Global Bus Route Optimization with Application to Microarchitectural Design Exploration

Dae Hyun Kim and Sung Kyu Lim
School of Electrical and Computer Engineering
Georgia Institute of Technology
{*daehyun, limsk*}*@ece.gatech.edu*

***Abstract*— Circuit and processor designs will continue to increase in complexity for the foreseeable future. With these increasing sizes comes the use of wide buses to move large amounts of data from one place to another. Bus routing has therefore become increasingly important. In this paper, we present a new bus routing algorithm that globally optimizes both the floorplan and the bus routes themselves. Our algorithm is based on creating a range of feasible bus positions and then using Linear Programming to optimally solve for bus locations. We present this algorithm for use in microarchitectures and explore several different optimization objectives, including performance, floorplan area, and power consumption. Our results demonstrate that this algorithm is effective for efficiently generating feasible routes for complex modern designs and provides better results than previous approaches.**

I. Introduction

As systems become larger and more complicated, global routing of signal wires becomes increasingly important for objectives such as performance and power consumption. These objectives, however, are strongly related to various constraints, such as routability and temperature. For example, a minimum bus area objective increases performance and decreases the power consumption needed to drive buses, but it requires the use of the center location of the floorplan for routing. This necessarily entails decreased routability and increased temperature. Because all of these factors are interrelated, they should be optimized simultaneously.

Recent trends towards large numbers of signal wires have generated interest in bus routing rather than single-wire routing [1]. Using a simple approximation, the runtime of single-wire routing is N times bigger than that of bus routing with an N-bit bus. In microprocessors and DSPs that have many cores, interconnects are major performance bottlenecks. The ever increasing number of bits used in modern designs causes the need for bus routing to be beyond doubt. Yet, bus routing is totally different from single-wire routing because of the significant widths of buses. This makes previous global routing algorithms ([2], [3], [4], [5], [6]) unusable for this application.

Previous works [1], [7], [8], [9] addressed various global bus routing algorithms. However, they are either incomplete in terms of mixed objectives, such as routability, execution time, and performance, or not applicable to complicated routing problems, such as in microprocessors where many functional modules are connected with wide buses. [10] tries to solve global bus routing problems with a straightforward method. However, it routes buses one by one so this approach is inferior to concurrent methods.

This material is based upon work supported by the National Science Foundation under CAREER Grant No. CCF-0546382 and MARCO C2S2.

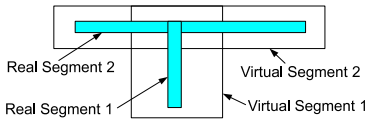

Fig. 1. Virtual segments and Real segments

In this paper, we present a new bus routing algorithm (**H-LP**) based on Hanan grids [11] and linear programming (LP). We first describe our algorithm, compare it with other algorithms, and show its application to a 64-bit microarchitecture. Our experiments show the efficiency of H-LP with respect to various objectives, such as runtime, area, and performance. The contributions of this paper are as follows:

- We develop a new bus routing algorithm (H-LP) based on linear programming. It routes all nets simultaneously and overcomes the limitation of the sequential method.
- Our algorithm consistently outperforms several well-known algorithms in the literature in terms of routability, performance, and area.

II. Problem Formulation

The bus routing problem is formulated as follows. Given:

1) Two metal layers (for horizontal and vertical)
2) Wire pitch (= wire width + spacing)
3) A set $B = \{B_1, B_2, ..., B_{N_B}\}$ of N_B blocks, each with a width and a height
4) A set $H = \{H_1, H_2, ..., H_{N_H}\}$ of N_H buses, each with the specified number of bits and a netlist

The constraints are:

1) No overlap exists between any two blocks.
2) No overlap exists between any two vertical (or horizontal) segments from two different buses.
3) Each bus should connect all blocks in its netlist.

The problem is to find the positions of all blocks and buses such that the constraints are satisfied while optimizing specified goals such as bus area or performance.

Before explaining the algorithm, we introduce two terminologies, *virtual segment* and *real segment*. A *virtual segment*

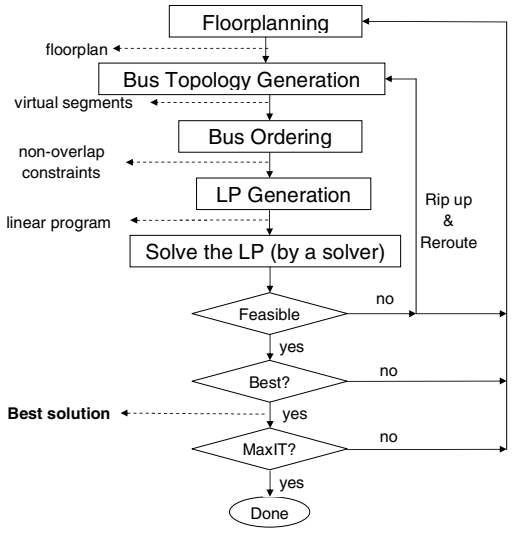

Fig. 2. H-LP bus routing algorithm.

is a range for the feasible placement of a *real segment*. A bus consists of *real segments* as in Figure 1.

III. Bus Routing Algorithm

Our bus routing algorithm (H-LP) is divided into five steps. Figure 2 shows the overall algorithm of H-LP. Each of the steps is explained in the following subsections.

A. Floorplanning

We use the Sequence Pair [12] floorplan representation and simulated annealing to floorplan. We employ three moves - swapping two modules in one of the sequences, swapping two modules in both sequences, and rotation of a module. The cost function is as follows.

$$F_{cost} = \alpha \cdot A + \beta \cdot R + \gamma \sum_{H_i \in H} (AF_i \cdot BA_i) \qquad (1)$$

In equation (1), α, β and γ are weighting factors, A is floorplan area, and R is a routability constant. R is either R_1(for successful routing) or R_2 (for failed routing). We explain this in detail in Section IV-D. AF_i and BA_i are access frequency and bus area of bus H_i, respectively. The access frequency is computed based on how many times the bus is used during architectural simulations. It is shown in [13] that the minimization of this factor improves the performance of the underlying architecture measured in terms of IPC (instructions per cycle). The bus area is related to power consumption objective.

B. Bus Topology Generation

In this step, we generate a bus topology for each bus one by one for a floorplan. Notice that this step generates virtual segments and we find real segments with Linear Programming.

We start bus topology generation by making a bounding box and extending the borders of all the blocks in a bus to make the Hanan grids within it. An example is shown in Figure 3.

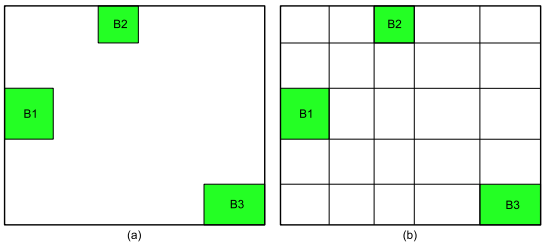

Fig. 3. (a) Blocks (B_1, B_2, and B_3) covered by a bus. (b) Hanan grids made from (a).

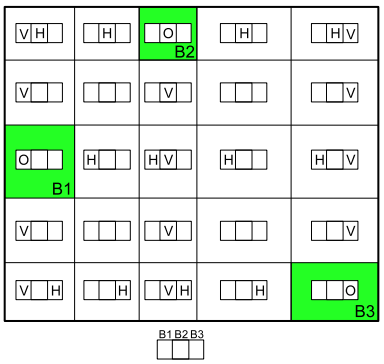

Fig. 4. Hanan grid marked with V/H/O.

The reason we do not use uniform grids is that uniform grids make alignment of segments harder. We restrict the routing range to the inside of the bounding box in order to prevent long detours.

Next, each grid is marked with V, H, or O as in Figure 4. V or H mark corresponding to B_n means the grid can be connected to the block B_n by a vertical or a horizontal segment respectively. O means the grid overlaps the block B_n so that it can be used for either a horizontal segment or a vertical segment. We define the *degree* of a grid as the number of marks on it.

The next step is to heuristically choose grids to route all the blocks connected by the bus. We first compute weights of all grids as follows.

$$S_V = \sum_{B_j \in 'V'} |CY(B_j) - CY(G_i)|$$

$$S_H = \sum_{B_k \in 'H'} |CX(B_k) - CX(G_i)|$$

$$W(G_i) = \frac{S_V + S_H}{degree} \qquad (2)$$

where $W(G_i)$ is a weight of grid G_i, and $CX(P)$ and $CY(P)$ are respectively X and Y coordinates of the center of a block or a grid P. Then, we sort all grids in decreasing order of *degree*. If there is a tie, a grid having smaller weight is chosen first. Then we select a seed grid that has at least *degree* 2 because grids of *degree* 1 have no connection point, and we iterate grid selection until we cover all blocks to be connected as in Figure 5. We show an example in Figure 6.

Two routings for the same bus are compared in Figure 6(b)

```
1: Sort all grids
2: Select the first grid
3: Form virtual segments
4: if unrouted blocks remain then
5:     Select a grid
6:     Form virtual segments
7: end if
8: Return virtual segments
```

Fig. 5. Grid selection algorithm.

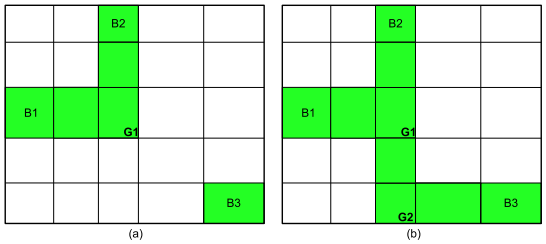

Fig. 6. (a) B_1 and B_2 are routed through G_1. (b) B_3 is routed by G_2.

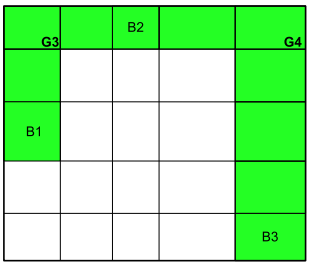

Fig. 7. Different routing obtained by choosing grids G_3 and G_4.

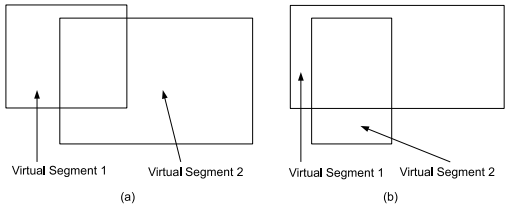

Fig. 8. Bus ordering for two vertical segments. (a) [left:segment 1, right:segment 2] is better. (b) [left:segment 2, right:segment 1] is better.

and Figure 7. We assume that B_1 is a source and B_2 and B_3 are sinks. Figure 6(b) has minimum bus area and minimum SSL (Source-to-Sink Length) for all sinks. On the other hand, Figure 7 has bigger bus area and bigger SSL for the sink B_3. However, we can benefit from the routing in Figure 7 as the bus does not occupy the center region thereby decreasing congestion and increasing the chance of generating a feasible LP. We compare this algorithm with [14] for Bus Topology Generation in Section V-B.

C. Bus Ordering

The segments generated from the previous step are virtual. We find real segments by converting the virtual segments into a LP and solving it. However, we need to satisfy non-overlap conditions between horizontal (or vertical) segments from different buses. Without a bus ordering step, we need Integer Linear Programming (ILP) to formulate non-overlap constraints. ILP formulation for two vertical segments S_i and S_j is as follows.

$$\begin{aligned} X_{S_i} + W_{S_i} &< X_{S_j} + Z_{i,j} \cdot M_{max} \\ X_{S_j} + W_{S_j} &< X_{S_i} + (1 - Z_{i,j}) \cdot M_{max} \end{aligned} \tag{3}$$

In the above equation, $Z_{i,j}$ is a binary variable which is zero or one, X_p is a x-coordinate of bottom-left corner of the segment p, and W_p is the width of the segment p. M_{max} is a large value. ILP formulation for y-coordinates is similar.

Solving ILP, however, needs huge amount of time if there are lots of integer variables. We show comparisons of ILP with LP in Section V. In summary here, we definitely need bus ordering if the number of bus increases.

In the bus ordering step for LP formulation, we compare the relative locations of two virtual segments having same direction. For example, if two segments S_i and S_j are vertical, and the left side of S_i is to the left of bus S_j, having the real segment of S_i be to the left of the real segment of S_j is better because it gives us higher degree of freedom. We also compare the right sides of S_i and S_j to refine the ordering as in Figure 8. This increases the chances of creating a feasible LP formulation.

D. LP Generation

After bus ordering, we convert the constraints into LP. The objective of the LP can be varied according to our goal; however, constraints cannot be changed. LP generation is explained in detail in Section IV. We use version 5.5.0.12 of lp_solve [15] to solve the LP.

E. Rip up and Re-route

Routing failure comes from an infeasible LP caused by congestion. When the routing fails, we first revisit the Bus Topology Generation step and try to make different bus topologies to make the LP feasible. If this does not provide feasible solutions after several iterations, we go back to the Floorplanning step and perturb the current floorplan as in Figure 2. The Bus Ordering step also has room to make the LP feasible but its effect is small.

There are various techniques for making different bus topologies, such as changing the weights in Equation (2) or selecting a different seed grid. Our choice is to include a virtual segment congestion map. For this, we create a uniform grid for the entire floorplan to see where congestion occurs. When a virtual segment is placed, we add an expectation value to the grids covered by the virtual segment. This expectation value is computed using the following equation:

$$EX = \frac{BW}{SH} \cdot GH \tag{4}$$

where BW is bus width, SH is the virtual segment height (for a horizontal segment) and GH is the grid height. This can be refined if a more accurate model is needed.

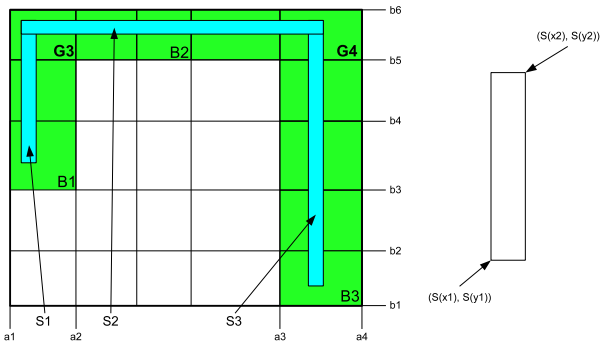

Fig. 9. Coordinates of the segments.

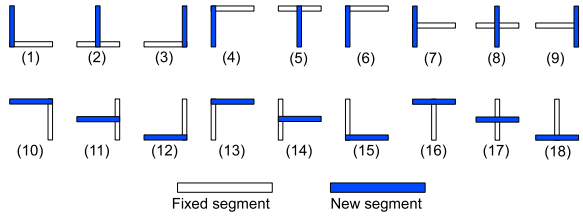

Fig. 10. 18 connection cases between two segments

Using the congestion map, we rip up virtual segments in congested regions and choose other grids for those buses. The LP is then reformulated and solved. If we fail to formulate a feasible LP within a pre-determined number of iterations then we go back to the Floorplanning step.

IV. LP Formulation

The inputs needed for LP formulation are the set of virtual segments and the orders of the segments. We show LP formulation with an example in Figure 9.

A. Constraints

We have two types of constraints in LP formulation. The first constraint is the allowed range of segments. For example, segment S_1 in Figure 9 should connect the grid B_1 (which is also a block) to the grid G_3. Therefore the bottom and the top of S_1 should be inside B_1 and G_3 respectively.

Moreover, S_1 should be connected to S_2, so dependencies exist among connected segments. There are 18 connection cases as in Figure 10. Here we show a formulation only for case (1) because of limited space. Let $F(\cdot)$ be the coordinate of a fixed segment and $N(\cdot)$ be the coordinate of a new segment connected to the fixed segment. Then the coordinates of the new segment of case (1) are:

$$\begin{aligned} N(x_1) = F(x_1) \quad , \quad N(y_1) = F(y_1) \\ N(x_2) = N(x_1) + BW = F(x_1) + BW \end{aligned} \tag{5}$$

If there is no segment connected to the top of the new segment, $N(y_2)$ becomes a constant. If it is connected to another segment, the range of $N(y_2)$ is affected by the segment.

TABLE I

Comparison of LP with ILP for bus ordering (Scaled to values from LP). $\#OVR$ is the average number of non-overlap conditions.

	#blk	#bus	$Time$	$BusArea$	$Routability$	$\#OVR$
apte-s	9	5	1.02	0.91	1	6.6
apte	9	36	∞	-	-	380
xerox-s	10	6	0.94	0.96	1	12.4
xerox	10	98	∞	-	-	3486
μarch	20	51	∞	-	-	982

B. Constraints - Non-overlap

If a vertical segment V_1 is supposed to be in the left of a vertical segment V_2, the constraint is formulated as follows:

$$V_1(x_1) + BW_1 \leq V_2(x_1) \tag{6}$$

y-coordinates of horizontal segments are formulated similarly.

C. Objective Function

We hand an objective function and constraint inequalities to a LP solver. We show two examples in the following:

$$Bus\ Area\ :\ \sum_{\forall segments} (S(x_2) - S(x_1)) \cdot (S(y_2) - S(y_1)) \tag{7}$$

Notice that the equation (7) is linear because $S(y_2) - S(y_1)$ and $S(x_2) - S(x_1)$ are BW when the segment is horizontal and vertical, respectively.

$$SSL\ :\ \sum_{H_i \in H} F(f_i) \cdot SSL_i \tag{8}$$

SSL has a strong relation with performance. In Equation (8), $F(f_i)$ is a constant obtained from access frequency [13] of bus H_i and SSL_i is the SSL of bus H_i.

D. Routability

Here we explain the R value in Equation (1). Since real segments are determined by LP, we have coordinates for all buses if the LP is feasible. Otherwise, we have no coordinates. Therefore the routability ($= \frac{No.\ of\ routed\ buses}{No.\ of\ buses}$) of H-LP is either 0% or 100%. Thus we need only two constants R_1 and R_2 for R.

V. Experimental Results

We implemented H-LP using C++. We use two benchmarks to compare bus routers. The first is MCNC benchmark and the second is μarch used in [10]. The widths of the widest bus and the most narrow bus of μarch are, respectively, 512 bits and 7 bits. The average is 107.6 bits. SimpleScalar [16] and the SPEC 2000 benchmarks [17] are used for IPC measurement.

TABLE II

COMPARISON OF [14] WITH OUR ALGORITHM FOR BUS TOPOLOGY GENERATION (SCALED TO VALUES FROM OUR ALGORITHM).

	#blk	#bus	$BusArea$	$Routability$
apte-s	9	5	0.99	1
apte	9	36	1.04	1
xerox-s	10	6	1.00	1
xerox	10	98	1.08	0.95
μarch	20	51	1.07	0.84

TABLE III

COMPARISON WITH [7] AND [10]. R DENOTES ROUTABILITY IN PERCENTAGE AND A DENOTES THE FLOORPLAN AREA IN mm^2.

			[7]		[10]		H-LP	
	#blk	#bus	R	A	R	A	R	A
apte	9	36	80.9	55.46	100	48.77	100	48.66
xerox	10	98	79.8	22.2	100	21.54	100	21.23
hp	11	37	76	12.57	100	14.05	100	13.45
ami33	33	53	78.6	1.42	100	1.31	100	1.32
ami49	49	156	81.6	39.7	100	43.72	100	43.14

A. LP vs ILP for Bus Ordering

We first compare the results of bus ordering by LP and ILP in Table I. $\#OVR$ is the average number of non-overlap conditions, so it is same as the number of binary variables in ILP. As Table I shows, the runtime to solve LP and ILP is similar when the number of buses is small, and the bus area of ILP is 5 - 10% smaller. However, it takes a very long time to solve ILP when there are lots of buses. In our experiment, solving apte (#blk:9, #bus:36) did not finish within 7 days. This shows why we need bus ordering step along with LP relaxation.

B. Bus Topology Generation Algorithm Comparison

The Steiner Min-Max Tree (SMMT) [14] is a well-known Steiner tree construction algorithm to avoid congestion in multiple net routing. We compare SMMT with our algorithm for Bus Topology Generation. We used same bus ordering step for both algorithms for fair comparisons. Routability for each algorithm is computed as:

$$\frac{\#\ of\ successful\ trials}{\#\ of\ total\ trials}$$

These values are shown in Table II. We observe that bus areas are comparable, but routability of [14] is lower compared to that of ours. This shows that our algorithm is simple but well-suited for Bus Topology Generation.

C. Comparison with Previous Works

We next compare H-LP with [7] and [10] in Table III. We observe that H-LP is consistently outperforming [7] and [10] in terms of routability and floorplan area. Note that [7] considers timing constraints and performs buffer/channel insertion, which explains routing failures in most cases.

In Table IV, we compare H-LP with [9] and [10]. We observe the following:

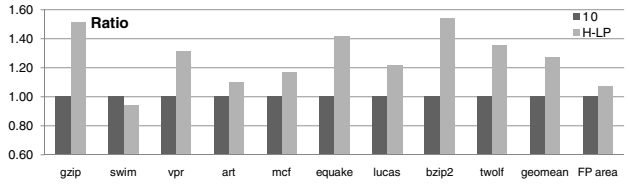

Fig. 11. IPC Comparison of Performance-driven routing (scaled to [10]).

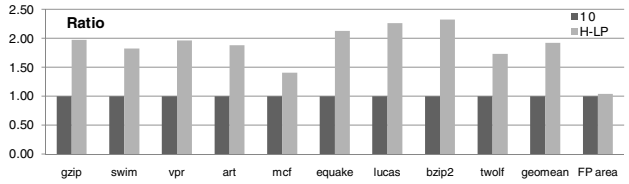

Fig. 12. IPC Comparison of Area-driven routing (scaled to [10]).

- The scalability and efficiency of H-LP becomes evident for large circuits such as ami49 and μarch. Minimizing bus area is an important goal because it affects not only power consumption for driving gates, but also the thermal profile which is caused by joule heating [18], [19]. H-LP always has smaller bus area while keeping the floorplan area similar to that of [10].
- H-LP runs slower than [10] for a few small benchmarks, such as apte, xerox or hp, but still faster than [9]. In fact, the runtime of H-LP for small benchmarks can be significantly decreased if we do floorplanning without routing at the beginning and do routing for the last few floorplans.
- [9] does not work well for complex benchmarks, such as μarch, as it uses restricted shapes of buses. [10] or H-LP, however, have higher routability even for those benchmarks because of the flexibility of bus shapes.

D. Applications to Microarchitecture

Figures 11 through 13 show comparisons of IPC (Instructions Per Cycle) for three objectives.[1] We observe that our LP-based concurrent approach consistently outperforms [10] that is based on sequential approach. In addition, Figure 14 proves that our performance-objective function works well to boost performance results. However, there is a trade-off between performance, area and power as shown in Table V. High performance comes at a cost of slightly larger floorplan area, bus area, and power consumption.

VI. CONCLUSION

As the complexity of modern designs reaches ever higher, bus routing has become of extreme importance to ensure ceaseless performance increase. In this paper, we developed a new bus routing algorithm (H-LP) based on Hanan grids and Linear Programming. Our algorithm globally optimizes the placement of all buses to ensure performance. Also our algorithm is capable of optimizing multiple objectives for

[1]We compare only with [10] because [9] could not route all buses. We need 100% routability to get the delay of buses.

TABLE IV

COMPARISON WITH [9] AND [10]. T DENOTES RUNTIME IN SECONDS, A DENOTES THE FLOORPLAN AREA IN mm^2, BA DENOTES THE BUS AREA IN mm^2 AND R_{max} DENOTES MAXIMUM ROUTABILITY IN TERMS OF % ($=\frac{\#\ of\ routed\ buses}{\#\ of\ buses} \times 100$). (A/M = AVERAGE/MAXIMUM NUMBER OF BLOCKS A BUS CONNECTS) THE OBJECTIVES OF [10] AND H-LP ARE FLOORPLAN AREA AND BUS AREA.

				[9]			[10]				H-LP			
	#blk	#bus	A/M	T	A	R_{max}	T	A	BA	R_{max}	T	A	BA	R_{max}
apte	9	5	2.60/3	77	49.73	100	13	48.14	3.58	100	65	48.37	3.11	100
apte	9	36	2.47/4	3533	58.60	100	143	48.77	1.35	100	108	48.66	0.76	100
xerox	10	6	2.50/3	90	20.42	100	15	21.31	1.86	100	64	21.75	1.11	100
xerox	10	98	2.31/4	-	-	-	246	21.54	2.39	100	631	21.23	1.40	100
hp	11	14	2.29/3	213	9.38	100	38	11.83	1.07	100	66	13.20	0.84	100
hp	11	37	3.49/6	-	-	-	268	14.05	0.10	100	188	13.45	0.09	100
ami33-1	33	8	4.13/6	353	1.29	100	289	1.32	0.10	100	285	1.28	0.08	100
ami33-2	33	18	2.39/4	529	1.33	100	220	1.28	0.09	100	278	1.35	0.08	100
ami33	33	53	2.30/7	5128	1.45	100	272	1.31	0.42	100	289	1.32	0.30	100
ami49-1	49	9	4.00/6	560	38.87	100	412	40.08	12.80	100	570	40.82	11.94	100
ami49-2	49	12	3.58/6	1664	38.10	100	493	40.72	14.04	100	564	40.98	11.77	100
ami49-3	49	15	3.53/6	1564	41.12	100	614	41.29	17.77	100	552	43.19	15.38	100
ami49	49	156	3.62/7	-	-	-	2599	43.72	10.39	100	870	43.14	8.37	100
$\mu arch$	20	51	3.20/9	6942	45.66	86.27	1583	45.05	13.77	100	760	41.83	12.72	100

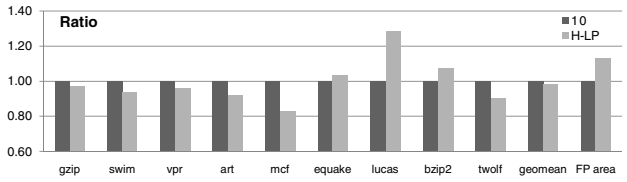

Fig. 13. IPC Comparison of Power-driven routing (scaled to [10]).

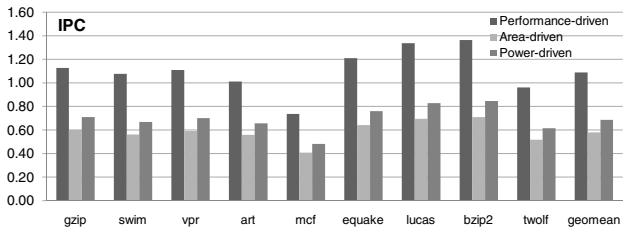

Fig. 14. IPC Comparison of H-LP for three objectives.

different applications. We presented results and applications for various microarchitectural goals. The H-LP algorithm is superior to other previous bus routing algorithms in terms of execution time, floorplan area, bus area, and routability. Experimental results show that this algorithm is scalable and efficient for today's highly complex designs. 100% routability is acheived for all benchmark circuits and better optimization results show superiority and applicability of H-LP.

TABLE V

FLOORPLANNING & ROUTING RESULTS OF H-LP BASED ON VARIOUS OBJECTIVES (SCALED TO BOLDFACE).

	Performance-driven	Area-driven	Power-driven
Floorplan Area	1.13	**1.00**	1.14
Bus Area	1.32	1.05	**1.00**
Power	1.51	1.25	**1.00**

REFERENCES

[1] F. Mo and R. Brayton, "Semi-Detailed Bus Routing with Variation Reduction," in *Proc. Int. Symp. on Physical Design*, 2007.

[2] C. Y. Lee, "An Algorithm For Path Connections And Its Applications," *IRE Trans. on Electronic Computers*, 1961.

[3] J. Soukup, "Fast Maze Router," in *Proc. ACM Design Automation Conf.*, 1978.

[4] J. Heisterman and T. Lengauer, "The efficient solution of integer programs for hierarchical global routing," *IEEE Trans. on Computer-Aided Design*, 1991.

[5] R. T. Hadsell and P. H. Madden, "Improved Global Routing through Congestion Estimation," *Proc. ACM Design Automation Conf.*, 2003.

[6] M. Cho, K. Lu, K. Yuan, and D. Z. Pan, "BoxRouter 2.0:Architecture and Implementation of a Hybrid and Robust Global Router," *Proc. IEEE Int. Conf. on Computer-Aided Design*, 2007.

[7] F. Rafiq, M. Chrzanowska-Jeske, H. Yang, M. Jeske, and N. Sherwani, "Integrated Floorplanning With Buffer/Channel Insertion for Bus-Based Designs," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 2003.

[8] H. Xiang, X. Tang, and M. Wong, "Bus-Driven Floorplanning," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 2004.

[9] J. Law and E. Young, "Multi-Bend Bus Driven Floorplanning," in *Proc. Int. Symp. on Physical Design*, 2005.

[10] D. H. Kim and S. K. Lim, "Bus-Aware Microarchitectural Floorplanning," in *Proc. Asia and South Pacific Design Automation Conf.*, 2008.

[11] M. Hanan, "On Steiner's Problem with Rectilinear Distance," *SIAM Journal on Applied Mathematics*, 1966.

[12] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "Rectangle packing based module placement," in *Proc. IEEE Int. Conf. on Computer-Aided Design*, 1995, pp. 472–479.

[13] M. Ekpanyapong, J. Minz, T. Watewai, H.-H. Lee, and S. K. Lim, "Profile-guided microarchitectural floorplanning for deep submicron processor design," in *Proc. ACM Design Automation Conf.*, 2004.

[14] C. Chiang, M. Sarrafzadeh, and C. K. Wong, "Global Routing Based on Steiner Min-Max Trees," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 1990.

[15] E. U. of Technology, "LP_solve," ftp://ftp.es.ele.tue.nl/pub/lp_solve/.

[16] T. M. Austin, "Simplescalar tool suite," http://www.simplescalar.com.

[17] S. P. E. Corporation, "Spec2000," http://www.spec.org.

[18] H. Schafft, "Thermal Analysis of Electromigration Test Structures," *IEEE Trans. on Electron Devices*, 1987.

[19] M. P. K. Banerjee and A. Ajami, "Analysis and Optimization of Thermal Issues in High-Performance VLSI," in *Proc. Int. Symp. on Physical Design*, 2001.