An Architecture-Driven Metric for Simultaneous Placement and Global Routing for FPGAs *

Yao-Wen Chang and Yu-Tsang Chang†

Department of Computer and Information Science, National Chiao Tung University, Hsinchu 300, Taiwan

Abstract

FPGA routing resources typically consist of segments of various lengths. Due to the segmented routing architectures, the traditional measure of wiring cost (wirelength, delay, congestion, etc) based on geometric distance and/or channel density is no longer accurate for FPGAs. Researchers have shown that the number of segments, instead of geometric (Manhattan) distance, traveled by a net is the most crucial factor in controlling the routing delay and cost in an FPGA. Further, the congestion information of a routing channel shall be measured by the available segments of specific lengths, instead of the density in a channel alone. In this paper, we propose an architecture-driven *metric* for simultaneous FPGA placement and global routing. The new metric considers the available segments and their lengths to optimize the wiring cost for placement and global routing. Experiments by employing a cluster growth placement and maze routing to *demonstrate* the new metric show respective average reductions of 8%, 20%, and 19% in the number of tracks used (area), maximum net delay, and average net delay based on the Lucent Technologies ORCA2C-like and the Xilinx XC4000EX-like architectures, compared with the traditional metric of geometric distance and channel density.

1 Introduction

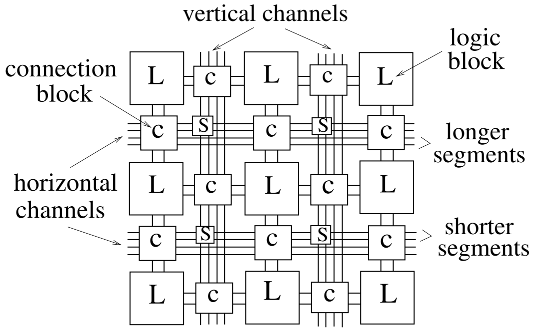

Figure 1: The array-based FPGA model.

Array-based architectures, illustrated in Figure 1, are used in many popular commercial *field-programmable gate arrays* (FPGAs) [14, 21]. An array-based FPGA is composed of a two-dimensional array of *logic blocks* interconnected by vertical and horizontal routing channels. The logic blocks, denoted by *L* in Figure 1, are customized to realize logic functions. The connection blocks, denoted by *C* in Figure 1, connect logic-block pins to routing channels. The routing channels comprise general routing resources used to make connections among pins. An intersection area of a horizontal and a vertical routing channels is referred to as a *switch block*, denoted by S in Figure 1. The switch blocks house programmable switches. FPGA routing uses programmable switches to make connections. The switches usually have high resistance and capacitance, and thus incur significant delays. To improve circuit performance and maintain reasonable routability simultaneously, the routing channels are usually segmented, and thus routing tracks consist of wires with a versatile set of lengths. For example, the routing channels in the Lucent Technologies ORCA2C and the Xilinx XC4000EX-series FPGAs contain *single-length lines*, *double-length lines*, *quad-length lines*, and/or *longlines* [14, 21]. (Figure 2 shows the corresponding three types of subchannels in a routing channel; note that quad-length lines are not shown in the figure.) Longer segments are intended for high fan-out, timing-critical signal nets. On the other hand, shorter segments are intended for short connections to avoid wasting routing resources. To optimize circuit performance without much sacrifice on routability, it is essential to consider the utilization of these various routing segments.

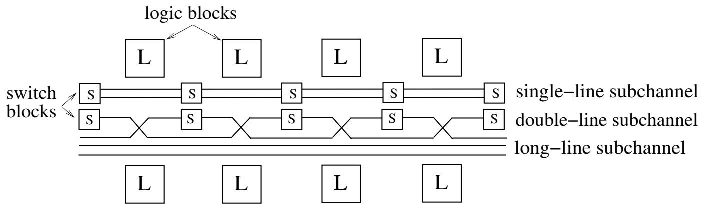

Figure 2: Three types of segments in a routing channel.

In this paper, we present an architecture-driven metric for simultaneous FPGA placement and global routing. The new metric is motivated by the intrinsic difference between FPGA routing architectures and those of the traditional ASICs. In traditional ASICs, geometric distance and/or channel density is an accurate metric for estimating wiring cost (wirelength, delay, congestion, etc) of a net; the metric is usually incorporated in popular wiring estimation schemes such as semi-perimeter, Steiner tree, minimum spanning tree, etc to guide circuit placement [18]. Due to the segmented architecture, the traditional measure is no longer suitable for FPGAs. Researchers have shown that the number of segments, instead of geometric distance, traveled by a signal is the most crucial factor in controlling routing delay and cost in an FPGA [10, 23]. (Obviously, however, the number of switches used by a signal is not necessarily proportional to the geometric distance of the signal.) Further, the congestion information of a routing channel shall be measured by the available segments in a specific subchan-

*This work was partially supported by the National Science Council of Taiwan under Grant No. NSC-89-2215-E-009-054. E-mail: {ywchang, ytchang}@cis.nctu.edu.tw.

†Yu-Tsang Chang is currently with Chip Implementation Center, National Science Council of Taiwan ROC, Hsinchu, Taiwan.

nel, instead of the density in a whole channel. Example 1 illustrates the importance of considering the segmented routing architecture for placement and global routing.

Example 1 *Figure 3 shows an FPGA layout, in which only related switch blocks are shown for clarity. Assume that node u is preplaced and only blocks B, C, and D are available for placing the other node v. In traditional ASIC placement, we would place v in the logic block B to obtain a solution with the minimum wirelength. Consider an FPGA with only double-length lines. Placing v in the logic block B (pin b) would consume four segments with segments r_1 and r_2 being wasted. (The wasted unit-length portions are denoted by dotted lines.) Note that an unused segment is not only a waste of routing resources, but also increases delay due to the larger capacitance incurred by the unused portion [3]. In contrast, placing v in the logic block D (pin d), which has a larger geometric distance, would also use four segments, but all segments are fully utilized. The reader can verify that placing v in the logic block C (pin c), which has the same geometric distance as pin d, requires five segments and wastes two segments of unit length. Obviously, placing v in the logic block D (pin d) gives the best solution although the corresponding geometric distance is not the smallest.*

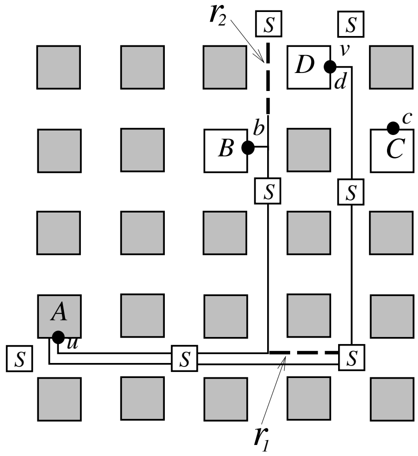

Figure 3: Placing node v in one of the three empty slots, B, C, and D, with the other node u being preplaced in A.

The placement and routing problems have been studied extensively in the literature, e.g., [1, 2, 4, 5, 6, 7, 8, 9, 12, 15, 16, 17, 19, 20, 22]. Existing FPGA placers and/or routers are based mainly on geometric distance and/or channel density as their cost (wirelength, delay, congestion) metric [1, 5, 7, 9, 12, 15, 19, 20]. Due to the segmented routing architecture, they cannot effectively utilize the FPGA resources with a versatile set of segments. To fully consider the segmented routing resources in FPGAs, we propose an architecture-driven metric for simultaneous placement and global routing. The architecture-driven metric simultaneously considers congestion, wiring, and delay costs; it defines congestion cost based on the available segments of each type and wiring and delay costs based on the number of segments used as well as the wastage of segments. To explore the effects of the architecture-driven metric on routing solutions, we incorporate the architecture-driven and traditional metrics into a simultaneous placement and global routing algorithm (based on a simple cluster growth placement and a maze routing approaches) to place and globally route the VPR/SEGA benchmarks [3, 4]. The global routing results generated from each metric are then fed into the SEGA detailed router (which can consider the segmented routing architecture) to obtain the final routing solutions. The results show that the architecture-driven metric achieves respective *average* reductions of 8%, 20%, and 19% in the number of tracks used (area), maximum net delay, and average net delay based on the Lucent Technologies ORCA2C-like and the Xilinx XC4000EX-like architectures in the detailed routing solutions, compared with the traditional metric of geometric distance and channel density.

The rest of this paper is organized as follows. Section 2 gives the problem specifications. Section 3 presents our architecture-driven metric. Section 4 gives the simultaneous placement and global routing algorithm. Experimental results are reported in Section 5, and finally the conclusion is given in Section 6.

2 Problem Specification

A net is said to be a *complete route* if the route of the net does not waste any segment length; otherwise, it is an *incomplete route*. For example in Figure 3, the route between A and D is complete while that between A and B (C) is incomplete. Intuitively, a complete route is superior to an incomplete route since it uses routing resources more effectively.

As mentioned earlier, the objective for simultaneous FPGA placement and global routing is no longer to find routes that travel the shortest geometric distance and balance the overall channel density, but to find those that consume the minimum total number of segments, utilize the segments of various lengths most effectively, preferably resulting in all complete routes, and make the subchannel density for each type of segments most balanced. *The Simultaneous FPGA Placement and Global Routing Problem* is defined as follows. Given a set of slots (logic-block pin locations) $Z = \{z_1, z_2, \ldots, z_l\}$ in an FPGA and a netlist $N = \{n_1, n_2, \ldots, n_m\}$, each $n_i, 1 \leq i \leq m$, with a set of pins (terminals) $T_{n_i} = \{t_1, t_2, \ldots, t_r\}$, assign each $t_j \in T_{n_i}$, $1 \leq j \leq r$ to a unique pin location $z_k, 1 \leq k \leq l$, and each net n_i to a sequence of segments in routing channels so that the routability and timing performance are optimized.

In this paper, we are most concerned with the problem of finding an effective metric to guide the simultaneous FPGA placement and global routing. By effective, we mean one that can simultaneously optimize the routability and timing performance for the problem being considered.

3 The Architecture-Driven Metric

To achieve good routability and timing performance, the metric must consider the three issues: (1) the number of segments used (delay), (2) the type and length of segments used (delay and routability), and (3) the density of a subchannel (routability). More specifically, the cost function is defined as follows. Let $p(u, v)$ be a routing path between pin u and pin v, and $\mathcal{T}$ be the set of the types of segments (e.g., $\mathcal{T}$=$\{S, D, L\}$ represents the set of single-length lines, double-length lines, and longlines). The cost for the route $p(u, v)$ by using the type-i segment $\phi_i : p(u, v) \rightarrow \Re^+ \cup \{0\}$, $i \in \mathcal{T}$, is given by

$$\phi_i(p) \quad = \quad \alpha s_i(p) + \beta w_i(p) + \gamma c_i(p), \tag{1}$$

where $s_i(p)$, $w_i(p)$, and $c_i(p)$ represent the respective cost functions for the number of segments used, the penalty (wastage) for an incomplete route, and channel density of the type-i subchannel, and α, β, and γ are user-specified parameters. Here, $\alpha + \beta + \gamma = 1$. Note that $s_i(p)$ and $w_i(p)$ are associated with delay while $w_i(p)$ and $c_i(p)$ with routability. Let $\delta_i(p)$ represent the delay cost. We have

$$\delta_i(p) \quad = \quad \alpha s_i(p) + \beta w_i(p). \tag{2}$$

Here, $s_i(p)$ is a delay function and is used to minimize the number of segments assigned to the route p because the number of switches (or segments) traveled by p is the most crucial factor in controlling the routing delay [10, 23]. $w_i(p)$ is a penalty function for an incomplete route and used to minimize the wastage due to assigning a long segment to a short connection. If p is an incomplete route, it would waste some portion of segment length. Note that $w_S(p)$ always equals zero since all routes are complete if only single-length segments are involved for routing. The values of $w_D(p)$ and $w_L(p)$ depend on the number of units of wasted length of the incomplete route p. $c_i(p)$ is a congestion function for controlling the density associated with the type-i subchannel. It will assign higher costs to route nets through congested areas to balance the net distribution among subchannels. As the example shown in Figure 3 which consists of only double-length segments, assuming $\alpha = \beta = 0.5$, we have $\phi_D(u,b) = 3.0, \phi_D(u,c) = 3.5$, and $\phi_D(u,d) = 2.0$. (For single-length segments, $\phi_S(u,b) = 3.0$, $\phi_S(u,c) = 4.0$, and $\phi_S(u,d) = 4.0$.)

Our architecture-driven metric, $\Phi : p(u,v) \rightarrow \Re^+ \cup \{0\}$, is then given by

$$\Phi(p) \quad = \quad \min_{i \in \mathcal{T}} \phi_i(p). \tag{3}$$

$\Phi(p)$ determines the type of segments and its corresponding routing path during placement and global routing to optimize delay and area performance. It computes the cost for routing on each type of segments and picks the one with the minimum cost for routing.

4 Simultaneous Placement and Global Routing

To *demonstrate* the architecture-driven metric, we need a vehicle to explore the effects of the metric on routing quality. Our solution is a simultaneous placement and global routing algorithm named *APR* which is based on a graph-search technique. Before we can apply the graph-search technique, we first need to model an FPGA as a graph such that the graph topology can represent the FPGA architecture. The following subsection gives an example of the graph modeling for the routing architecture with single-length lines, double-length lines, and long-lines. Similar modeling applies to the architectures with other types of segments.

4.1 Graph Modeling

For an FPGA with single-length, double-length, and longlines. Our solution is to model the FPGA as a "3-D 6-layer" graph. There is one layer for single-length lines, the *single-length layer*; four layers for double-length lines, *double-length layer* (since we need to model the switching among the two-dimensional array of switch blocks); one layer for longlines, the *longline layer*. Figure 4 illustrates the FPGA modeling. As shown in Figure 4, each logic block, connection block, or switch block is represented by a node, called a *logic node*, a *connection node*, and a *switch node*, respectively. Each connection between a connection block and a switch block, is modeled as an edge named a *channel edge*; each set of connections between a logic and a connection blocks is also modeled as an edge called a *connection edge*. There are *layer edges* between the corresponding switch node and connection node in different layers. Layer edges are used for switching an *incomplete* route between double-length segments. Paths in the graph represent routes on the FPGA, and vice versa. Weights associated with the edges represent area, delay, and congestion cost information. We shall denote the graph and model an FPGA F as $G_F(V,E)$. The edge set E is partitioned into the set of channel edges C_e, the set of connection edges O_e, and the set of layer edges L_e.

4.2 The APR Algorithm

The one-step APR algorithm combines a cluster growth based placer and a maze router [11]. Unlike the traditional metric which is based on geometric distance and channel density, APR is guided by the metric defined in Equation (3). APR places and routes net by net. For each net being placed currently, we prefer to place it in a position that will make its routing cost minimized.

The delay cost $\delta_i(p)$ (see Equation (2)) for a path p routed on single- and double-length lines can be computed by $\delta_S(p) = \alpha + \sum_{e \in p} d_S(e)$ and $\delta_D(p) = \frac{3}{2}\alpha + 4\beta + \sum_{e \in p} d_D(e)$, where

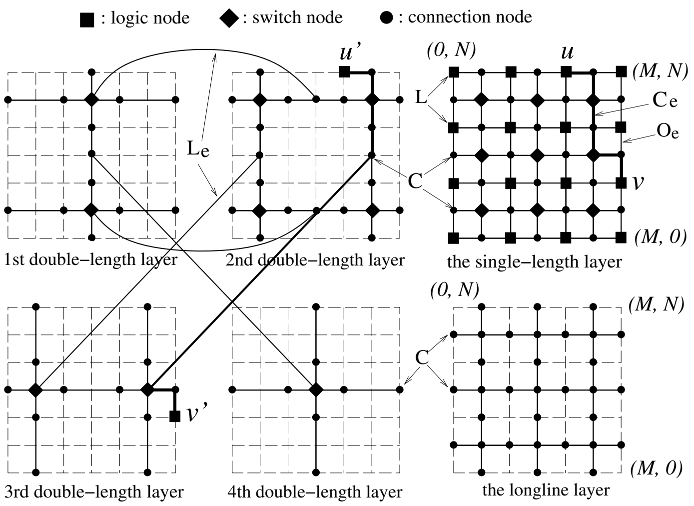

Figure 4: The graph modeling for an FPGA with 4×4 logic blocks. There are six layers for the three types of segments. (Note that the logic nodes and their corresponding connection edges are not shown on the four double-length [except those associated with u' and v'] and the longline layers for clarity. Also, the dashed lines are used for reference only, not part of the graph.)

$$d_S(e) = \begin{cases} \frac{\alpha}{2}, & \text{if } e \in C_e \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

and

$$d_D(e) = \begin{cases} \frac{\alpha}{4}, & \text{if } e \in C_e \\ \frac{\alpha}{2} + 2\beta, & \text{if } e \in L_e \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

As an example shown in Figure 4, $\delta_S(u,v) = 3\alpha$ because $p(u,v)$ passes four channel edges (C_e) and consumes three segments. $\delta_D(u',v') = 3\alpha + 6\beta$ because $p(u',v')$ passes four channel edges (C_e) and one layer edge (L_e) , and thus consumes three segments and wastes three unit-length portions of a double-length line.

For a path routed on longlines, it consumes either a "straight" longline or two longlines with a bend. Correspond the longline layer to the $M \times N$ plane in the first quadrant (see Figure 4). It suffices to consider the two or three points to define the delay cost for a path $p(u_i, u_f)$: *initial point* $u_i = (x_i, y_i)$, *bend point* $u_b = (x_b, y_b)$ if any, and *final point* $u_f = (x_f, y_f)$, where $0 \leq x_i, x_b, x_f \leq M$ and $0 \leq y_i, y_b, y_f \leq N$. We classify a route on the longline layer into the four categories and summarize its delay cost as follows:

$$\delta_L(p) = \begin{cases} \beta(M - |x_f - x_i|), & \text{case 1} \\ \beta(N - |y_f - y_i|), & \text{case 2} \\ \alpha + \beta(N - |y_f - y_b| + M - |x_i - x_b|), & \text{case 3} \\ \alpha + \beta(N - |y_i - y_b| + M - |x_f - x_b|), & \text{case 4} \end{cases} \tag{6}$$

where case 1 (2) is for a horizontal (vertical) straight path and case 3 (4) for a bend path. (Note that the cases 3 and 4 are different only in the definitions of initial and final points.)

The cost $c_i(p)$ (see Equation (1) associated with subchannel density is intended to evenly distribute the nets among subchannels so that the subchannel density is minimized after all nets are routed. We define the function $h_i : E \rightarrow \Re^+ \cup \{0\}$ to consider the cost for routing through jth subchannel of type-i segments as follows:

$$h_{i,j}(e, \vec{b_i}) = \begin{cases} b_{ij}, & e \in C_e \\ 0, & otherwise \end{cases}$$

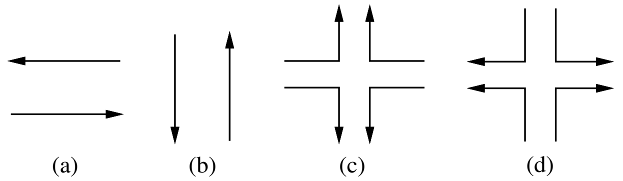

Figure 5: (a), (b), (c), (d): Respective Cases 1, 2, 3, and 4 routing patterns for longlines.

where $e \in E$, $\vec{b_i} = (b_{i1}, \ldots, b_{iK_i})$ is the current *density vector* for the type-i subchannel, b_{ij} is the density in jth subchannel of type-i segments, and K_i is the number of subchannels of type-i segments.

The whole procedure for the simultaneous placement and global routing is described as follows. Given an FPGA F, we first construct a graph G_F to model F. Initially, $b_{ij} \leftarrow 0$, $i \in \mathcal{T}$, $1 \leq j \leq K_i$, for each subchannel j of type-i. The placement stage runs in the cluster-growth manner. The pin (and thus the logic block) with the most connectivity will be placed and routed first. We choose the unplaced pin (block) that is most strongly connected to the placed pin and place the selected pin at a block so that the cost for the route p, $\phi_i(p), i \in \mathcal{T}$, is minimized. Note that for each net we try all types of subchannels to "virtually route" the target connection using the maze router and pick the one with the minimum cost and place the selected pin accordingly. After a net is routed, $\vec{b_i}$ needs to be updated to reflect the additional congestion resulting from the routing of the net. The process continues as placement and global routing proceeds.

5 Experimental Results

Our algorithm APR was implemented in Borland C++ Builder 3.0 Clint/Server Suite on a Pentium-II Celeron 366 PC with 128-MB RAM. We implemented the aforementioned simultaneous placement and global routing algorithm based on our architecture-driven metric which considers segments of multiple lengths and subchannel density. For the purpose of comparison, we also implemented the *same* simultaneous placement and global routing algorithm based on the traditional metric of geometric distance and channel density. The traditional metric, $\Phi' : p(u,v) \rightarrow \Re^+ \cup \{0\}$, is given by

$$\Phi'(p) \quad = \quad \alpha'(|u_x - v_x| + |u_y - v_y|) + \gamma' \sum_{e \in C_e, e \in p(u,v)} b_e. (7)$$

where $|u_x - v_x| + |u_y - v_y|$ is the geometric (Manhattan) distance between u and v, and b_e is the channel density associated with edge e. For the purpose of fair comparison for the two metrics, we let $\alpha = \alpha'$ and $\gamma = \gamma'$.

Circuit	#Logic blocks	#Nets	#2-pin connections
9symml	72	89	259
alu2	143	153	511
alu4	255	256	851
apex7	80	126	300
C1355	110	145	360
C499	110	145	360
C880	120	174	427
example2	120	205	444
k2	360	404	1256
term1	56	88	202
too_large	156	186	519
vda	210	225	722
z03D4	586	608	2135

Table 1: Characteristics of Benchmark Circuits.

We tested the two metrics on the famous VPR/SEGA benchmark circuits used in [3, 4]. Table 1 gives the names of the circuits, the numbers of logic blocks in the FPGAs, the number of nets, and the numbers of two-terminal connections in the circuit. As shown in Figure 6, we first ran the program sega2blif in SEGA [4] to convert the benchmark circuits to the BLIF format and then blifmap in VPR [3] to perform synthesis and logic block packing. The resulting netlists of logic blocks were then fed into APR with the respective metric (i.e., architecture-driven or geometric-distance metric) to obtain the placement and global routing results; the results were then fed into the SEGA detailed router [4] to complete the routing. Finally, we used the timing analyzer *horowitz* embedded in SEGA to compute the net delays. The overall CAD flow is illustrated in Figure 6.

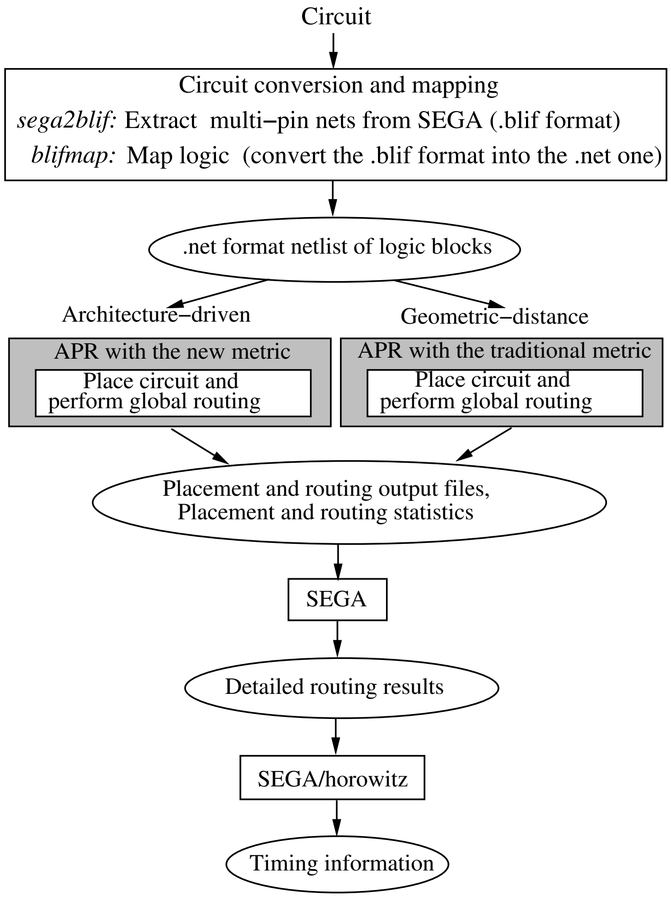

Figure 6: The CAD flow for our experiments.

Our experiments were based on the Lucent Technologies ORCA2C-like and the Xilinx XC4000EX-like switch-block and segmentation architectures. The numbers of single-length lines, double-length lines, quad-length lines and longlines used are proportional to those used in the ORCA2C and XC4000EX FPGAs; specifically, the respective numbers of tracks of single-length lines, double-length lines, quad-length lines, and longlines were $W - \lfloor 2W/5 \rfloor - \lceil W/5 \rceil$, 0, $\lfloor 2W/5 \rfloor$ and $\lceil W/5 \rceil$ for the ORCA2C-like parameters and $W - \lfloor 2W/15 \rfloor - \lceil 2W/5 \rceil - \lceil W/5 \rceil$, $\lfloor 2W/15 \rfloor$, $\lfloor 2W/5 \rfloor$, and $\lceil W/5 \rceil$ for the XC4000EX-like parameters, where W is the number of tracks in a channel. The switch-block architecture for both the ORCA2C-like and the XC4000EX-like FPGAs is shown in Figure 7. For the architecture-driven metric, we set $\alpha = 0.81$, $\beta = 0.09$, and $\gamma = 0.10$ in the experiment. Note that we chose a significantly larger value for α because the number of segments used by a net, as mentioned earlier, is the most crucial factor in controlling the delay and resource utilization in a circuit.

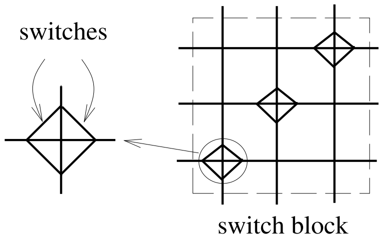

Figure 7: The switch-block architecture.

To explore the effects of the two metrics, all benchmark circuits were placed and routed using the same net order. Because net order often affects the performance of a placer and router, we used the following three net-ordering schemes to avoid possible biases:

- Original net order: Sequentially extract a net from a netlist file.
- Random net order: Randomly extract a net from a netlist file.
- Cluster growth-based net order: Extract a net from a netlist file based on the connectivity of placed blocks.

Since our main goal is to make fair comparisons for the two metrics, no rip-and-reroute phase was incorporated in the APR router (optimization might bias the comparison).

Tables 2 and 3 list the experimental results, in which the respective columns associated with "Geometric-distance" and "Architecture-driven" give the results obtained by using the traditional metric of geometric distance and channel density and the architecture-driven metric with the consideration of segments of multiple lengths and subchannel density. Columns denoted by W, D_{max}, and D_{avg} give the total number of tracks required for routing completion, the critical net delay, and the average net delay, respectively. The results show that, no matter what net order was used, the architecture-driven metric outperforms the traditional one in both area and speed performance with *average* improvements of 8%, 20%, and 19% in the number of tracks for routing completion (area), the critical net delay (speed), and the average net delay (speed), respectively. The results show the effectiveness of our architecture-driven metric. The runtimes for APR based on the architecture-driven metric ranged from 3 sec to 12 min for placing and globally routing a circuit listed in Tables 2 and 3.

We also tested the effects of the parameters α (α'), β, and γ (γ') on speed performance. Figure 9 shows the average net delays for the benchmark circuit k2 obtained by fixing the parameter $\beta = 0.09$ and setting α (α') from 0.1 to 0.9 and $\gamma(\gamma') = 1 - \alpha - \beta$, using the cluster-growth based net order and the Xilinx XC4000EX-like architecture. Note that α (α') is associated with the delay cost and γ (γ') the area cost. Setting α (α') close to 1.0 leads to timing-driven placement and routing for both metrics—the architecture-driven metric is based on the number of segments used while the traditional metric is based on the geometric distance traveled by a net. As shown in Figure 9, the architecture-driven metric consistently outperforms the traditional one by large margins. In particular, the greater the α (α'), the larger the difference, which reflects the phenomenon that the number of segments, instead of geometric distance, traveled by a net is the most crucial factor in controlling the routing delay in an FPGA. Though not presented here, the results for other circuits are similar.

It should be noted that what we have presented is a new metric which is intended for *general* FPGA placement and global routing formulation such as the min-cut, Steiner-tree, shortest-path based algorithms, etc [18]. Thus the most important concern shall be the investigation of the effects of the two metrics. This is why we kept our

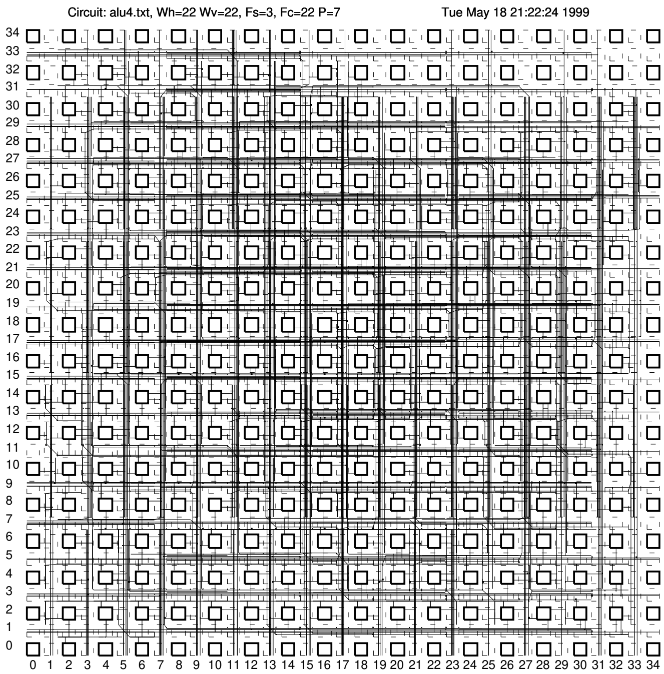

Figure 8: The routing solution for alu4 obtained by SEGA fed by the global routes generated from APR based on the ORCA2C-like architecture-driven metric and the cluster growth-based net order ($W = 22$).

experimental factors simple; for instance, we used the cluster growth based placer and the maze router to explore the effects, and no optimization such as rip-up and re-route was incorporated.

6 Concluding Remarks

We have presented an architecture-driven metric for simultaneous placement and global routing for FPGAs. The metric considers routing segments of multiple lengths and subchannel density. Experimental results show that it is superior to the traditional metric of geometric distance and channel density in both area and speed performance. In this paper, we applied the greedy cluster growth-based placer and the maze router as a vehicle to *demonstrate* the effects of the two metrics on routing quality; the architecture-driven metric, however, is intended for *general* FPGA placement and global routing formulations such as the min-cut, Steiner-tree, shortest-path based algorithms, etc.

Acknowledgments

We thank the authors of [4] for the SEGA package and Dr. Vaughn Betz and Guy Lemieux of University of Toronto for their helpful comments on the package.

References

[1] M. J. Alexander, et. al., "Performance-Oriented placement and routing for field-programmable gate arrays," *Proc. of EuroDAC*, pp. 80–85, 1995

[2] M. Alexander and G. Robins, "New performance-driven FPGA routing algorithm," *Proc. of DAC*., pp. 562-567, June 1995.

[3] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*, Kluwer Academic Publishers, 1999.

[4] S. Brown, G. Lemieux, and M. Khellah, "Segmented routing for speed-performance and routability in field-programmable gate arrays", *Journal of VLSI Design*, 4(4), pp. 275–291, 1996.

[5] Y.-W. Chang, S. Thakur, K.Zhu, and D. F. Wang, "A new global routing algorithm for FPGAs," *Proc. of ICCAD*, pp. 380–385, 1994.

Circuit Name	Original net order						Random net order						Cluster-based net order					
	Geometric-distance			Architecture-driven			Geometric-distance			Architecture-driven			Geometric-distance			Architecture-driven		
	W	D_{max}	D_{avg}	W	D_{max}	D_{avg}	W	D_{max}	D_{avg}	W	D_{max}	D_{avg}	W	D_{max}	D_{avg}	W	D_{max}	D_{avg}
9symml	14	61.2	9.7	13	6.6	7.6	14	70.3	9.8	14	71.2	9.9	14	68.9	9.6	12	64.8	7.4
alu2	21	144.0	23.2	20	95.9	15.3	21	193.2	24.3	20	97.4	17.2	19	96.4	15.1	18	92.2	12.0
alu4	24	140.2	26.4	23	96.5	19.4	26	164.2	27.1	24	126.1	20.1	27	178.1	26.2	22	95.9	17.7
apex7	13	60.5	9.2	13	47.9	9.1	15	63.9	9.7	13	49.7	11.4	16	30.7	8.3	13	48.8	8.7
C1355	17	76.3	9.6	16	54.3	8.9	18	140.6	11.2	17	66.3	8.9	17	70.2	9.7	16	48.6	8.4
C499	15	75.3	10.2	15	69.2	8.8	16	79.2	10.7	16	96.6	10.2	15	73.2	10.0	14	66.5	8.6
C880	15	78.8	11.8	15	45.6	11.9	17	79.8	12.7	16	115.7	12.5	15	43.5	9.4	15	44.3	9.1
example2	17	111.7	11.2	17	123.4	10.9	19	189.2	12.3	19	209.6	12.1	16	71.2	9.7	16	80.3	9.3
k2	36	230.8	31.0	35	184.8	26.1	38	304.5	31.5	36	197.1	27.3	34	220.3	30.1	34	181.4	25.4
term1	11	30.6	6.8	9	28.9	6.5	12	40.1	7.2	10	47.0	7.7	11	25.6	5.6	9	20.7	5.3
too_large	24	129.1	16.4	22	80.2	14.8	26	143.4	17.1	23	80.2	15.2	22	133.2	16.3	22	78.4	13.5
vda	25	143.2	21.9	22	108.4	15.3	27	166.3	21.9	24	109.4	16.5	25	141.9	21.1	20	107.5	14.7
z03D4	35	203.7	22.6	35	194.2	20.9	39	382.9	24.4	37	373.7	22.2	33	163.6	20.9	33	129.4	20.4
Avg	20.5	114.3	16.2	19.6	91.5	13.5	22.2	154.6	16.9	20.7	126.2	14.7	20.3	101.3	14.8	18.7	81.5	12.3
Comp.	**1.05**	**1.24**	**1.19**	**1.00**	**1.00**	**1.00**	**1.07**	**1.23**	**1.15**	**1.00**	**1.00**	**1.00**	**1.08**	**1.24**	**1.19**	**1.00**	**1.00**	**1.00**

Table 2: The SEGA detailed routing results based on the Lucent Technologies ORCA2C-like architectures and the placement and global routing data obtained by using the two metrics.

Circuit Name	Original net order						Random net order						Cluster-based net order					
	Geometric-distance			Architecture-driven			Geometric-distance			Architecture-driven			Geometric-distance			Architecture-driven		
	W	D_{max}	D_{avg}	W	D_{max}	D_{avg}	W	D_{max}	D_{avg}	W	D_{max}	D_{avg}	W	D_{max}	D_{avg}	W	D_{max}	D_{avg}
9symml	14	55.1	9.3	12	63.3	7.1	15	56.8	9.5	12	64.1	9.3	14	51.1	9.3	12	63.4	7.1
alu2	22	100.6	21.5	19	88.2	13.4	22	135.0	21.7	20	94.5	15.2	22	100.5	15.3	19	93.5	12.0
alu4	26	133.2	24.8	23	112.6	18.4	27	135.2	25.1	24	127.2	18.8	26	132.0	24.8	23	96.9	16.0
apex7	15	52.1	9.1	12	49.1	8.8	15	55.8	11.2	13	52.3	11.1	15	30.5	8.3	12	47.9	8.3
C1355	15	87.7	9.7	14	56.2	8.5	15	117.9	11.3	15	67.9	8.8	15	67.5	9.6	14	46.5	8.0
C499	19	81.1	9.7	15	76.2	9.3	20	82.6	10.8	15	102.2	10.6	19	79.2	10.5	15	67.8	8.2
C880	19	57.0	11.9	19	48.1	11.5	19	80.5	12.3	20	77.1	11.8	19	46.0	9.9	19	47.0	9.2
example2	19	126.6	10.7	19	124.7	10.5	20	187.3	12.6	19	212.0	11.8	18	72.8	9.6	18	67.3	9.8
k2	35	182.6	28.9	34	174.8	23.9	37	206.9	29.8	37	177.1	25.1	34	180.3	28.9	33	174.2	23.8
term1	12	31.3	6.6	10	28.9	6.4	12	40.9	7.8	10	48.4	7.6	12	26.3	5.8	10	21.3	5.1
too_large	22	117.7	17.4	22	84.4	15.1	24	127.3	16.7	23	90.0	15.5	22	113.7	15.8	22	66.1	12.9
vda	25	153.7	20.2	23	67.1	14.1	27	159.2	21.8	23	78.2	15.7	25	152.9	20.1	22	72.4	13.4
z03D4	38	163.9	21.9	37	185.6	19.9	41	191.8	22.2	39	192.7	21.1	35	153.1	20.3	35	143.5	19.8
Avg	21.6	103.2	15.5	19.9	89.2	12.8	22.6	119.0	16.3	20.8	106.4	14.0	21.2	92.8	14.4	19.5	77.5	11.8
Comp.	**1.08**	**1.16**	**1.20**	**1.00**	**1.00**	**1.00**	**1.09**	**1.12**	**1.17**	**1.00**	**1.00**	**1.00**	**1.08**	**1.19**	**1.22**	**1.00**	**1.00**	**1.00**

Table 3: The SEGA detailed routing results based on the Xilinx XC4000EX-like architectures and the placement and global routing data obtained by using the two metrics.

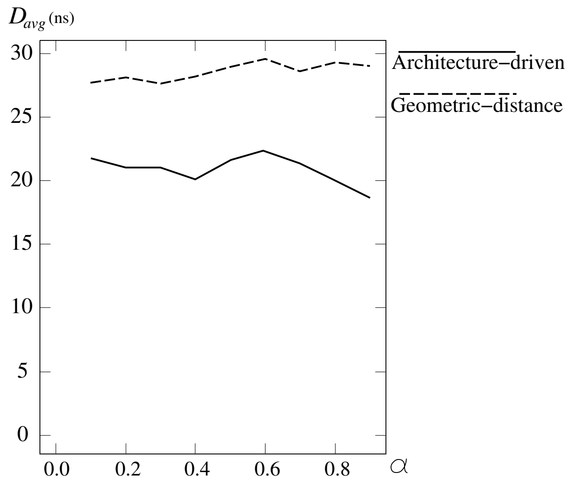

Figure 9: Average net delays for k2 obtained by fixing $\beta = 0.09$ and setting α from 0.1 to 0.9 and $\gamma = 1 - \alpha - \beta$ (based on the cluster-growth based net order and the XC4000EX-like architecture).

[6] Y.-W. Chang, K. Zhu, and D. F. Wong, "Timing-driven routing for symmetrical-array-based FPGAs," *ACM Trans. on Design Automation of Electronic Systems*, Vol. 5, No. 3, July 2000.

[7] C. -D. Chen, Y. -S. Lee, C. -H. Wu, and Y. -L. Lin, "TRACER-fpga: a router for RAM-based FPGA's," *IEEE Trans. on CAD*, vol. 14, no. 3, pp. 371–374, Mar. 1995.

[8] C. Ebeling, L. McMurchie, S. A. Hauck, and S. Burns, "Placement and routing tools for the triptych FPGA," *IEEE Trans. on VLSI Systems*, pp. 473–482, Dec. 1995.

[9] J. Frankle, "Iterative and adaptive slack allocation for performance-driven layout and FPGA routing," *Proc. of DAC*, pp. 536–542, 1992.

[10] M. Khellah, S. Brown, and Z. Vranesic, "Modelling routing delays in SRAM-based FPGAs," *Proc. of Canadian Conf. on VLSI*, 1993.

[11] C. Y. Lee, "An algorithm for path connection and its application", *IRE Trans. Electronic Comput.*, Vol. EC-10, pp 346–365, 1961.

[12] Y.-S. Lee and C.-H. Wu, "A performance and routability driven router for FPGA's considering path delay," *Proc. of DAC*, pp. 557-561, 1995.

[13] G. G. Lemieux and S. D. Brown and D. Vranesic, "On two-step routing for FPGAs," *Proc. of ISPD*, pp. 60–66, Apr. 1997.

[14] Lucent Technologies Inc., *ORCA OR2CxxA (5.0 V) and OR2TxxA (3.3 V) Series Field-Programmable Gate Arrays*, Aug. 1996.

[15] L. McMurchie and C. Ebeling, "Pathfinder: A negotiation-based performance-driven router for FPGAs," *Proc. of FPGAs*, pp. 111–117, 1995.

[16] S. K. Nag and R. A. Rutenbar, "Performance-directed simultaneous place and route for Island-style FPGA's," *Proc. of ICCAD*, pp. 332–338, 1995.

[17] S. K. Nag and R. A. Rutenbar, "Performance-driven simultaneous placement and routing for FPGA's," *IEEE Trans. on Computer-Aided Design*, Vol. 17, No. 6, pp. 499–518, June 1998.

[18] N. Sherwani, *Algorithms for VLSI Physical Design Automation*, 2nd Ed., Kluwer Academic Pub., 1995.

[19] N. Togawa, M. Sato, and T. Ohtsuki, "Maple: A simultaneous technology mapping placement and Global routing Algorithm for field-programmable gate arrays," *Proc. of ICCAD*, pp.156–163, 1994.

[20] Y.-L. Wu and M. Marek-Sadowska, "Orthogonal greedy coupling —A new optimization approach to 2-D FPGA routing," *Proc. of DAC*, June 1995.

[21] Xilinx Inc., *The Programmable Logic Data Book*, 1996.

[22] K. Zhu, Y.-W. Chang, and D. F. Wong, "Timing-driven routing for symmetrical-array-based FPGAs," *Proc. of ICCD*, October 1998.

[23] K. Zhu and D. F. Wong, "Switch bound allocation for maximizing drRoutability in timing-driven routing of FPGA's," *IEEE Trans. on CAD*, vol. 17, no. 4, pp.316–323, Apr. 1998.