A Min-Cost Flow Based Detailed Router for FPGAs [*]

Seokjin Lee
Dept. of ECE
The University of Texas at Austin
Austin, TX 78712

Yongseok Cheon
Dept. of Computer Sciences
The University of Texas at Austin
Austin, TX 78712

Martin D. F. Wong
Dept. of ECE
University of Illinois at Urbana-Champaign
Urbana, IL 61801

ABSTRACT

Routing for FPGAs has been a very challenging problem due to the limitation of routing resources. Although the FPGA routing problem has been researched extensively, most algorithms route one net at a time, and it can cause the net-ordering problem.

In this paper, we present a detailed routing algorithm for FPGAs based on min-cost flow computations. Using the min-cost flow approach, our algorithm routes all the nets connected to a common logic module simultaneously. At each stage of the network flow computation, we guarantee optimal result in terms of routability and delay cost. For further improvement, we adopt an iterative refinement scheme based on the Lagrangian relaxation technique. The Lagrangian relaxation approach transforms the routing problem into a sequence of Lagrangian subproblems. At each iteration of the algorithm, Lagrangian subproblems are solved by our min-cost flow based routing algorithm. Any violation of congestion constraints is reflected in the value of corresponding Lagrangian multiplier. The Lagrangian multipliers are incorporated into the cost of each routing rosource node and guide the router.

Because our min-cost flow based algorithm minimizes cost function while it maximizes the flow, our algorithm finds congestion-free routing solutions with minimum total delay. Comparison with VPR router shows that our router uses less or equal number of routing tracks with smaller critical path delay as well as total routing delay.

Keywords

FPGA routing, min-cost flow algorithm, Lagrangian relaxation

1. INTRODUCTION

Due to their low manufacturing cost and time, field programmable logic arrays (FPGAs) have been very popular for rapid system prototyping, logic emulation, and reconfigurable computing. Figure 1 shows a typical array-based FPGA architecture. An array-based FPGA consists of a two-dimensional array of logic modules, vertical and horizontal routing channels, and switch modules. The logic modules contain combinational and sequential circuits to realize logic functions. The routing channels and the switch modules comprise the routing resources of an FPGA. The

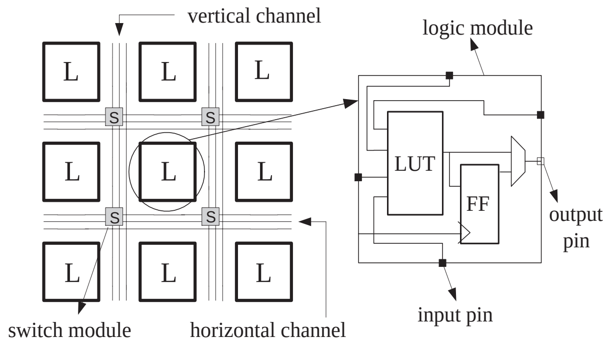

Figure 1: Typical FPGA architecture

routing channels usually have various lengths of wire segments to improve circuit performance and maintain reasonable routability at the same time. Routing of an FPGA is performed by programming the switches to connect the wire segments. Due to their high RC delays and large area, the routability of switch modules are usually limited.

It is widely known that the feasiblity of FPGA design is most constrained by routing resources, and routing delays dominate the performance of FPGAs. The routing problem of the array-based FPGAs has been extensively studied by researchers[4, 6, 9, 12, 13, 14, 16, 18]. Most FPGA routers route one net at a time, and they suffer from net-ordering problem in which routing results may vary significantly depending on the ordering of the nets to route. The PathFinder algorithm[14] alleviates this problem by well-designed rip-up and reroute scheme. The VPR router[16], based on a careful implementation of the PathFinder algorithm, is a very successful placement and routing tool.

In this paper, we present an effective routability driven detailed routing algorithm, which also considers routing delay. Our algorithm first considers the problem of routing all the nets connected to one common logic module through logically equivalent input pins. We assume the LUT-based logic modules for our target architecture. Our algorithm exploits the fact that the input pins of a LUT are logically equivalent, and it routes all the nets connected to one LUT simultaneously. Our algorithm is based on min-cost flow computations, and it guarantees to find a congestion-free wire type assignment solution if one exists. Furthermore, it can find a solution with minimum total delay at the same time. Although network flow frameworks have been used for various

[*]This work was partially supported by the National Science Foundation under grant CCR-0244236

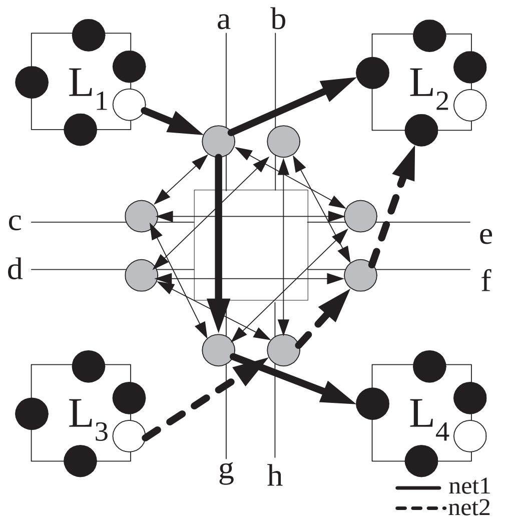

Figure 2: A routing graph and the routing trees for two nets. Black nodes represent input pin nodes, white nodes correspond to output pin nodes, and the gray nodes correspond to the wire segments. Edge pairs between the nodes are represented by the bidirectional edges for simplicity.

routing algorithms[3, 7, 11, 15], most of those algorithms need to solve the multicommodity flow problems which cannot guarantee integer flow solutions. Although [15] proposed an algorithm based on min-cost flow algorithm, it can only handle the nets connected to a common node.

To alleviate the possible ordering problem in LUT selection, and to further improve the routing results, we adopt an iterative refinement scheme based on Lagrangian relaxation. In the Lagrangian relaxation framework, the routing problem is transformed into a sequence of subproblems called the Lagrangian subproblems. Each subproblem corresponds to the routing problem for all the nets connected to a LUT, and it is solved by min-cost flow computation. At each iteration of our algorithm, violations in congestion constraints are reflected in the value of corresponding Lagrangian multipliers, and the multipliers guide the router.

The rest of the paper is organized as follows. The FPGA detailed routing problem is defined in section 2. In section 3, we present the flow network graph construction scheme and our network flow based algorithm for routing nets together with the Lagrangian relaxation framework for iterative refinement. Experimental results are shown in section 4, and we conclude the paper in section 5.

2. PROBLEM FORMULATION

Routing of an FPGA is performed by programming the switches to connect the wire segments. Unlike the interconnection tracks in custom ICs, a wire segment in an FPGA cannot be shared by different nets. Together with performance constraints, this congestion constraints make FPGA routing a very challenging problem. The problem of routing FPGAs is assigning nets to routing resources to route all nets successfully. The routing architecture of an FPGA can be modeled with a routing graph $G(V, E)$. A set of nodes V in this directed graph represents the input pins and output pins of logic modules, and the wire segments. The set of edges E corresponds to feasible connections between the routing resources represented by nodes. We can attribute a set of capacity R and a set of cost C to each node and edge in G. A capacity of a node (or an edge) denotes the available number of the pin or wire segment (switch), and a cost of a node (an edge) represents the routing delay through the corresponding routing resource. For detailed routing, capacities for all the routing resources can be set to 1. A route of a net corresponds to a subtree in G. The root of a 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 2 shows an example of a routing graph and routing trees. Nodes and edges are superimposed on the corresponding routing resources. The problem addressed in this paper is stated below:

FPGA detailed routing problem: *Given a routing resource graph G for an FPGA architecture, find vertex disjoint routing trees in G for all the nets.*

To alleviate the net ordering problem, we route several nets connected to a common logic module simultaneously rather than routing nets one by one. As shown in Figure 1, we assume a widely used model of a logic module, which consists of several look-up tables (LUTs) and flip-flops. In this paper, we assume that a logic module has one LUT in it for the simplicity of presentation unless stated otherwise. Because all the inputs of a LUT are logically equivalent, they are permutable when they are assigned to the routes for the nets connected to the LUT. In our routing algorithm, we route all the nets connected to a LUT simultaneously. For multiple-pin nets, we route a portion of net (net segment) branched from a partial routing tree previously constructed for the net. In the example shown in Figure 2, both of net1 and net2 are connected to a logic module L_2. Net1 is a multiple-pin net. Suppose the routing tree for net1 is already constructed partially (from L_1 to L_4 using wire segments a and g) when the logic module L_4 is processed. Then, only the portion of the tree branching from this partially constructed tree needs to be routed when the module L_2 is handled. Before stating the overall routing problem, we define a smaller problem as follows:

The Routing for One LUT (ROL) Problem: *Given a routing resource graph $G(V, E)$ and a LUT, find routes for all the net segments connected to the LUT such that each edge and node is used at most once.*

Because we will use min-cost flow computation to solve ROL problem, we can also minimize the total cost of the routes for the nets. By solving the ROL problem for each LUT, we can solve the overall routing problem for an FPGA. As ROL problem for each LUT is solved, branches of routings trees for the nets are gradually constructed, and after solving ROL for all the LUTs, we can get the routing trees for all the nets in an FPGA. In the example of Figure 2, although only a portion of net1 is routed when L_4 is processed, by the time when routing for L_2 is finished, routing trees for both of net1 and net2 are fully constructed. The FPGA detailed routing problem can be formulated as:

$$\begin{array}{ll} \text{Minimize} & \sum_{i,k} c_i x_{ik} \\ \text{subject to} & \\ & \sum_k x_{ik} \le 1 \quad \forall i \in V, and \\ & \boldsymbol{x} \in \boldsymbol{X} \end{array}$$

where x_{ik} is a decision variable for each node (or edge) defined as

$$x_{ik} = \begin{cases} 1, & \text{if the routing of net } k \text{ uses node (or edge) } i \\ 0, & \text{otherwise} \end{cases}$$

c_i is the cost of node (or edge) i, $\boldsymbol{x}$ is the vector of x_{ik}, and $\boldsymbol{X}$ is the set of all possible routes of each net.

3. ALGORITHM DESCRIPTION

In this section, we describe the algorithms to solve the problems introduced in the previous section. By performing the min-cost max flow computations on the flow network which is constructed from G, our algorithm for the ROL problem, ROL_NF, can solve the ROL problem in polynomial time. Because each routing resource can be accessed by several nets, there can be dependency between the flows computed for each LUT. To avoid this ordering problem of LUTs processed by ROL_NF, we adopted an iterative refinement scheme based on 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 being multiplied by coefficients called Lagrangian multipliers. In our scheme, the congestion constraints are relaxed, and the Lagrangian multipliers used to relax the congestion constraints are added to the cost for each node to guide the router to find less congested routes for the nets.

3.1 Min-cost flow algorithm to route nets for one LUT

To solve the ROL problem for a LUT, we construct the flow network from the routing graph. We will apply a min-cost flow algorithm on the constructed flow network. Let $L = \{l_1, l_2, ..., l_m\}$ be a set of all LUTs in an FPGA. Given a routing graph $G(V, E)$ with capacity R and cost C and a LUT l_k, we construct the flow network $G_f(V_f, E_f)$ as follows:

1. $V_f = V \cup \{s, s_1, s_2, ..., s_n, t\}$, where s is a source node, and s_i is a subsource node, and t is a sink node of $G_f(V_f, E_f)$. n is the number of nets connected to l_k.

2. $E_f = E \cup E_s \cup E_{s'} \cup E_t$, where $E_s = \{(s, s_i)|i = 1, 2, ..., n\}$, $E_{s'} = \{(s_i, v)|i = 1, 2, ..., n, v \in T_i\}$, $E_t = \{(p_i, t)|p_i \in S_p\}$. T_i denotes a partially constructed routing tree for net i, and S_p is a set of nodes correspond to the input pins of l_k.

3. Edge Capacity: $r_f(e) = 1 \quad \forall e \in E_f$

4. Node Capacity:
$r_f(v) = 1 \quad \forall v \in V \cup \{s_1, s_2, ..., s_n\}$, node s and node t are incapacitated.

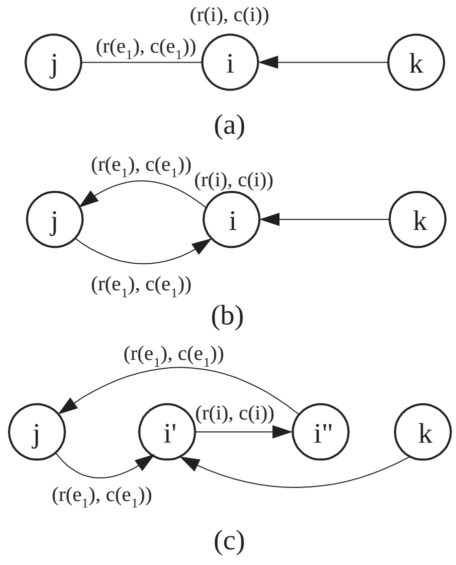

Figure 3: Network transformations: (a) Original network. Note that the edge between node i and node j (e_1) is an undirected edge. (b) Transformation to directed edges. (c) Node splitting. Node i is split to i' and i''. A capacity and a cost are assigned to the edge (i', i''). All the incoming edges to node i are connected to i', and an outgoing edge is replaced by an edge going out of node i''. (Node splitting for node j and node k are omitted.)

5. Edge Cost:

$$c_f(e) = \begin{cases} c(e), & \text{if } e \in E \\ 0, & otherwise \end{cases}$$

6. Node Cost:

$$c_f(v) = \begin{cases} c(v), & \text{if } v \in V \\ 0, & \text{if } v \in s, s_1, s2, ..., s3, t \end{cases}$$

The constructed flow network for a ROL problem is illustrated in Figure 4. Note that the subsource nodes and the edge set $E_{s'}$ are adaptively updated for each l_k depending on the number of nets connected to l_k and the partial routing tree of each net constructed in the process of solving ROL problems for other LUTs. By connecting a subsource node to all the nodes belonging to a partial routing tree for a net, we can find the best branching point for the Steiner tree. To make our problem conform to the classical network flow framework, we transformed $G_f(V_f, E_f)$ to a directed graph in which only edges have capacities and costs. Note that any undirected edges, which can be formed due to bidirectional wire segments, in $G_f(V_f, E_f)$ can be transformed to a pair of directed edges with the cost and the capacity of the original undirected edge [2]. By node splitting transformation, any node i with nonzero cost and capacity is transformed into the two nodes i' and i''. This transformation replaces each of the original edges (j, i) and (i, k) into (j, i') and (i'', k), respectively. It also adds an edge (i', i'') with the cost and the capacity of node i. Figure 3 shows an example of the network transformations.

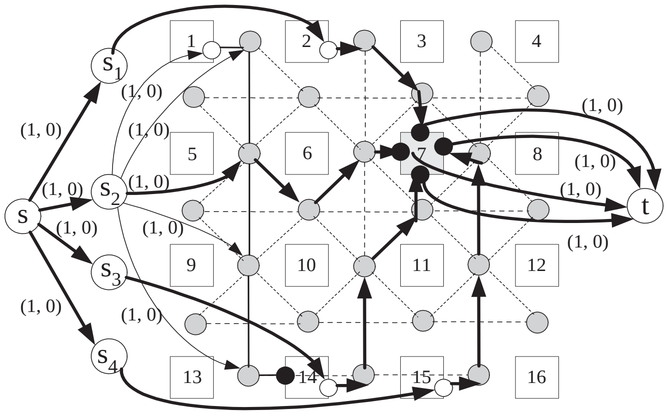

Figure 4: A flow corresponds to ROL for 4 net segments connected to l_7.

It can be shown that any flow in G_f is a routing solution for a subset of the given nets to route. Each flow from s to t through subsources corresponds to a route for a net segment connected to l_k. A node occupied by a flow corresponds to a wire segment used for the route, and the flow along an edge denotes that the corresponding switch is used for the route. If a flow f exists and $|f| = n$, then we can find a feasible solution for all the net segments connected to the LUT, and the cost of the flow is the cost of a solution to the ROL problem. Since we assigned 1 to all the edges and the number of edges connected to s is n, and it is the same as the number of edges connected to t, $|f^*| \leq n$, where f^* is the maximum flow in G_f. If $f^* < n$, there is no feasible solution to the ROL problem, and the min-cost maximum flow assigns routing resources to the routes for as many net segments connected to the LUT as possible with minimum total delay costs. The following theorem shows that the ROL problem can be exactly solved by a network flow computation on G_f.

THEOREM 1. *A min-cost maximum flow f^* in G_f corresponds to a solution to the ROL problem with minimum total delay cost. If the size of f^* is n, the number of net segments connected to a LUT, then the ROL problem for the LUT is feasible so that all the net segments connected to the LUT can be routed.*

From the computed flow in G_f, a solution to the ROL problem can be derived by a depth-first search from each input pin node of the LUT to each subsource node in G_f. Because each subsource node s_i is associated to a net segment and it is connected to the nodes belonging to the partial routing tree associated with that net, the node connected to the subsource node among the nodes in partial routing tree becomes a branching point for the branch from the routing tree to the LUT. Figure 4 shows a flow f corresponding to a ROL solution for 4 net segments connected to a LUT. In this example, the net from l_1 is a multiple-pin net, and a portion of this net is routed while l_{14} is processed. Hence, the subsource s_2 is connected to all the nodes along the partially connected routing tree. Because only a flow of size one can flow through s_2, only one node is selected as a branching point. The routes correspond to the flow solution of Figure 4 is shown in Figure 5. Figure 6 summarizes our ROL_NF algorithm.

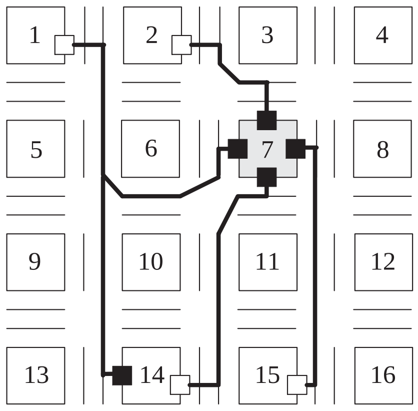

Figure 5: Routes correspond to the flow computed in Figure 4

```
Algorithm    ROL_NF
Input: $G(V, E)$, $R$, $C$, $l_k$
Output: routes for all net segments to $l_k$
begin
1. Construct the flow network $G_f(V_f, E_f)$
2. Assign costs and capacities
3. Run min-cost max-flow algorithm on $G_f(V, E)$
4. Derive the corresponding routes from the computed flow
end
```

Figure 6: Algorithm ROL_NF

There are several polynomial-time optimal algorithms available for finding a min-cost maximum flow in a network[2]. Deriving a solution to the ROL problem from a flow can be done in $O(E)$ time. Thus, the ROL problem can be solved efficiently as stated in the following theorem, if we adopt the double scaling algorithm[1].

THEOREM 2. *The ROL_NF algorithm exactly solves the ROL problem in $O(VE\log\log R_{max}\log(VC_{max}))$ time, where R_{max} is the maximum value of the capacities and C_{max} is the maximum value of the costs.*

3.2 Iterative refinement using Lagrangian relaxation

In this section we solve the FPGA detailed routing problem. We apply the ROL_NF algorithm successively on all the LUTs in an FPGA. To avoid ordering problem of selecting LUTs, we adopt an iterative refinement scheme based on Lagrangian relaxation. We relax the congestion constraints in the FPGA detailed routing problem formulated in the previous section. Each of the constraints is multiplied by the corresponding Lagrangian multiplier, and added to the objective function. Let

$$L_\lambda(\boldsymbol{x}) \quad = \quad \sum_k \sum_i c_i x_{ik} + \sum_i \lambda_i (\sum_k x_{ik} - 1)$$

circuit	# of LUTs	# of tracks		critical path delay (ns)			total wire length		
		vpr	FlowRoute	vpr	FlowRoute	improve	vpr	FlowRoute	improve
9symml	26	10	9	26.7	25.1	6.0%	472	434	8.0%
term1	32	13	12	25.3	23.3	7.9%	627	600	4.3%
apex7	63	13	13	26.1	21.3	18.4%	843	738	12.5%
example2	101	17	16	29.6	23.2	21.6%	1488	1402	5.8%
alu2	56	17	17	54.7	49.2	10.1%	1660	1560	6.0%
too-lrg	52	19	19	31.2	30.2	3.2%	1721	1590	7.7%
vda	114	23	23	46.5	38.9	16.3%	3254	2987	8.2%
alu4	390	33	33	143.4	122.5	14.6%	5130	4694	8.5%
s298	490	27	27	274.0	194.7	28.9%	18816	16319	13.3%

Table 1: Comparisons between VPR and FlowRoute in number of used routing tracks, delay, and total wirelength.

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

$$LS_\lambda : \quad \text{Minimize} \quad L_\lambda(\boldsymbol{x})$$

For any given $\boldsymbol{\lambda}$, we note that

$$\begin{aligned} L_\lambda(\boldsymbol{x}) &= \sum_k \sum_i c_i x_{ik} + \sum_i \lambda_i (\sum_k x_{ik} - 1) \\ &= \sum_k \left\{ \sum_i c_i x_{ik} + \sum_i \lambda_i x_{ik} \right\} - \sum_i \lambda_i \\ &= \sum_k \sum_i (c_i + \lambda_i) x_{ik} - \sum_i \lambda_i \end{aligned}$$

Because $\sum_i \lambda_i$ is a constant term, we can solve LS_λ for a given $\boldsymbol{\lambda}$ by solving

$$LS'_\lambda \quad = \quad \min \left\{ \sum_k \sum_i (c_i + \lambda_i) x_{ik} \right\}$$

We solve this reduced Lagrangian subproblem LS'_λ by solving the ROL problems for all LUTs in L after assigning $(c_i + \lambda_i)$ to each node (edge) i as a cost. After solving the ROL problem, we reset the capacity of the edges to allow the relaxation of the congestion constraints. To discourage using routing resources used in routing for nets to other LUTs, we define the c_i term as follows,

$$c_i = d_i * q_i$$

where d_i is a delay cost of node (edge) i, and q_i is the penalty term proportional to the number of nets using node (edge) i currently.

It is known that the minimum value of the Lagrangian subproblem for any given vector $\boldsymbol{\lambda}$ is a lower bound on the optimal objective function value of the original optimization problem. Hence, the tightest lower bound to the optimal objective function value is obtained by solving

$$L^* = \max_{\lambda \geq 0} LS_\lambda$$

which is known as the Lagrangian dual problem. 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 $(j+1)$th iteration are updated by the subgradient method [2, 5] as follows:

```
Algorithm   FlowRoute
Input: $G_f$, $R$, $C$, $L$
Output: Routing trees for all nets in an FPGA
begin
1. Initialize $\boldsymbol{\lambda}$
2. for each $l_k$ in $L$ do
3.       Rip up all the nets connected to $l_k$
4.       Call ROL_NF
5.       Update costs and reset capacities
6. Update $\boldsymbol{\lambda}$
7. Repeat Step 2-6 until no shared resource exists
end
```

Figure 7: Algorithm FlowRoute

$$\lambda_i^{j+1} = \max \left\{ 0, \lambda_i^j + \theta_j (\sum_k x_{ik} - 1) \right\}$$

where θ_j is a step size with the property that $\lim_j \theta_j \rightarrow 0$, and $\lim_j \sum \theta_j \rightarrow \infty$. We used $\theta_j = a/(j+b)$ as a step size, where a and b are constants. By multiplying the amount of violations of the congestion constraints, overly subscribed routing resources can be penalized over iterations.

Figure 7 summarizes our algorithm for the FPGA detailed routing problem, FlowRoute.

4. EXPERIMENTAL RESULTS

We have implemented our algorithms in C programming language on a SUN Sparc Ultra 5 (360MHz) with 128M memory. The experiments are performed on 9 circuits from MCNC benchmark [17]. The placed netlists were generated using the placer in VPR [16]. We assumed a symmetrical-array-based FPGA, 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 compared the minimum number of tracks per channel to achieve feasible routings for all nets, critical path delay, and the total wire length from FlowRoute with those from VPR router. Because FlowRoute is basically a congestion-driven detailed router, we compared our results from the ones obtained by running VPR router in congestion-driven mode. Results are shown in Table 1. FlowRoute used smaller number of tracks per channel for 3 circuits. Although FlowRoute is a congestion driven router, because it is based on min-cost flow computation, it shows improvement in critical path delay up to 28.9 %(average 14.1 %). Because the objective function of the problem of this paper is the total sum of delays of routing resources, we also compared total wire length. The wire lengths in the table are represented as integer multiple of one logic module length. The total wire length used to route all nets are reduced up to 13.3 % (average 8.3 %).

5. CONCLUSIONS

In this paper, we proposed a congestion-driven detailed router for FPGAs. In our algorithm, we route all the net segments connected to a LUT simultaneously rather than routing one net at a time. By routing several net segments simultaneously using min-cost flow computation, our algorithm can alleviate the net ordering problem. To avoid ordering problem in selecting LUTs, we adopted an iterative refinement scheme based on Lagrangian relaxation. Each of Lagrangian subproblems is solved by successive application of min-cost flow based routing algorithm on all the net segments connected to each LUT.

We could find feasible routings for the benchmark circuits with less or equal number of routing tracks per channel compared to VPR router. Furthermore, the total delays of the nets are also reduced, which can contribute to reducing critical path delay of the circuit. We compared our algorithm with VPR router and the experimental results shows that our algorithm is very effective.

6. REFERENCES

[1] R. K. Ahuja, A. V. Goldberg, J. B.Orlin, and R. E. Tarjan, "Finding minimum-cost flows by double scaling," *Mathematical Programming 53*, pp. 243-266, 1992.

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

[3] C. Albrecht, "Provably good global routing by a new approximation algorithm for multicommodity flow," *Proceedings of ACM ISPD-00*, pp. 19-35, 2000

[4] M. Alexander, G. Robins, "New performance-driven FPGA routing algorithms," *Proc. ACM/IEEE Design Automation Conference*, pp. 562-567, 1995.

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

[6] S. Brown, M. Khellah, G. Lemieux, "Segmented Routing for Speed-Performance and Routability in Field-Programmable Gate Arrays," in *Journal of VLSI Design*, 1996.

[7] R. C. Carden, C. K. Cheng, "A global router using an efficient approximate multicommodity multiterminal flow algorithm," *ACM/IEEE DAC-91*, pp. 316-321, 1991.

[8] 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.

[9] Yao-Wen Chang, D. F. Wong, C. K. Wong, "FPGA Global Routing Based on a New Congestion Metric," in *Proc. Intl. Conf. on Circuit Design*, 1995, pp. 372-378.

[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] J. Huang, X. L. Hong, C. K. Cheng, and E. S. Kuh, "An efficient timing-driven global routing algorithm," *ACM/IEEE DAC-93*, pp. 596-600, 1993.

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

[13] G. G. F. Lemieux, S. D. Brown, D. Vranesic, "On Two-Step Routing For FPGAs," in *Proc. International Symposium on Physical Design*, 1997, pp. 60-66.

[14] 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.

[15] G. Meixner, U. Lauther, "A new global router based on a flow model and linear assignment," *Proc. ICCAD-90*, pp.44-47, 1990.

[16] 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.

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

[18] 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.