Timing-Driven Routing for FPGAs Based on Lagrangian Relaxation [*]

Seokjin Lee
Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, TX 78712
seokjin@cs.utexas.edu

D. F. Wong
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712
wong@cs.utexas.edu

ABSTRACT

As interconnection delay plays an important role in determining circuit performance in FPGAs, timing-driven FPGA routing has received much attention recently. In this paper, we present a new timing-driven routing algorithm for FPGAs. The algorithm finds a routing with minimum critical path delay for a given placed circuit using the Lagrangian relaxation technique. Lagrangian multipliers used to relax timing constraints are updated by subgradient method over iterations. Incorporated into the cost function, these multipliers guide the router to construct routing tree for each net. During routing, the exclusivity constraints on each routing resources are also taken care of to route circuits successfully. Experimental results on benchmark circuits show that our approach outperforms the state-of-the-art VPR router.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids—*Placement and routing*; J.6 [**Computer Applications**]: Computer-Aided Engineering—*computer-aided design(CAD)*

General Terms

Algorithms, Experimentation

Keywords

FPGA, timing-driven routing, Lagrangian relaxation

1. INTRODUCTION

Field Programmable Gate Arrays (FPGAs) have become very popular for rapid system prototyping, logic emulation and reconfigurable computing because of their low manufacturing cost and time. The problem of routing FPGAs can be considered as that of finding routing resourses to be assigned to signals. While meeting overall timing constraints, the router needs to assign all signals to routing resourses successfully.

Because of limited amount of the routing resourses, a key issue in routing of FPGAs has been to distribute the connections among the routing channels so that the maximum channel density is minimized. Several approaches [4, 10, 15] focused mainly on minimizing the use of resources have been proposed. As interconnection delay plays an important role in determining circuit performance, timing-driven routing has received much attention recently. Various algorithms [5, 8, 11, 12, 13, 17] considering timing constraints have been proposed.

In timing-driven routing problems, the timing constraints are specified by the delays from the primary inputs to the primary outputs [9], so there is a designated delay bound for each path from the primary input to the primary output. For a directed path, the slack is defined as the difference between the required times and actual propagation times along that path. In most of timing-driven routing algorithms, slacks of paths are calculated to get delay bounds on paths. Slacks are distributed to each net according to weight functions in [8], the ratio of actual delays to slacks were used in heuristics of [11]. PathFinder algorithm [12] seeks balance between eliminating congestion and minimizing delay of critical paths using the slack ratio which is defined as the ratio of the longest path containing a net to the critical path delay. The VPR is a well known FPGA placement and routing system [13]. The VPR router, based

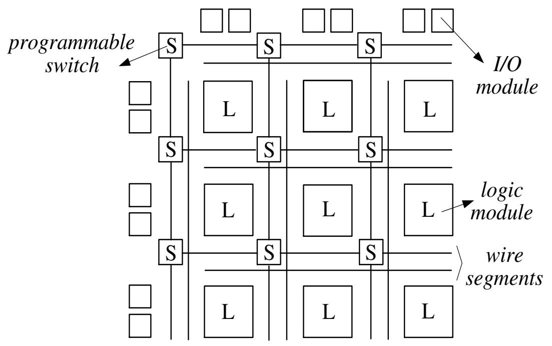

Figure 1: A typical FPGA architecture.

[*]This work was partially suppoerted by the National Science Foundation under grant CCR-9912390

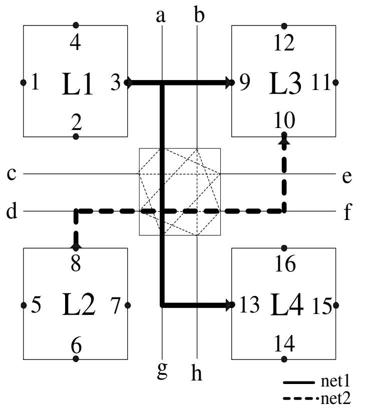

Figure 2: An example of routing in FPGA with 2 tracks(4 wire sements) per channel and 1 programmable switch module. In each logic module, the left/bottom pins are input pins, and the right/top pins are output pins.

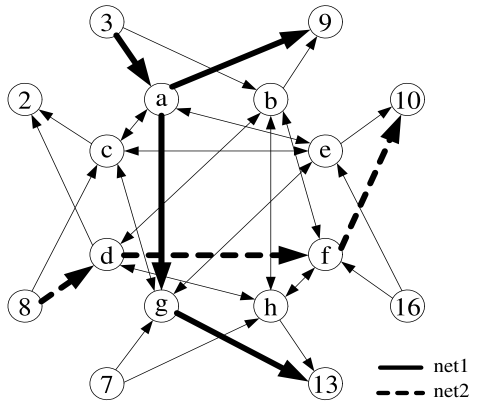

Figure 3: A routing graph and the routing of two nets corresponding to the example of Figure 2.

on a careful implementation of the PathFinder algorithm, is known to be the best routing tool to date.

In this paper, we present an effective timing-driven routing algorithm for FPGAs. Our algorithm solve the problem of minimizing delay of critical paths subject to arrival time constraints. In our approach, the timing constaints are handled in a mathematical programming framework based on the Lagrangian relaxation. The Lagrangian relaxation approach transforms the routing problem into a sequence of subproblems called the Lagrangian subproblems. Each subproblem can be greatly simplified by exploiting the network topology [6]. At each iteration of our algorithm, change in delay of each source-sink pair of a net is reflected on the value of its corresponding Lagrangian multiplier. Incorporated into the cost function, these multipliers guide the router. We conduct experiments on MCNC benchmark circuits, and demonstrate the performance of our approach by comparison with VPR [13].

The rest of this paper is organized as follows. In Section 2, we describe the FGPA routing problem. In Section 3, we formulate the timing-driven FPGA routing problem. Our timing-driven routing algorithm is presented in Section 4. We present experimental results in Section 5 and conclude the paper in Section 6.

2. THE FPGA ROUTING PROBLEM

As shown in Figure 1, a typical FPGA consists of three major components: logic modules, routing resources, and input/ouput(I/O) modules. The logic modules contain combinational and sequential circuits and implement logic functions. The routing resources consist of prefabricated wire segments and programmable switches. Routing of a FPGA is performed by programming the switches to connect the wire segments. Due to their high RC delays and large area, routability of switch modules is usually limited. Different from the interconnection tracks in custom ICs, a wire segment in an FPGA cannot be shared by different nets. This constraint on routing resources is called exclusivity constraint. Together with performance constraints, these features make FPGA routing a very challenging problem.

A net in a circuit is usually routed by connecting several wire segments with switches, and the problem of routing FPGAs can be considered to be that of assigning nets to routing resources to route all nets successfully while satisfying overall performance constraints [12]. An example of FPGA routing is shown in Figure 2. In this example, net1 is a net connecting pin3, pin9, and pin13, and it is routed by connecting segments a and g with these pins. Similarly, net2 is routed by connecting pin8, pin10, segment d, and segment f. In other words, net1 is assigned to segments a, g, and switches connecting them, and net2 is assigned to segment d, f, and switches connecting them. Suppose there is another net in our example, and it connects pin7 to pin2. The shortest routed path can be achieved by connecting segment g or h with segment c or d. But, segment g and d are already used to route other nets, and segment h and segment c cannot be connected because there is no switch between these segments in the switch module. Net3 needs to be routed by connecting more than 2 segments. If net3 belongs to a critical path of the circuit, this routing can degrade the performance of the circuit. This problem can be solved if we assign net1 to segment b and h, instead of segment a and g. In that case, segment g will be available to use and there is a switch that connects this segment with segment c.

The routing architecture of an FPGA can be modeled with a routing graph $G_r(V_r, E_r)$, which is a directed graph. The set of vertices V_r represents the input pins and the output pins of logic modules, and the wire segments. The set of edges E_r represents the feasible connections between the nodes. A route of a net in an FPGA corresponds to a subtree in G_r. This subtree is called a routing tree for the net. The root of the routing tree is the source of the net, and all the leaf nodes are the sinks of the net. Because no resource can be shared by different nets, the routing trees for the nets are vertex disjoint. Figure 3 shows G_r of the FPGA shown in Figure 2. It shows the routing trees of net1 and net2.

Given a routing graph and a netlist, the FPGA routing problem is to find vertex disjoint routing trees in G_r for all the nets while satisfying performance constraints.

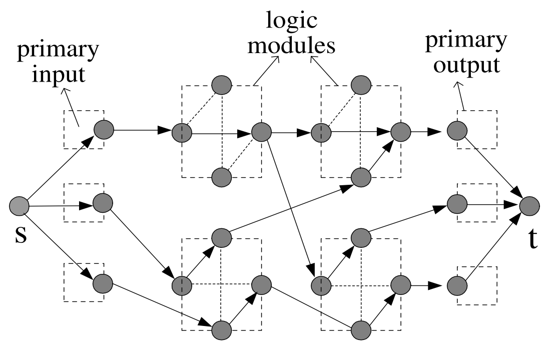

Figure 4: Timing graph of a placed netlist.

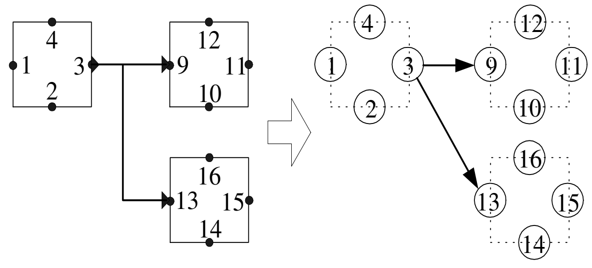

Figure 5: A multiple fanout net and its corresponding edges in timing graph.

3. TIMING-DRIVEN FPGA ROUTING

In this section, we consider the timing-driven FPGA routing problem. As in VPR [13], we use Elmore delay [7] to model components in FPGAs for the purpose of delay calculation. The source-to-sink delay of a wire-switch chain along the routing resources can be calculated from RC values specified by the architecture of FPGAs. Delay through an input-output pair in a logic module can be calculated by the architecture specific values of input driver capacitances, output resistances, and delay from input pin to output pin. Delays through input/output modules can be obtained similarly.

In timing-driven routing, the timing constraints on a circuit are specified as the arrival times at the primary inputs or outputs of storage elements, and the required times at the primary outputs or inputs of storage elements [9]. But, especially for a large circuit, the number of possible signal paths from primary inputs to primary outputs can be exponential in number of nets. By partitioning the constraints on delays along paths into constraints on delay of each source-sink pair, we can handle this difficulty. For a given placed netlist with a set of inputs and outputs, our goal is to route all nets such that the delay of critical paths is minimized while delay constraints and exclusivity constraints are satisfied.

To perform timing analysis for the timing-driven routing, we construct a timing graph $G_t(V_t, E_t)$ which is a directed acyclic graph from the input netlist. As shown in Figure 4, the vertices of timing graph correspond to primary inputs, primary outputs, and inputs and outputs of logic modules. The edges of the timing graph correspond to the source-sink pairs of each net or input-output pairs of logic modules. Note that an edge in G_t is different from a net in the netlist. Because we decompose timing constraints along the paths into those on source-to-sink delays, each source-sink pair corresponds to an edge in G_t even for the net with multiple fanouts. Figure 5 shows an example. In this example, source-sink pair (pin3, pin9), and (pin3, pin12) belong to different paths. For consistency in our notations, two fictitious node s and t are introduced. Node s is connected to all the primary inputs, and all the primary outputs are connected to node t.

Let E_S be the subset of E_t connected to node s, and E_T be the subset of E_t connected to node t. Let E_M be all the other edges. The arrival time at node u is denoted by a_u. Let D_{uv} be the delay along the edge (u,v). For an edge (u,v), D_{uv} represents the routing delay of source-sink pair (u,v) of a net or the delay between input-output pair (u,v) in a logic module. Let T_k be the routing tree for net k. For a source-sink pair (u,v) of net k, D_{uv} can be expressed in terms of resistances and capacitances of the routing resources along the path as follows:

$$D_{uv} = \sum_{i \in path(u,v)} d_i \tag{1}$$

where $path(u,v)$ is the set of nodes along the path from source u to sink v in T_k, and d_i denotes the delay contribution of node i to the delay along the path (u,v). D_{sv} denotes the arrival time of each primary input. Then the problem of routing with minimimum critical path delay under timing and exclusivity constraints is to find the vertex disjoint routing trees $\boldsymbol{T} = \{T_1, T_2, \ldots, T_n\}$ for all the nets such that

$$\begin{array}{lll} \text{Minimize} & a_t & \\ \text{Subject to} & & \\ & a_u \le a_t & \forall (u,t) \in E_T \\ & a_u + D_{uv} \le a_v & \forall (u,v) \in E_M \\ & D_{sv} \le a_v & \forall (s,v) \in E_S \end{array}$$

4. ALGORITHM DESCRIPTION

In this section, we solve the problem of minimizing the critical path delay under timing and exclusivity constraints using Lagrangian relaxation. Lagrangian relaxation is a general technique for solving optimization problems with difficult constraints. In Lagrangian relaxation, constraints are relaxed and added to the objective function after multiplied by constants called Lagrangian multipliers. By doing this we have a new optimization problem called the Lagrangian subproblem for each fixed vector of the Lagrangian multipliers. For a given vector of the Lagrangian multipliers, the optimal solution of a Lagrangian subproblem gives the lower bound close to the optimal objective function value of the original problem. The problem of finding such a vector is called the Lagrangian dual problem. By solving the Lagrangian subproblem with a vector obtained by solving the Lagrangian dual problem, we can obtain the lower bound close to the optimum value of the objective function of the original optimization problem. More details can be obtained in [1, 2].

In Section 4.1, we present the Lagrangian relaxation framework to solve the timing-driven routing problem. In Section

```
Algorithm LR_ROUTE
Input: Timing graph $G_t(V_t, E_t)$ and
       Routing graph $G_r(V_r, E_r)$
Output: A routed netlist
begin
1. Initialize $\boldsymbol{\lambda}$ as an arbitrary vector in $\boldsymbol{\Lambda}$
2. Call NET_ROUTE with $\boldsymbol{\lambda}$ to solve $LS'_\lambda$.
3. Compute $a_u$ for each $u \in V_t$.
       Set each $a_u$ to the smallest possible value
       that satisfies the timing constraints in a
       topological order from $s$ to $t$.
4. Update $\lambda_{uv}$ for each $(u,v) \in E_t$.
5. Project $\boldsymbol{\lambda}$ to the nearest vector in $\boldsymbol{\Lambda}$.
6. Repeat Step 2-5 until no shared resource exists.
   $(a_t - L^*) \leq$ error bound.
end
```

Figure 6: Algorithm LR_ROUTE

4.2, we show the algorithm that solves the Lagrangian subproblems by routing nets for a given vector of the Lagrangian multipliers.

4.1 Lagrangian Relaxation

We relax the timing constraints in the original problem. The exclusivity constraints are handled by the net router which solves the Lagrangian subproblem. Each of the constraints is multiplied by the corresponding Lagrangian multiplier, and added to the objective function. Let

$$\begin{aligned} L_\lambda(\boldsymbol{a}, \boldsymbol{T}) = \; & a_t + \sum_{(u,t)\in E_T} \lambda_{ut}(a_u - a_t) \\ & + \sum_{(u,v)\in E_M} \lambda_{uv}(a_u + D_{uv} - a_v) \\ & + \sum_{(s,v)\in E_S} \lambda_{sv}(D_{sv} - a_v) \end{aligned}$$

This relaxed objective function is called the Lagrangian function, and the Lagrangian subproblem associated with the fixed set of Lagrangian multipliers $\boldsymbol{\lambda}$ is

$$LS_\lambda : \quad \text{Minimize} \quad L_\lambda(\boldsymbol{a}, \boldsymbol{T})$$

Because the minimum value of $L_\lambda(\boldsymbol{a}, \boldsymbol{T})$ for any vector $\boldsymbol{\lambda}$ is a lower bound on the optimal objective function value of the original problem, the lower bound close to the optimal objective value of the original problem is obtained by solving

$$L^* = \max_{\lambda \geq 0} L_\lambda(\boldsymbol{a}, \boldsymbol{T})$$

which is known as the Lagrangian dual problem.

There are conditions on the Lagrangian multipliers $\boldsymbol{\lambda}$ corresponding to the optimal solution of the original problem, and they can be derived using the Kuhn-Tucker optimality condition [2]. This condition implies $\partial L(\lambda)/\partial a_u = 0$ $\forall u \in V_t$. By applying the Kuhn-Tucker condition to the Lagrangian function, we obtain the following optimality conditions on $\boldsymbol{\lambda}$:

$$1 = \sum_{(u,t)\in E_T} \lambda_{ut} \tag{2}$$

$$\sum_{(w,v)\in E_t} \lambda_{wv} = \sum_{(u,w)\in E_t} \lambda_{uw} \quad \forall w \in V_t - \{s,t\} \tag{3}$$

Due to unique structure of our problem, we can greatly simplify LS_λ by applying these conditions. By rearranging the terms, the Lagrangian function $L_\lambda(\boldsymbol{a}, \boldsymbol{T})$ can be written as

$$\begin{aligned} L_\lambda(\boldsymbol{a}, \boldsymbol{T}) = \; & (1 - \sum_{(u,t)\in E_T} \lambda_{ut}) a_t \\ & + (\sum_{(w,u)\in E_t} \lambda_{wv} - \sum_{(u,w)\in E_t} \lambda_{uw}) a_w \\ & + \sum_{(u,v)\in E_S \cup E_M} \lambda_{uv} D_{uv} \end{aligned}$$

When $\boldsymbol{\lambda}$ satisfies the optimality conditions, $L_\lambda(\boldsymbol{a}, \boldsymbol{T})$ is simplified to

$$L'_\lambda(\boldsymbol{T}) = \sum_{(u,v)\in E_S \cup E_M} \lambda_{uv} D_{uv}$$

Then LS_λ is simplified to

$$LS'_\lambda : \quad \text{Minimize} \quad L'_\lambda(\boldsymbol{T})$$

LS'_λ doesn't have the a_u terms and solving LS_λ is equivalent to solving LS'_λ.

To solve the Lagrangian dual problem, an iterative approach is used. At each iteration, we solve LS_λ by solving LS'_λ for a given $\boldsymbol{\lambda}$, and then update the Lagrangian multipliers for the next iteration using the solution of the current iteration. The Lagrangian multipliers for $(r+1)$th iteration are updated by the subgradient method [1, 2] as follows:

$$\begin{aligned} \lambda_{ut}^{r+1} &= \max\{0, \lambda_{ut}^r + \theta_r(a_u - a_t)\} & \forall (u,t) \in E_T \\ \lambda_{uv}^{r+1} &= \max\{0, \lambda_{uv}^r + \theta_r(a_u + D_{uv} - a_v)\} & \forall (u,v) \in E_M \\ \lambda_{sv}^{r+1} &= \max\{0, \lambda_{sv}^r + \theta_r(D_{sv} - a_v)\} & \forall (s,v) \in E_S \end{aligned}$$

where θ_r is a step size with the property that $\lim_r \theta_r \to 0$, and $\lim_r \sum \theta_r \to \infty$. Let $\boldsymbol{\Lambda}$ be nonnegative $\boldsymbol{\lambda}$ which satisfies the optimality conditions (2) and (3). Because we solve LS'_λ instead of LS_λ, updated $\boldsymbol{\lambda}$ is projected to the nearest vector in $\boldsymbol{\Lambda}$ at each iteration. Figure 6 summerizes our algorithm. With a given vector $\boldsymbol{\lambda}$, algorithm NET_ROUTE solves LS'_λ. The NET_ROUTE algorithm is presented in the following section. By solving LS'_λ with the optimal $\boldsymbol{\lambda}$ found by the algorithm, we can find the routing such that critical path delay can be minimized.

4.2 Routing Nets

In this section, we consider solving the simplified Lagrangian subproblem LS'_λ with a given vector $\boldsymbol{\lambda}$. By routing each net using an appropriate cost function, we can solve this problem. The objective function $L'_\lambda(\boldsymbol{T})$ of this problem implies that the edges in the timing graph need to be routed such that the Elmore delay weighted with the Lagrangian multipliers is minimized for a given $\boldsymbol{\lambda}$, because D_{uv} denotes the routing delay of an edge (u,v). While routing, however, exclusivity constraints also need to be satisfied so that all

```
Algorithm NET_ROUTE
Input: A placed netlist, $\boldsymbol{\lambda}$, $G_r(V_r, Er)$
Output: A routed netlist
begin
1. for each net $k$ do
2.     Rip up routing for net $k$
3.     for each sink $v$ of net $k$ do
4.         Maze route from source to sink,
               where cost at node $i$ is
                   $C_i = \lambda_{uv} d_i + \mu_i$
               for each node $i$ of $V_r$
5.         Update $p_i$ for all $i$'s
           in $path(u,v)$
end
```

Figure 7: Algorithm NET_ROUTE

the nets should be routed successfully. To handle the exclusivity constraints, a decision variable for each node in G_r is defined as follows:

$$x_{ik} = \begin{cases} 1, & \text{if the routing tree } T_k \text{ for net } k \text{ uses node } i \\ 0, & \text{otherwise} \end{cases}$$

From LS'_λ and exclusivity constraints, the net routing problem is that of constructing the routing trees $\boldsymbol{T}$ for all nets in the placed netlist for a set of given multipliers λ_{uv}'s such that

$$\begin{array}{ll} \text{Minimize} & \sum_{(u,v) \in E_S \cup E_M} \lambda_{uv} D_{uv} \\ \text{Subject to} & \\ & \sum_k x_{ik} \leq 1 \quad \forall i \in V_r \end{array}$$

This problem can be solved by Lagrangian relaxation. Let

$$\begin{aligned} L_\mu(\boldsymbol{x}) &= \sum_{(u,v) \in E_S \cup E_M} \lambda_{uv} D_{uv} + \sum_{i \in V_r} \mu_i (\sum_k x_{ik} - 1) \\ &= \sum_k \left\{ \sum_{(u,v) \in E_k} \lambda_{uv} D_{uv} + \sum_{i \in V_r} \mu_i x_{ik} \right\} - \sum_{i \in V_r} \mu_i \end{aligned}$$

where E_k is a set of source-sink pairs belonging to the routing tree T_k for net k. Note that $\sum_{i \in V_r} \mu_i$ is a constant term. NET_ROUTE algorithm constructs the routing trees for all nets that minimize

$$L'_\mu(\boldsymbol{x}) = \sum_k \left\{ \sum_{(u,v) \in E_k} \lambda_{uv} D_{uv} + \sum_{i \in V_r} \mu_i x_{ik} \right\} \quad (4)$$

Our net routing algorithm is similar to the PathFinder algorithm [12]. Given a routing graph, NET_ROUTE iteratively constructs a minimum cost routing tree for each net. It rips up one net at a time, and reroutes with updated cost. While routing a net, each source-sink pair is routed sequentially in decreasing order of λ_{uv}. Initially, nodes in routing graph are allowed to be shared by multiple nets. After each iteration, the cost of sharing resources is gradually increased, and only the nets with higher criticality try to use the nodes with higher costs.

In the placed netlist, each net can have multiple sinks, and each source-sink pair (u, v) of the net has corresponding Lagrangian multiplier λ_{uv}. Each routing resource i has corresponding Lagrangian multiplier μ_i. To achieve feasible routing that minimizes $L'_\mu(\boldsymbol{x})$, NET_ROUTE uses the Lagrangian multipliers as weights for the cost of using resources. From equation (1), $\lambda_{uv} D_{uv}$ term of (4) can be expressed as

$$\lambda_{uv} D_{uv} = \sum_{i \in path(u,v)} \lambda_{uv} d_i$$

Hence, each node i on the $path(u, v)$, its contribution to (4) is given by

$$C_i = \lambda_{uv} d_i + \mu_i$$

In this equation, the first term is delay control term, and the second term is congestion control term. Source-sink pairs belonging to more critical paths have larger λ_{uv}'s, and NET_ROUTE constructs routing tree with costs that biased more to the delay cost than to congetion cost for those pairs.

In PathFinder algorithm, the congestion-sensitive term is defined as

$$c_i = b_i * p_i$$

where b_i is a base cost for a routing resource, and p_i is a penalty term for congestion control. Each c_i can be interpreted as a Lagrangian multiplier, and it plays the same role as μ_i, but it is updated in a different way from the subgradient method. In our current implementation, we adopted multiplier c_i for congestion control, and we set $\mu_i = c_i$. Figure 7 summarizes the algorithm NET_ROUTE.

5. EXPERIMENTAL RESULTS

The proposed timing-driven routing algorithm was implemented in C on a SUN SPARC workstation. The experiments are performed on 17 large circuits from MCNC benchmark [16]. The placed netlists were generated using the placer in VPR [14]. We assumed a symmetrical-array-based FPGA [3], where each logic block contains four 4-input lookup tables and four flip-flops. We set $F_s = 3$ and $F_c = W$, where W is the number of wire segments of each channel. F_s denotes the number of connections for each wiring segment entering the switch box. F_c denotes the number of tracks to which each logic block pin can connect. For the purpose of comparison, we used identical intrinsic delay values and timing models of VPR.

We performed routing on each circuit with fixed channel width. We obtained this fixed number of tracks per channel by running VPR on timing-driven mode. The critical path delays and runtime were compared after running LR_ROUTE on each circuit with these channel width. Results are shown in Table 1. *LUTs/FFs* column shows the number of LUTs and flip-flops in each circuit. The critical path delays of the circuits routed with LR_ROUTE and VPR are shown under *delay* column. We also compared runtime for routing each circuit, and it is shown under *runtime* column. Among 17 benchmark circuits, LR_ROUTE yields better results for 13 circuits, and the critical path delays are shorter up to 33% with comparable runtime.

6. CONCLUSION

In this paper, we introduced LR_ROUTE, a new timing-driven routing algorithm for FPGAs. In our algorithm, we

Circuit	LUTs /FFs	# of Tracks	delay(ns)		runtime(s)	
			VPR	LR_ROUTE	VPR	LR_ROUTE
alu4	1522	33	46.6	46.2	58	57
apex2	1878	43	61.5	49.3	61	46
apex4	1262	41	45.4	48.9	29	41
bigkey	1707	24	41.7	27.8	53	62
clma	8383	51	125.0	96.4	531	464
des	1591	24	43.5	48.1	44	42
diffeq	1497	29	48.8	48.6	32	31
dsip	1370	25	29.6	27.6	53	78
elliptic	3604	40	77.1	71.3	151	256
ex1010	4598	44	83.5	75.2	248	351
ex5p	1064	43	44.8	43.7	22	34
frisc	3556	43	81.5	84.3	121	171
misex3	1397	37	42.5	49.4	50	49
pdc	4575	61	96.5	95.0	304	465
s298	1931	28	98.7	91.5	71	85
seq	1750	35	55.9	47.0	55	67
spla	3690	56	94.7	74.0	203	234

Table 1: Critical path delay and runtime comparison between VPR and LR_ROUTE

handled the timing constraints in a mathematical programming framework based on Lagrangian relaxation. Experimental results show that the new router outperformed the state-of-the-art VPR router.

7. REFERENCES

[1] R. K. Ahuja, T. L. Magnanti and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall, 1993.

[2] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms, 2nd ed.* New York: Wiley, 1993.

[3] S. Brown, R. Francis, J. Rose, Z. Vranesic, *Field-Programmable Gate Arrays*, Kluwer Academic Pub., 1992.

[4] S. Brown, J. Rose, Z. Vranesic, "A Detailed Router for Field-Programmable Gate Arrays," *IEEE Trans. on Computer-Aided Design*, May 1992, pp. 620-627.

[5] Yao-Wen Chang, D. F. Wong, Kai Zhu, and C. K. Wong, "On a New Timing-Driven Tree Problem," in *Proc. Intl. Conf. on Computer-Aided Design*, 1994, pp. 380-385.

[6] C.-P. Chen, C. C. N. Chu, D. F. Wong, "Fast and Exact Simultaneous Gate and Wire Sizing by Lagrangian Relaxation," in *Proc. Intl. Conf. on Computer- Aided Design*, 1997, pp. 614-621.

[7] W. C. Elmore, "The transient response of damped linear network with particular regard to wideband amplifiers,"*J. Appl. Phys.*, vol. 19, 1948, pp. 55-63.

[8] J. Frankle, "Iterative and Adaptive Slack Allocation for Performance-driven Layout and FPGA Routing," in *Proc. ACM/IEEE Design Automation Conference*, 1992, pp 536-542.

[9] R. B. Hitchcock, Sr., G. L. Smith, D. D. Cheng, "Timing Analysis of Computer Hardware," *IBM J. Research and Development*, vol. 26, No.1, Jan. 1982, pp. 100-105.

[10] J. S. Swartz, V. Betz, J. Rose, "A Fast Routability-Driven Router for FPGAs," *Proc. ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, 1998, pp. 140-149.

[11] Y.-S. Lee, C.-H. Wu, "A performance and routability driven router for FPGA's considering path delay," in *Proc. Design Automation Conference*, 1995, pp. 557-561.

[12] L. McMurchie, C. Eberling, "PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs," in *Proc. ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, 1995, pp. 111-117.

[13] V. Betz, J. Rose, "VPR: A New Packing, Placement and Routing Tool for FPGA Research," in *Proc. the 7th Annual Workshop on Field Programmable Logic and Applications*, 1999, pp. 213-222.

[14] V. Betz, *VPR and T-VPack User's Manual*, University of Toronto, 2000

[15] Y.-L. Wu, M. Marek-Sadowska, "Graph Based Analysis of FPGA Routing," In *Proc. of Euro-DAC*, 1994, pp. 104-109.

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

[17] K. Zhu, Y.-W. Chang, D. F. Wong, "Timing-Driven Routing for Symmetrical-Array-Based FPGAs," in *Proc. Intl. Conf. on Computer Design*, 1998, pp. 628-633.