, E)$. Each vertex in H corresponds to a net in G. An edge is placed between two nets (vertices) in H if they travel through a common C block in G, *i.e.*, each net contains a vertex in G with the same C block (x, y) label. Thus, an edge in H rep-

Table 1. Comparison of architectural features (upper rows) and routing techniques (lower rows) used by FPGA routers.

	LocusRoute	CGE	SEGA	GBP	OGC	IKMB	TRACER	SROUTE	FPR	VPR
exploits pin equivalence	y			n	n	n	n	y	n	y
exploits output pin doglegs		y	y	y	y	y	y	y	y	
exploits input pin doglegs		y/n[a]	y/n[a]	y	y	y	y	n	y	
exploits long wire segments	y[b]	n	y	n	n	n	n	n	n	y[b]
performs rip-up and re-routing	y	y	n	n	n	y	y	n	n	y
greedy selection mechanism	y	y	y	y	y	n	n	y	y[c]	n
Steiner tree/arborescence based global routing	n			n	n	y	n	n	y	n
shortest-path global routing	y			n	n	n	n	n	y	n
maze global routing	n			n	n	n	y	y	n	y
guaranteed performance bounds	n	n	n	n	n	y[d]	n	n	y[d]	n
net-order dependent results	y	y	n	n	n	y	n	y	y	n

[a] CGE/SEGA will not dogleg at sinks if one end of the 2-pin input netlist always connects to the driver.
[b] A 'bend cost' can be applied to help better exploit long wire segments.
[c] A greedy selection is done to assign global routes at each partitioning step.
[d] Each net is guaranteed to use $\leq 2\times$ minimum number of wires (out of those remaining in the FPGA after previous nets are all routed).

resents an incompatibility between two nets to be assigned to the same track domain.

Similar confronting graphs can be built for $G'_{dl}(V, E)$ and $G'_{ddl}(V, E)$, denoted $H'_{dl}(V, E)$ and $H'_{ddl}(V, E)$, respectively. However, for these graphs an edge is never placed between two-point nets (vertices) which are part of the same multipoint net because they may be safely assigned to the same track domain. An example graph G, and the resulting confronting graphs H, H'_{dl} and H'_{ddl} are shown in Figure 2. Note that G is shown embedded in an array of logic blocks to illustrate the global route.

Using the confronting graph, the detailed routing problem is mapped to a graph (vertex) colouring problem. In this perspective, the vertices of H must be assigned a colour (track domain) such that no two adjacent vertices are assigned the same colour, and the minimum number of colours is to be used. The graph colouring problem is NP-complete on general graphs [9], so heuristics are commonly used to solve it.

This minimum number of colours required to colour H is called the *chromatic number*, denoted $\chi(H)$. It is important to note that $\chi(H)$ represents the minimum number of routing tracks required for detailed routing of G, and a routing solution with this many tracks is guaranteed to exist.

The *Mapping Anomaly* [20] is the observation that G may be constructed such that $\chi(H)$ can be arbitrarily higher than the maximum channel density, D_{max}. Since H is implicitly produced by the global router, the detailed router has no control over $\chi(H)$. Additionally, the global router attempts to minimize D_{max} and not $\chi(H)$ directly, so it may construct pathologically bad H configurations. This observation was used in [20] to support the notion that global and detailed routing should be combined.

The results presented in this paper suggest the Mapping Anomaly may not be a concern if routing doglegs are permitted, but it is a problem if doglegs are not allowed. This is intuitive because doglegs permit an 'escape hatch' for a signal to avoid interference, effectively reducing the net's length. Doglegs in the confronting graph have the effect of splitting a vertex in H and spreading the connectivity among the split vertices. The freedom to colour the split vertices similarly or differently, depending on the colour of adjacent vertices, often means that fewer colours are required.

It is desirable to compute $\chi(H)$ and use it to determine the quality of the detailed routing heuristic. However, we could not find an effective way to directly compute it. Instead, we compute a well-known lower bound: the *clique number* of H, or $\omega(H)$. The clique number of a graph is the size of the largest *clique*, or completely connected subgraph. Clearly, at least $\omega(H)$ different colours are needed to colour the largest clique because all of its vertices are adjacent to each other. Since all of the nets in a C block are completely connected (thus, forming a clique), the following useful relation is developed:

$$D_{max} \leq \omega(H) \leq \chi(H)$$

A similar relationship holds for the H'_{dl} and H'_{ddl} graphs.

The clique number is useful in two ways. First, it forms a tighter lower bound to gauge the quality of SEGA. Second, it helps show the presence of the Mapping Anomaly, as follows. If ω is much larger than D_{max}, then χ must be large, so the Mapping Anomaly is present. However, if ω is comparable to D_{max} then it may or may not be present. In this case, it is not present only if the graph can be coloured (routed) with a few colours (tracks) more than D_{max}.

5. METHODOLOGY AND TOOLS

The approach used in this study is empirical. That is, a set of benchmark circuits is input to a CAD tool chain and the routing results are analysed. The CAD tool chain consists of a new placement and global routing tool, VPR [3], and a detailed routing tool, SEGA [5, 11]. Two sets of benchmarks are used: older benchmarks provide a means of comparing to previously published results, and newer benchmarks allow more rigourous testing of the tools. The process and tools used are described in detail below. All of the tools, circuits, and results are available for download.[2]

5.1. Benchmark Preparation

Benchmark circuits from [11] were widely used to produce comparative results between routers. To use a new VPR placement or global routing for these circuits, they had to be converted to a format understood by VPR. A short program, `sega2blif`, was written to extract the multipoint nets from the SEGA input, and output the connectivity information in BLIF format. The same tool also wrote out a placement file which could optionally be used by VPR.[3]

The new benchmark circuits used in this study are from the CAD Benchmarking Laboratory (CBL) LGSynth93 suite [6]. A total of 198 circuits were converted to BLIF, optimized with SIS [17] and mapped into 4-input LUTs with FlowMap [8]. They were then run through the `blifmap` tool included with VPR to remove clock signals and, where possible, pack flip-flops into logic blocks. Clock

[2] `http://www.eecg.utoronto.ca/~lemieux/sega`

[3] Note that while SEGA permits I/O pins to be in the four corners of the periphery, VPR does not. Consequently, any corner I/O signals were moved as short a distance as possible to the next available I/O pad.

signals are removed because it is assumed that a global clock routing resource is available to route them.

5.2. Placement and Routing

All benchmarks were placed and global routed using VPR and detail routed using SEGA. The exact tool setup is described below.

The default VPR placement options were used for all benchmark circuits. However, for the older benchmarks, VPR was sometimes told to use the old placement information. Also, VPR required a parameter to describe the number of physical I/O pins that fit in the pitch of a logic block. This pitch was set to two for the new benchmarks, since this is comparable to current technology, and to an appropriate value for older ones. For the new benchmarks, VPR was allowed to choose the smallest square logic block array that fit the I/O padframe or logic block demand. However, the older benchmarks were consistently restricted to the original FPGA dimensions.

For global routing, VPR requires a logic block architecture specification. A logic block identical to the one used previously was specified: four functionally-equivalent input pins, one on each side, and two electrically-equivalent output pins on the right and bottom sides. VPR also allows a bend cost to be specified. We varied the bend cost between 0 and 10 on a subset of the new benchmarks and experimentally determined that a value of 1.25 gave the lowest total D_{max} and the lowest total number of tracks required by SEGA to route. Finally, VPR was restricted to route a net within 3 logic blocks of a bounding box formed by the sources and sinks.

Prior to detailed routing, the VPR multipoint net format had to be converted to a two-point net format for SEGA. To do this, a `vpr2sega` tool was written. This program can operate in one of two modes: *doglegs* and *driverdoglegs*. In *doglegs* mode, G'_{dl} is constructed as follows. A VPR net is read in and the distances between all pins along the global route are computed and used as edge weights in a complete graph spanning all the pins. A minimum spanning tree (MST) is then constructed, starting at the source, according to Prim's algorithm. Each edge in the MST is converted back to a two-point net that follows the global route and joins the pins. In *driverdoglegs* mode, a two-point netlist G'_{ddl} is constructed between the driver and every sink. Although this representation is not as concise as G'_{dl}, it implicitly instructs SEGA that a net can connect to multiple track domains only at the driver pin.

Once the netlist is converted, SEGA is used with the *Area* cost function to find the minimum number of tracks required to route.

5.3. SEGA Netlist Analysis

To analyse the SEGA netlist for D_{max} and construct the confronting graph, H, and its properties, a new tool, `chandens`, was written. It reports the maximum channel density and computes $\omega(H)$. Although computing $\omega(H)$ is known to be NP-hard in general [9], we have employed a branch-and-bound scheme with reasonable success. One of the most difficult benchmarks to evaluate with `chandens` in this fashion was `pdc`, requiring about 60 CPU hours on a 167MHz UltraSPARC. Most other benchmarks were evaluated in a matter of seconds to minutes.

Optionally, the `chandens` tool can also build the two-point net versions of the confronting graph, H'_{dl} and H'_{ddl}. Since these graphs are generally less connected than H, this has still proven to be computationally feasible for most benchmarks. However, due to memory limitations we were unable to completely evaluate some of the largest benchmarks. In these cases, the largest clique size found at the time of failure, indicated by a $\geq$ symbol, is used instead.

6. RESULTS AND ANALYSIS

6.1. Comparison to Previous Routers

The routing results for the older benchmark suite are shown in Table 2. When the old ALTOR placement is used, the VPR/SEGA combination routed all benchmarks with a total of 89 tracks, or 5 tracks fewer than IKMB. TRACER is the only router that produced better results, using only 85 tracks. If the placement is modified, the VPR/SEGA combination performed better than all others, using 9 fewer tracks than SPLACE/SROUTE[4] and 41 fewer than FPR. It is unexpected that a two-step router would perform as well as the combined routers. For these results, SEGA required one routing track more than the minimum predicted by the clique size and two more tracks than the minimum predicted by D_{max}, on average. In the worst case, SEGA required two tracks above the clique size.

6.2. Results with New Benchmarks

The 198 new benchmarks were all placed and routed. The results for the 20 largest benchmarks (ranging in size from 1046 to 8381 logic blocks each) are presented in Table 3. Note that some entries, denoted with a $\geq$ symbol, could not be exactly computed in a reasonable time because of excessive memory demands by SEGA and `chandens`. The VPR D_{max} column refers to the maximum channel density, the old lower bound for detailed routing. The $\omega(H'_{dl})$ column shows the new lower bound for detailed routing with doglegs, based on the clique number of the doglegs confronting graph. From Table 3, it is clear that $\omega(H'_{dl})$ is often larger than D_{max} and therefore provides a tighter bound for detailed routing. On average, the clique number tightens the bound by 1.1 tracks.

The SEGA G'_{dl} column shows the actual channel width required by SEGA to route the benchmark with doglegs. On average, SEGA requires two tracks above $\omega(H'_{dl})$ to route these large benchmarks, or 3.1 tracks over D_{max}. As a result, the Mapping Anomaly is not significantly present when doglegs are permitted.

The $\omega(H'_{ddl})$ and SEGA G'_{ddl} columns show the clique number and channel width required to route with driver doglegs. Although $\omega(H'_{ddl})$ could not be computed exactly for some circuits, it is relatively unchanged from $\omega(H'_{dl})$. Despite this, SEGA requires 55% more tracks than before to route these circuits. For some benchmarks such as `s298`, driver doglegs is a considerable restriction which requires 160% more routing tracks than before. However, other benchmarks such as `tseng` were relatively unaffected. Although the Mapping Anomaly is clearly not present in `tseng`, we cannot tell whether it is present in `s298`. The poor performance by SEGA may be caused by the Mapping Anomaly or by poor heuristic behaviour. To prove that the Mapping Anomaly is not at fault, we would need to find a valid colouring of `s298` with just over 6 colours.

Lastly, the $\omega(H)$ column in Table 3 shows the clique number of the confronting graph produced on the multipoint netlist. This column represents the lower bound for routing if no doglegs are permitted at all. Since these clique sizes are considerably larger than D_{max}, the Mapping Anomaly is present.

The data in Table 3 shows that the Mapping Anomaly has no effect if doglegs are permitted in the architecture. Although the driver dogleg restriction does not increase the clique size, it cannot be said for certain whether the Mapping Anomaly is present. However, it is strongly present if doglegs are not permitted at all. In this case, the global router should attempt to compensate for the 'confronting' nets. One way to do this is to perform combined routing, as previous routers have done, with the objective of minimizing the final channel width. Another way would be to use a better metric than channel density during global routing. For example, it may be reasonable to compute the clique number (or an estimate) of the confronting graph as global routing is done. The best way to approach this problem is a topic for future research.

[4] Note that among these routers, only SROUTE restricts doglegs to drivers. This restriction places it at a disadvantage in comparison to the others, yet it still performs well.

Table 2. Channel widths required to route older benchmarks. New results are in boldface.

Placement	ALTOR							SPLACE	FPR	ALTOR	**VPR**
Global R.	LocusRoute									**VPR**	
Detailed R.	D_{max}	CGE	SEGA	GBP	OGC	IKMB	TRACER	SROUTE		**SEGA**	
9symml	9	9	9	9	9	8	6	7	7	9	**7**	**6**
alu2	10	12	10	11	9	9	9	9	8	10	**8**	**7**
alu4	13	15	13	14	12	11	11	12	9	13	**10**	**8**
apex7	13	13	13	11	10	10	8	9	6	9	**10**	**5**
example2	17	18	17	13	12	11	10	11	7	13	**10**	**5**
k2	16	19	16	17	16	15	14	15	11	17	**14**	**10**
term1	9	10	9	10	9	8	7	8	5	8	**8**	**5**
too_large	11	13	11	12	11	10	9	11	8	11	**10**	**7**
vda	14	14	14	13	11	12	11	12	10	13	**12**	**9**
TOTAL	112	123	112	110	99	94	85	94	71	103	**89**	**62**

Table 3. Channel widths using VPR and SEGA for placement, global and detailed routing of the 20 largest benchmarks.

Circuit	VPR D_{max}	$\omega(H'_{dl})$	SEGA G'_{dl}	$\omega(H'_{ddl})$	SEGA G'_{ddl}	$\omega(H)$	Circuit	VPR D_{max}	$\omega(H'_{dl})$	SEGA G'_{dl}	$\omega(H'_{ddl})$	SEGA G'_{ddl}	$\omega(H)$
alu4	7	9	10	9	16	19	frisc	9	10	13	≥ 10	18	15
apex2	8	9	11	10	20	27	misex3	8	9	12	9	17	19
apex4	9	10	12	10	19	26	pdc	11	12	16	≥ 12	≥ 31	44
bigkey	6	7	8	7	9	9	s298	5	6	7	6	18	26
clma	9	≥ 10	14	≥ 10	≥ 24	30	s38417	6	≥ 7	8	≥ 7	10	11
des	6	7	9	7	11	11	s38584.1	7	≥ 8	9	≥ 8	12	11
diffeq	6	7	9	7	10	11	seq	8	10	12	10	18	24
dsip	5	6	7	6	9	9	spla	10	11	14	≥ 11	26	38
elliptic	8	9	11	≥ 10	16	20	tseng	6	6	8	7	9	9
ex1010	8	10	11	≥ 9	22	29	AVG.	7.6	≥ 8.7	10.7	≥ 8.8	≥ 16.6	20.4
ex5p	10	11	13	11	16	19	TOTAL	152	≥ 174	214	≥ 176	≥ 331	407

6.3. Graphical Results

In the graphs on the following page, we show the same routing results in a different fashion with all 198 benchmarks included.[5] The benchmarks are uniformly spread along the horizontal axis. The vertical axis shows the channel width, in discrete steps, of the routed circuits. All of the data could be presented in one graph, but we chose instead to separate them for clarity. Because of this, the vertical axis has different scales in the graphs. To further improve clarity, we sort the order of the benchmarks differently in each graph. This allows us to better illustrate trends in the data.

Figures 3 and 4 show how $\omega(H'_{dl})$ and $\omega(H'_{ddl})$, respectively, form a tighter bound than D_{max} for detailed routing. In Figure 3, the SEGA result with doglegs is shown to be very close to the lower bound given by $\omega(H'_{dl})$. In this case, SEGA typically requires only one routing track above the minimum to find a solution. The Mapping Anomaly is not present because SEGA found a valid routing which is close to D_{max}. In this graph, SEGA is exhibiting excellent behaviour.

The corresponding SEGA result for driver doglegs is shown in Figure 4. Although many circuits require less than three routing tracks above $\omega(H'_{ddl})$, a few require significantly more. In these cases, since $\omega(H'_{ddl})$ is close to D_{max} we are not certain whether this is a result of the Mapping Anomaly or poor heuristic behaviour.

In Figure 5 the clique sizes of H, H'_{dl}, and H'_{ddl} are compared. The graph indicates that the dogleg and driver dogleg clique sizes are very similar, but the no-doglegs clique size, $\omega(H)$, can grow very large. Since D_{max} is lower than the lowest line on this graph, the Mapping Anomaly must be strongly present in the circuits on the left of the graph. As a result, detailed routing without doglegs will use up many more tracks than what is predicted by D_{max}, even if a perfect algorithm is used. Although not shown, it is interesting to note that the channel width from routing G'_{ddl} with SEGA roughly follows (but usually remains below) $\omega(H)$, even though there is no direct relationship between them. We speculate that this may be caused by the presence of the Mapping Anomaly in H_{ddl} that does not take the form of a clique until some vertices are merged as in H.

[5] The few results which could not be properly computed are all included in Table 3 and are approximated by the values shown there.

7. CONCLUSIONS

The comparison to previous results has shown that a two-step global and detailed router can be competitive with the one-step approaches when routability is important. In fact, only TRACER was able to provide a lower track count than the VPR and SEGA combination. This result indicates that one should consider more than just routing in the minimum number of tracks when deciding whether a one or two-step router is appropriate. Some of the other issues include maintainability, design time, expected memory use and compute time, partitioning of software development effort, circuit delay, and result quality monitoring. This last point is interesting because the global router and detailed router can be separately optimized, and the progress of each can be recorded. Further, the clique number and chromatic number of the confronting graphs serve as improved lower bounds for detailed routing if certain architectural assumptions are made.

The experimental results show that it is important to consider the Mapping Anomaly in a global router if no doglegs are permitted, but it is not important to do so if they are. If only driver doglegs are allowed, we suggest that the Mapping Anomaly may be present, but we do not have proof. Since this case is important for existing FPGAs, more research is needed to confirm this.

One way for a global router to account for the Mapping Anomaly is to perform a detailed route internally, hence becoming a one-step router. However, another way is to estimate the chromatic num-

ber or clique number as the global route is performed. Minimizing this new number would be the new optimization goal for the global router. This is an open problem for future research.

Another way to interpret the routing data is that doglegs (at the driver and input pins) may be very useful architectural features to reduce the channel width required for routing. This raises another topic of future interest: is it area-efficient to fully support doglegs?

8. ACKNOWLEDGEMENTS

We wish to thank Vaughn Betz for his insightful comments and for making the necessary modifications to VPR. Mike Hutton did the painstaking effort of optimizing and mapping all of the benchmarks used in this study. Mike Hutton and Steve Wilton also provided valuable comments on an early version of this paper.

Figure 3. The number of tracks required by SEGA to route with full doglegs is slightly higher than the lower bounds formed by $\omega(H'_{dl})$ and D_{max}.

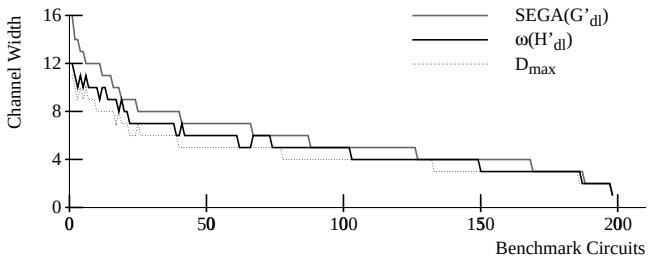


Figure 4. The number of tracks required by SEGA to route with only driver doglegs is more pronounced than the lower bounds implied by $\omega(H'_{ddl})$ and D_{max}.

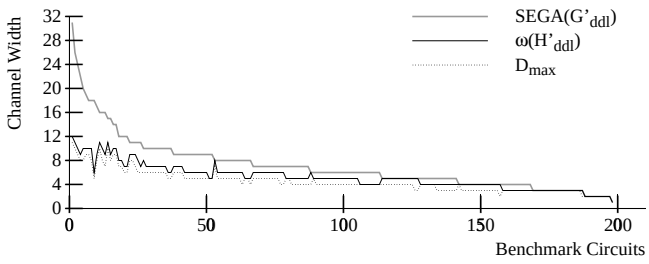
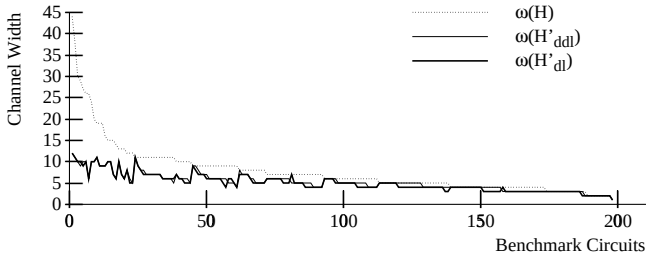


Figure 5. The lower bound required for routing without doglegs, $\omega(H)$, is significantly higher than the bounds with doglegs and driver doglegs, $\omega(H'_{dl})$ and $\omega(H'_{ddl})$, respectively. This large difference in bounds shows that the Mapping Anomaly is strongly present if doglegs are not permitted.



REFERENCES

[1] M.J. Alexander, G. Robins, "New Performance-Driven FPGA Routing Algorithms," *Design Automation Conference*, June 1995.

[2] M.J. Alexander, J.P. Cohoon, J.L. Ganley, G. Robins, "Performance-Oriented Placement and Routing for Field-Programmable Gate Arrays," *European Design Automation Conference*, September 1995.

[3] V. Betz, J. Rose, "Directional Bias and Non-Uniformity in FPGA Global Routing Architectures," *IEEE/ACM International Conference on Computer-Aided Design*, pp. 652–659, 1996.

[4] S. Brown, J. Rose, Z.G. Vranesic, "A Detailed Router for Field-Programmable Gate Arrays," *IEEE Transactions on Computer Aided Design*, 11(5), pp. 620–628, May 1992.

[5] S. Brown, G. Lemieux, M. Khellah, "Segmented Routing for Speed-Performance and Routability in Field-Programmable Gate Arrays," *Journal of VLSI Design*, 4(4), pp. 275–291, 1996.

[6] CAD Benchmarking Laboratory, North Carolina State University, LGSynth93 suite, `http://www.cbl.ncsu.edu/www/`

[7] C.-D. Chen, Y.-S. Lee, A.C.-H. Wu, Y.-L. Lin "A Performance and Routability Driven Router for FPGAs Considering Path Delays," *IEEE Transactions on Computer-Aided Design*, 14(3), pp. 371–374, March 1995.

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

[9] M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, New York, NY, 1979.

[10] Y.-S. Lee, A.C.-H. Wu, "A Performance and Routability Driven Router for FPGAs Considering Path Delays," *Design Automation Conference*, June 1995.

[11] G. Lemieux, S. Brown, "A Detailed Router for Allocating Wire Segments in FPGAs," *ACM/SIGDA Physical Design Workshop*, Lake Arrowhead, CA, pp. 215–226, April 1993.

[12] Lucent Technologies, *Field-Programmable Gate Arrays Data Book*, October 1996.

[13] L.E. McMurchie, C. Ebeling, "PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs," *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 111–117, February 1995.

[14] J.S. Rose, W.M. Snelgrove, Z.G. Vranesic, "ALTOR: An Automatic Standard Cell Layout Program," *Canadian Conference on Very Large Scale Integration*, pp. 169–173, November 1985.

[15] J.S. Rose, "Parallel Global Routing for Standard Cells," *IEEE Transactions on Computer-Aided Design*, 9(10), pp. 1085–1095, October 1990.

[16] J. Rose, S. Brown, "Flexibility of Interconnection Structures in Field-Programmable Gate Arrays," *IEEE Journal of Solid State Circuits*, 26(3), pp. 277–282, March 1991.

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

[18] B. Tseng, J. Rose, S. Brown, "Using Architectural and CAD Interactions to Improve FPGA Routing Architectures," *First International ACM/SIGDA Workshop on Field-Programmable Gate Arrays*, pp. 3–8, February 1992.

[19] S.J.E. Wilton, Architectures and Algorithms for Field-Programmable Gate Arrays with Embedded Memories, *Ph.D. Dissertation*, University of Toronto, 1997. An on-line version of this dissertation may be found at `http://www.ee.ubc.ca:80/home/staff/faculty/stevew/etc/www/`.

[20] Y.-L. Wu, M. Marek-Sadowska, "An Efficient Router for 2-D Field-Programmable Gate Arrays," *European Design Automation Conference*, pp. 412–416, Paris, 1994.

[21] Y.-L. Wu, M. Marek-Sadowska, "Orthogonal Greedy Coupling — A New Optimization Approach to 2-D FPGA Routing," *Design Automation Conference*, June 1995.

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