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 resource 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 resources to be assigned to signals. While meeting overall timing constraints, the router needs to assign all signals to routing resources successfully.

Because of limited amount of the routing resources, 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
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 constants 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 FPGA 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/output (I/O) modules. The logic modules consist of 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 pins 3, pin9, and pin13, and it is routed by connecting segments a and g with these pins. Similarly, net2 is routed by connecting pins 8, 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逻辑 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 sub-tree in \( G_r \). This sub-tree 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.
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 \((\text{pin}3, \text{pin}9)\), and \((\text{pin}3, \text{pin}12)\) 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:

\[
D_{uv} = \sum_{i \in \text{path}(u, v)} d_i
\]

where \( \text{path}(u, v) \) 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 \((u, v)\). \( D_{uv} \) denotes the arrival time of each primary input. Then the problem of routing with minimum critical path delay under timing and exclusivity constraints is to find the vertex disjoint routing trees \( T = \{T_1, T_2, \ldots, T_n\} \) for all the nets such that

Minimize \( a_t \)

Subject to

\( a_u \leq a_t \) \quad \forall(u, t) \in E_T
\]

\( a_u + D_{uv} \leq a_v \) \quad \forall(u, v) \in E_M

\( D_{uv} \leq a_v \) \quad \forall(s, v) \in E_S

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 optimal 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
4.2 Routing Nets

In this section, we consider solving the simplified Lagrangian subproblem $LS^*_L$ with a given vector $\lambda$. By routing each net using an appropriate cost function, we can solve this problem. The objective function $L'_{\lambda}(T)$ 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 $\lambda$, 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
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 $LS_1$ and exclusivity constraints, the net routing problem is that of constructing the routing trees $T$ for all nets in the placed netlist for a set of given multipliers $\lambda_{uv}$’s such that

$$ \text{Minimize} \quad \sum_{(u,v) \in E_S \cup E_M} \lambda_{uv} D_{uv} $$

$$ \text{Subject to} \quad \sum_k x_{ik} \leq 1 \quad \forall i \in V_r $$

This problem can be solved by Lagrangian relaxation. Let

$$ L_\mu(x) = \sum_{(u,v) \in E_S \cup E_M} \lambda_{uv} D_{uv} + \sum_i \mu_i \left( \sum_k x_{ik} - 1 \right) $$

$$ = \sum_k \left\{ \sum_{(u,v) \in E_k} \lambda_{uv} D_{uv} + \sum_i \mu_i x_{ik} \right\} - \sum_i \mu_i $$

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

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

Figure 7: Algorithm NET_ROUTE

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 tops 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 $\lambda_{uv}$. 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 $\lambda_{uv}$. Each routing resource $i$ has corresponding Lagrangian multiplier $\mu_i$. To achieve feasible routing that minimizes $L'_\mu(x)$, 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 $h(u,v)$, 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 $\lambda_{uv}$’s, and NET_ROUTE constructs routing tree with costs that biased more to the delay cost than to congestion 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 $\mu_i$, 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 $F_c = 3$ and $F_r = W$, where $W$ is the number of wire segments of each channel. $F_r$ 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 LRROUTE 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 LRROUTE 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, LRROUTE 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 LRROUTE, a new timing-driven routing algorithm for FPGAs. In our algorithm, we
Table 1: Critical path delay and runtime comparison between VPR and LR_ROUTE

<table>
<thead>
<tr>
<th>Circuit</th>
<th>LUTs /FFs</th>
<th># of Tracks</th>
<th>delay(ns)</th>
<th>runtime(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>alu4</td>
<td>1522</td>
<td>33</td>
<td>46.6</td>
<td>46.2</td>
</tr>
<tr>
<td>apex2</td>
<td>1878</td>
<td>43</td>
<td>61.5</td>
<td>49.3</td>
</tr>
<tr>
<td>apex4</td>
<td>1262</td>
<td>41</td>
<td>45.4</td>
<td>48.9</td>
</tr>
<tr>
<td>bigkey</td>
<td>1707</td>
<td>24</td>
<td>41.7</td>
<td>27.8</td>
</tr>
<tr>
<td>clma</td>
<td>8383</td>
<td>51</td>
<td>125.0</td>
<td>96.4</td>
</tr>
<tr>
<td>des</td>
<td>1591</td>
<td>24</td>
<td>43.5</td>
<td>48.1</td>
</tr>
<tr>
<td>diffeq</td>
<td>1497</td>
<td>29</td>
<td>48.8</td>
<td>48.6</td>
</tr>
<tr>
<td>dsp</td>
<td>1370</td>
<td>25</td>
<td>29.6</td>
<td>27.6</td>
</tr>
<tr>
<td>elliptic</td>
<td>3604</td>
<td>40</td>
<td>77.1</td>
<td>71.3</td>
</tr>
<tr>
<td>ex1010</td>
<td>4598</td>
<td>44</td>
<td>83.5</td>
<td>75.2</td>
</tr>
<tr>
<td>ex5p</td>
<td>1064</td>
<td>43</td>
<td>44.8</td>
<td>43.7</td>
</tr>
<tr>
<td>frisc</td>
<td>3556</td>
<td>43</td>
<td>81.5</td>
<td>84.3</td>
</tr>
<tr>
<td>misex3</td>
<td>1397</td>
<td>37</td>
<td>42.5</td>
<td>49.4</td>
</tr>
<tr>
<td>pdc</td>
<td>4575</td>
<td>61</td>
<td>96.5</td>
<td>95.0</td>
</tr>
<tr>
<td>s298</td>
<td>1931</td>
<td>28</td>
<td>98.7</td>
<td>91.5</td>
</tr>
<tr>
<td>seq</td>
<td>1750</td>
<td>35</td>
<td>55.9</td>
<td>47.0</td>
</tr>
<tr>
<td>spla</td>
<td>3690</td>
<td>56</td>
<td>94.7</td>
<td>74.0</td>
</tr>
</tbody>
</table>

7. REFERENCES


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