Design and Implementation of FPGA Router for Efficient Utilization of Heterogeneous Routing Resources

Deepak Rautela, Rajendra Katti
Department of Electrical and Computer Engineering
North Dakota State University
Fargo, ND-58105, USA
{Deepak.Rautela,Rajendra.Katti}@ndsu.edu

Abstract

The routing resources available in recent FPGA architectures (e.g., Xilinx Virtex-II) are very different from the older generation of FPGAs (e.g., Xilinx XC4000). The latest FPGA architectures have heterogeneous routing resources which include directly driven wires of different lengths and connectivity. Since routing resources in FPGAs are fixed, it is very important for the routing algorithms to fully exploit the potential of new routing architectures. FPGA routing architectures are usually represented as a routing resource graph (RRG). In this paper we present a simplified scheme to build the RRG for FPGA architectures with heterogeneous routing resources. Using our RRG construction scheme we have built a routability driven FPGA router named "Bison". We also present two dynamic weight update based heuristics which we have incorporated into the router, so that efficient utilization of routing resources can be achieved.

1 Introduction

FPGA routers can be divided into two groups. The combined global-detailed routers [9, 2], which determine a complete routing path in one step, and the two-step routers, which first perform global routing [4] to determine the channel segments each net will use, and then perform detailed routing [8] to determine the wires each net will use within each of the specified channel segments. Most of the FPGA routing algorithms are usually implemented to target older generation of FPGAs. The recently developed FPGA architectures have heterogeneous routing resources which include directly driven wires of different lengths and connectivity. VPR [2], the academic placer and router targets XC4000 style architecture, which also includes wires of different length and connectivity, but these wires are not directly driven. The new FPGA architectures (e.g., Xilinx Virtex-II and Altera Stratix) use directly driven routing switches which result in a decrease in the routing area. The directly driven routing architecture is also more tolerant of routing stress and allows a smaller channel width to achieve a given performance [5]. Longer wire segments are intended for high-fanout and time critical signal nets. Shorter wire segments are intended for short connections to avoid wasting routing resources. It is very important for the FPGA routing algorithms to fully utilize the limited routing resources because they take up a significant portion of the chip area [6]. In [14], the authors propose an architecture-driven metric, which considers the available wire segments and their lengths to optimize the wiring cost for placement and global routing, but the targeted architecture was based on Lucent Technologies ORCA2C and Xilinx XC4000EX. In [7], authors present a wire-type assignment algorithm for Xilinx Virtex-II architecture. The algorthm is intended as an intermediate stage between global and detailed routing and is based on iteratively applying min-cost max flow technique to simultaneously route many nets. The algorithm is implemented for segments in a single channel only and the wire segment utilization data is not provided.

In this paper, we present a simplified approach to build the RRG for the latest FPGA routing architectures. We have successfully implemented this approach to build a routability driven router named Bison. We also present two dynamic weight update based heuristics, which we have incorporated into the router, so that efficient utilization of routing resources can be achieved. The rest of the paper is organized as follows. The FPGA routing architecture targeted in this paper is described in section 2, and the problem is defined in section 3. In section 4, we present the simplified RRG construction scheme. In section 5, we present the dynamic weight update heuristics for efficient utilization of different wire types. Experimental results are shown in section 6, and our conclusions are in Section 7.

Proceedings of the IEEE Computer Society Annual Symposium on VLSI
New Frontiers in VLSI Design
0-7695-2365-X/05 $20.00 © 2005 **IEEE**

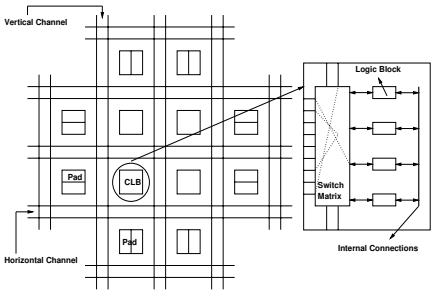

Figure 1. Target FPGA architecture

2 Architecture Details

In this section we describe the simplified FPGA architecture that we have targeted to implement our router. The fabric of latest FPGA architectures consists of CLBs (Configurable Logic Blocks), RAM, Multiplier blocks, power PC cores, and heterogeneous routing resources. In this paper we assume a homogeneous FPGA fabric architecture and only study the efficient utilization of heterogeneous routing resources. As shown in Figure 1, our architecture assumes an array-based FPGA, which consists of CLBs, input-output pads, and horizontal/vertical channels of interconnect wires. Each CLB consists of a switch matrix, logic blocks, and internal CLB local connections. Logic blocks are made up of basic logic elements (BLEs). A BLE consists of a 4-input look-up-table (LUT) and a flip-flop (FF). The BLE output can be either the registered or unregistered version of the LUT output. In Virtex-II architecture, a combination of 2 LUTs and 2 FFs is referred to as a slice, and each CLB can have 4 slices/CLB, thus each CLB has a maximum of 8 LUTs and 8 FFs. We simplify the creation of routing tool by assuming that each of the BLE inputs can be connected to any of the LB inputs or any of the BLE output. In Figure 2, the wire types used in Virtex-II are shown [1]. White squares denote CLBs and black squares denote route switch boxes. The architecture assumed in this paper contains all the routing resources available in Virtex-II architecture. The only exception are the fast local interconnections from the logic block outputs to the logic block inputs, which are not included in our architecture because we are only concerned with global routing resources. We assume that both horizontal and vertical channels have the same structure. All wires except the long lines are unidirectional and route signals in all the four directions. The long lines, which span the width of a horizontal channel or the height of a vertical channel, are bidirectional. Direct lines route signals to neighboring blocks, double lines route signals to every first or second block, type 1 HEX lines route signals to every third or sixth block and type2 HEX lines route signals to all the blocks upto the sixth block away except the first one. All the switch matrices have the same connection topology but different wires in the matrix have irregular connection topologies.

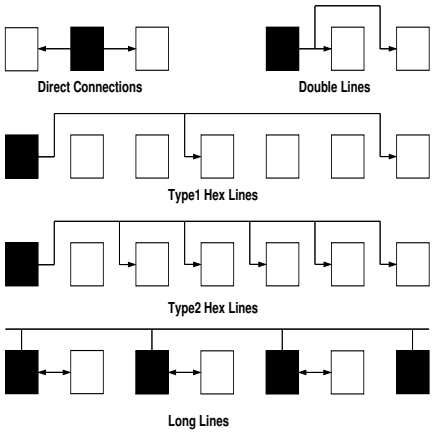

Figure 2. Virtex-II Wires

3 Problem Definition

Modern FPGA architectures incorporate heterogeneous routing resources, and in [11] and [12], the authors have shown that the number of different wire type segments used, instead of geometric (Manhattan) distance between the driver-driven pin is the most crucial factor in controlling the routing delay and cost in an FPGA. For a distance of 2 CLBs between a driver-driven pair, the number of possible routes is limited. Such a distance can be routed using either of the following: one double line, two double lines, two direct lines, a direct line and a double line, a type1 HEX line etc. The authors in [12], have shown that the delays for these different types of lines are almost constant and have no relation to each other. For example, delay of a HEX line is not three times the delay of a double line or six times the delay of a direct line but is only slightly larger than the delay of a double or direct line. Hence, the FPGA routing problem in current architectures is no longer to find route that travels the shortest geometric distance, but to find the route that consumes the minimum total number of segments and utilizes the wires of various lengths most effectively. The efficient routing resource utilization (ERRU) problem can thus be stated as the optimization of following two problems:

- Minimize the wastage of wire segment *length* for double and HEX lines. For example, if a double line is used for direct connection then there is wastage of 1 wire segment length.
- Maximize the utilization of wire *segments* available

Proceedings of the IEEE Computer Society Annual Symposium on VLSI
New Frontiers in VLSI Design
0-7695-2365-X/05 $20.00 © 2005 **IEEE**

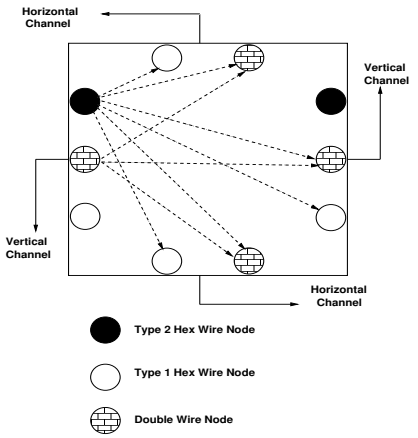

Figure 3. Typical Switch Matrix Connection.

with long lines. For example, if there are 10 CLBs in a horizontal row and the horizontal long line only connects two of these CLBs then there is a wastage of 8 long line segments.

4 Implementation Details

The presence of direct, double, long, type1 and type2 HEX wires make the construction of RRG a complex task. In this section we present a simplified RRG construction scheme, so that wires of different type can be accurately modeled. Using our simplified RRG construction approach, researchers can make changes to the graph and address other FPGA routing problems easily. Our RRG construction scheme is similar to VPR, but we have made some changes to accommodate the new routing resources available in Virtex-II style FPGA architectures.

4.1 Routing Resource Graph

Each wire is modelled as a node in the graph and to insert correct number of wires in the graph, we associate each wire with the CLB from where it (originates) is driven. Each side of a CLB is used to route wires in a specific direction as shown in Figure 3. Wire nodes on the top and the bottom of the CLB, route nets in the horizontal channel, in the west and the east direction respectively. Wire nodes on the left and the right of the CLB, route nets in the vertical channel, in the south and the north direction respectively. The presence of a direct wire from each CLB implies that there exists a direct wire from each CLB in all the four directions. This is true for all the wire types supported by the architecture assumed in this paper except the long wire. Long wires are bidirectional and can be driven from the end of any of its segments. Each node has a parameter "capacity" associated with it, which is the maximum number of different nets which can use this node in a legal routing. For example, if three direct wires are originating from each switch matrix, then each direct wire node on all the four sides of the CLB has a capacity of three. Each switch is modeled as a directed edge (for unidirectional switches, such as buffers), or a pair of directed edges (for bidirectional switches, such as pass transistors) between the two appropriate nodes. To reduce the size of RRG we model fixed number of wire nodes for each CLB. Each side of the CLB is modeled with one node of each wire type, and if more number of wires of that wire type exists then we just increase the capacity of the node. Since in our architecture, we have five different wire types (direct, double, long, type1 and type2 HEX), each CLB is modeled with five nodes for all the four sides. The connection between long wire segments is different from other wire types since it uses the long wire nodes present in the horizontal and vertical channels. Each channel has two nodes for long wires. In horizontal channel they are used to route CLBs present at the top and bottom of the channel. In vertical channels they are used to route CLBs present on the left and right of the channel.

We assume that all the pins are logically equivalent. This means that a router can complete a given connection using any one of the input pins of a LUT. We model this logical equivalence in the RRG by adding source and sink nodes. Each CLB is modeled with two nodes, a CLB source node, and a CLB sink node for all the logically equivalent output and input pins. Each pad is modeled with one node, a pad source node, or a pad sink node depending on whether the pad is input or output. To model the connections between input/output pins and wires in a CLB we have an edge from each CLB wire node to the CLB sink node. Similarly we have an edge from the CLB source node to all the wire nodes modeled by a CLB. The capacity of each source and sink node depends on the number of slices present in the logic block. In Virtex-II, all the switch matrices have the same connection topology, but different wire types in the matrix have irregular connection topologies. For example, a type2 HEX wire in a vertical channel has connection to vertical double wires, vertical type1 HEX wires, horizontal type1 HEX wires, and horizontal double wires. But a vertical double wire has connection to horizontal double wires and vertical double wires in the same switch matrix [7]. Since we have modeled wire nodes of each wire type for all the four sides of a CLB, any switch matrix connection topology can be created easily. In Figure 3, switch matrix connection topology for type2 HEX wire and double wire as described above has been shown. The Figure only shows the connection of type2 HEX and double wires present in the left vertical channel for the purpose of clarity, but the same connection topology is also present for type2 HEX and double wires present in the right vertical channel. In

this paper, we have performed our experiments assuming fully connected switch matrix, i.e., each wire node is connected to all the other wire nodes on all the other three sides.

4.2 Routing Algorithm

The routing algorithm for our router, Bison, is the Pathfinder negotiated congestion algorithm [9]. Pathfinder repeatedly rips-up and re-routes every net in the circuit until all congestion is resolved. Ripping-up and re-routing every net in the circuit once is called a routing iteration. A circuit routing in which some routing resource is overused, such as a wire being used by two different nets, is not a legal routing. When overuse exists at the end of a routing iteration, another routing iteration is performed to resolve this congestion. After each routing iteration the cost of overusing a routing resource is increased, so that the probability of resolving all congestion increases. In our implementation, we use the VPRs [3] cost function of using a routing resource node n, which is given by

$$NodeCost(n) = B(n) * P(n) * H(n)$$

$B(n)$ is the base cost of the node n. $H(n)$ is the historical congestion of node n and is increased after every routing iteration in which node n is overused and gives the router congestion memory. $P(n)$ is the present congestion cost of node n and it increases with the amount of overuse of the node. We then perform a breadth-first search by using the $TotalCost(n)$ of a node n as the sort value in the Priority Queue:

$$TotalCost(n) = NodeCost(n) + CostFromRT(n)$$

where $CostFromRT(n)$ is the total cost of the path from the current partial routing tree to node n.

5 Dynamic Weight Heuristics

The pathfinder algorithm iteratively routes one net at a time. If there is no criteria on using the various wire segments, then we can end up with a routing solution, with low fan-out nets using long wires and direct connections using double wires. Therefore, the resources need to be assigned weights in some proportion to their lengths and scarcity to avoid wastage of precious routing resources. $B(n)$ i.e., the base cost of the node is used to assign weights to all the wires, so that we can study the effective wire utilization of the static and dynamic weight update approaches. All the wire types are assigned $B(n) = 1$ for our static weighting scheme. This encourages the router to use as few of these resources as possible to route each connection. In the following sections we present two dynamic weight update schemes which optimize the ERRU problem.

5.1 Directed Search Heuristic

In this section we present heuristics to optimize the first part of the ERRU problem i.e., to minimize the wastage of wire segment length for double and HEX lines. The pathfinder algorithm is very well suited for FPGAs since it adapts very well to the RRG. It can be used to search for sinks in either a breadth-first or a A* directed search manner. The timing driven implementation of VPR router uses the A* method because it is faster than the breadth-first technique. To perform a directed-search we must be able

Algorithm 1 Directed Search Heuristic

```
/*
wiresVector[WireType][CLBSide][x][y]
WireType(0: Direct; 1: Double; 2: HEX-1; 3: HEX-2)
CLBSide (0: South; 1: West; 2: North; 3: East)
nodeWeight[Node][DistanceFromCLB]
(x1, y1): Coordinate of node n
(x2, y2): Coordinate of sink k
deltaX = x2 - x1; deltaY = y2 - y1;
*/
if(deltaX != 0 && deltaY == 0) {
//Update weight in horizontal direction
  CLBSide = (deltaX > 0) ? 0 : 2
  DistanceFromCLB =
     (abs(deltaX) >= 6 ) ? 6 : abs(deltaX);
  for(int i=0; i <= WireType; ++i) {
    node = wiresVector[i][CLBSide][x1][y1];
    ExpectedCost(n,k) =
    nodeWeight[node][DistanceFromCLB]*P(n)*H(n);
  }
}else if(deltaY != 0 && deltaX == 0) {
//Update weight in vertical direction
}else if(deltaX != 0 && deltaY != 0) {
//Update weight in horizontal & vertical direction
}else // deltaX = deltaY = 0
//No weight update required
```

to estimate the total remaining cost, $ExpectedCost(n, k)$, from the current node n to the sink node j. If the $ExpectedCost(n, k)$ is a lower bound, i.e., it is always less than or equal to the actual cheapest path cost from node n to sink k, then the directed-search is an A* search and it finds an optimal path. To compute the $ExpectedCost(n, k)$ to route from a wire segment, n, to the target sink k, VPR assumes that the connection to the sink k will be completed using wires of the same type (length) as node n. In the Virtex-II device different wire types in the switch matrix have irregular connection topologies. For example, a type2 HEX wire in a vertical channel has connection to vertical double wires, vertical type1 HEX wires, horizontal type1 HEX wires, and horizontal double wires. Thus, given the placement of two pins, estimating the number and types of wires that will be used to route them is a non-trivial problem [12]. As discussed in section 3, the FPGA routing problem in current architectures is to find a route that consumes the minimum total number of segments and utilizes

the wires of various lengths most effectively. Our directed-search heuristic achieves this objective by using a dynamic weighting scheme as shown in Algorithm 1. We perform a directed-search by using the $TotalCost(n)$ of a node n as the sort value in the Priority Queue:

$$TotalCost(n) = CostFromRT(n) + ExpectedCost(n, k)$$

where $CostFromRT(n)$ is the total cost of the path from the current partial routing tree to node n. $ExpectedCost(n, k)$ helps the pathfinder algorithm to search in the direction of the sink. If $deltaX$ for a driver driven pair of pins is 5 and $deltaY$ is 0, then some of the correct solutions are: 5 direct lines, 2 double and 1 direct line, 1 type1 HEX (length 3 segment) and 1 double line, 1 type2 HEX (length 5 segment) line. All of these connections are correct, but the best solution is the last one, because it uses 1 type1 HEX line and there is wastage of only 1 wire segment length. Therefore, from any source node in the RRG we can calculate the best possible way to reach any sink node at most 6 distance away from it. We store this information in

$$nodeWeight[Node][DistanceFromCLB]$$

array as $B(n)$, the base cost weight of the nodes. If the sink node is at a distance greater than 6 then we first calculate the best route to reach the node 6 distance away from the source and from there calculate new values of deltaX, deltaY and CLBSide to find the best route to the sink.

5.2 Long Line Heuristic

In this section we present heuristic to optimize the second part of the ERRU problem i.e., to maximize the utilization of wire segments available with long lines. Our long line heuristic is similar to the dynamic weighting scheme proposed by Nag and Rutenbar [10] and is shown in Algorithm 2. The heuristic is based on the semi-perimeter length

Algorithm 2 Long Line Heuristic

```
bbCost = semiPerimeter(boundingBox);
if (wireType == LONG) {
  B(longWireNode) =
  (bbCost >= 0.7 * legthOfLongLine * 2)? 0.1:1.0
}else
  B(n) = 1.0;  // For Direct, Double & HEX
```

of the bounding box of each net that is to be routed. For example, if more than 70% of twice the length of long line is inside the net's bounding box then we reduce the $B(n)$ value for long lines. This encourages the router to route the net using long lines with minimum wastage of segments.

6 Experimental Results

Our router, Bison, was implemented in C++/STL/BOOST. Boost Graph Library (BGL) was used to construct the RRG. We routed 20 MCNC benchmarks [13] using our router on Pentium-IV 3.0 GHz machine running Redhat Linux with 512 MB of memory. The properties of the benchmarks are summarized in Table 1. First, T-VPack [2] is used to map the netlist (in blif format) into logic clusters of 8 4-input LUTs and FFs to model the Virtex-II CLB which has 4 slices/CLB. Next, VPR placement tool is used for placement and the Bison routing tool is used to route the circuit. The circuit is repeatedly routed with different number of wires until the router finds the minimum number of wires required from each CLB. Finally, we use the routing information to calculate the effective wire segment utilization (EU). EU for long lines is equal to the number of wire segments used to route a net. If a long line has 10 segments and 5 of them are used, then EU = 50% for that long line. For double, type1 and type2 HEX lines, EU is equal to the percentage utilization of the length of the wire. For example, If a double line is used for direct connection then there is utilization of 1 wire segment length and EU = 50%. In Table 1, static and dynamic EU for double, long, type1 and type2 HEX lines is shown. The dynamic heuristic EU of all the wires for almost all the circuits show significant improvement. The only circuits (bigkey, des, dsip, and tseng), which do not show any significant long line improvement are pad limited in our architecture. In our implementation, we use the VPRs approach and each circuit is mapped into the smallest square FPGA that can accommodate it. For the pad limited circuits the size of FPGA is increased from its minimum size, and this causes our long line heuristic to fail. The source code for our router Bison is available at: http://www.cs.ndsu.nodak.edu/∼rautela/router.html.

7 Conclusions

In this paper, we presented a simplified scheme to build RRG for current FPGA architectures with heterogeneous routing resources. Although we target the Virtex-II style routing architecture, we believe that our RRG construction scheme can be used to model most of the recent architectures. We also presented two dynamic weighting heuristics to efficiently utilize different types of wires present in Virtex-II architecture. To experimentally show the merits of our approach, we used the RRG construction scheme and the heuristics, to build a routability driven router, named Bison. The results show significant improvement in effective utilization of double, long, type1 and type2 HEX lines when dynamic weight heuristics were used over static weighting scheme.

Table 1. Dynamic Weight Heuristic Results

Circuit	LUT /FFs	CLBs	Pads	Nets	Min # of wires	Effective Utilization %							
						Double		HEX–1		HEX–2		Long	
						Static	Dyn	Static	Dyn	Static	Dyn	Static	Dyn
alu4	1522	191	22	780	3	78.56	90.44	75.45	74.21	65.69	69.91	6.56	50.36
apex2	1878	235	41	1227	3	77.84	87.71	71.54	77.56	64.43	68.83	10.38	29.19
apex4	1262	158	28	858	3	77.52	88.20	72.94	79.29	66.14	69.97	13.48	35.24
bigkey	1707	214	426	1040	2	78.07	87.84	79.71	85.27	77.30	78.70	3.10	3.70
clma	8383	1048	144	5394	4	78.83	88.75	76.65	81.72	67.33	72.46	4.63	23.02
des	1591	199	501	1210	2	75.36	89.25	74.75	82.78	72.72	76.98	3.32	4.25
diffeq	1497	188	103	1023	3	75.05	86.29	66.03	70.44	59.67	66.34	8.33	25.04
dsip	1370	172	426	762	2	74.80	83.56	79.87	86.52	75.95	78.45	2.85	5.68
elliptic	3604	451	245	2281	3	77.94	87.79	75.53	82.37	67.94	72.64	4.84	14.28
ex1010	4598	575	20	3014	4	79.40	89.08	71.82	77.23	64.66	69.52	6.03	24.81
ex5p	1064	133	71	765	3	76.71	87.96	71.10	75.86	65.09	69.25	13.86	36.60
frisc	3556	445	136	2013	3	77.35	87.30	74.03	80.22	66.95	71.37	7.05	19.99
misex3	1397	175	28	841	3	78.75	88.33	71.43	75.06	64.16	68.69	11.48	35.94
pdc	4575	572	56	2548	4	79.90	89.45	76.05	82.28	67.98	72.27	6.94	27.39
s298	1931	242	10	789	3	77.81	89.79	70.93	77.60	64.67	71.07	9.59	51.54
s38417	6406	801	135	4362	3	73.85	83.11	71.34	76.04	61.89	68.50	4.76	18.85
s38584.1	6447	806	342	4164	3	74.76	84.75	73.80	78.60	65.37	70.41	3.65	9.88
seq	1750	219	76	1054	3	77.09	87.17	72.89	78.33	64.81	69.29	10.41	35.09
spla	3690	462	62	2076	4	78.56	89.84	75.45	81.33	65.69	70.52	6.56	27.00
tseng	1407	131	174	798	2	77.23	87.45	73.46	79.29	65.60	71.82	7.04	9.55
Avg						**77.27**	**87.70**	**73.74**	**79.10**	**66.70**	**71.35**	**7.24**	**24.37**

References

[1] Xilinx Inc. Virtex-II platform FPGAs: Complete data sheet, 2004.

[2] V. Betz and J. Rose. VPR: A new packing, placement and routing tool for FPGA research. In *Proceedings of the 7th International Workshop on Field-Programmable Logic and Applications*, pages 213–222. Springer-Verlag, 1997.

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

[4] Y.-W. Chang, S. Thakur, K. Zhu, and D. F. Wong. A new global routing algorithm for FPGAs. In *Proceedings of the 1994 IEEE/ACM International Conference on Computer-Aided Design*, pages 356–361. IEEE Computer Society Press, 1994.

[5] D. L. et. al. The Stratix routing and logic architecture. In *FPGA '03: Proceedings of the 2003 ACM/SIGDA eleventh International Symposium on FPGAs*, pages 12–20. ACM Press, 2003.

[6] R. Jayaraman. Physical design for FPGAs. In *Proceedings of the 2001 International Symposium on Physical Design*, pages 214–221. ACM Press, 2001.

[7] S. Lee, H. Xiang, D. F. Wong, and R. Y. Sun. Wire type assignment for FPGA routing. In *Proceedings of the 2003 ACM/SIGDA eleventh International Symposium on FPGAs*, pages 61–67. ACM Press, 2003.

[8] G. G. F. Lemieux, S. D. Brown, and D. Vranesic. On two-step routing for FPGAs. In *Proceedings of the 1997 International Symposium on Physical Design*, pages 60–66. ACM Press, 1997.

[9] L. McMurchie and C. Ebeling. Pathfinder: a negotiation-based performance-driven router for FPGAs. In *Proceedings of the 1995 ACM third International Symposium on FPGAs*, pages 111–117. ACM Press, 1995.

[10] S. K. Nag and R. A. Rutenbar. Performance-driven simultaneous place and route for island-style FPGAs. In *Proceedings of the 1995 IEEE/ACM International Conference on Computer-Aided Design*, pages 332–338. IEEE Computer Society, 1995.

[11] T. Taghavi, S. Ghiasi, A. Ranjan, S. Raje, and M. Sarrafzadeh. Innovate or perish: FPGA physical design. In *Proceedings of the 2004 International Symposium on Physical Design*, pages 148–155. ACM Press, 2004.

[12] M. Wang, A. Ranjan, and S. Raje. Multi-million gate FPGA physical design challenges. In *Proceedings of the 2003 International Conference on Computer-Aided Design*, page 891. IEEE Computer Society, 2003.

[13] S. Yang. Logic synthesis and optimization benchmarks user guide version. In *Yang, S. Logic synthesis and optimization benchmarks user guide version 3.0. Tech. rep., Microelectronics Center of North Carolina, Jan. 1991.*, 1991.

[14] C. Yao-Wen and C. Yu-Tsang. An architecture-driven metric for simultaneous placement and global routing for FPGAs. In *Proceedings of the 37th Conference on Design Automation (DAC'00)*, pages 567–572. IEEE Computer Society, 2000.