A New Global Routing Algorithm for FPGAs *

Yao-Wen Chang†, Shashidhar Thakur†, Kai Zhu‡, and D.F. Wong†

†Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712-1188

‡AT&T Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974

Abstract

As in traditional ASIC technologies, FPGA routing usually consists of two steps: global routing and detailed routing. Unlike existing FPGA detailed routers, which can take full advantage of the special structures of the programmable routing resources, FPGA global routing algorithms still greatly resemble their counterparts in the traditional ASIC technologies. In particular, the routing congestion information of a switch block essentially is still measured by the numbers of available rows and columns in the switch block. Since the internal architecture of a switch block decides what can route through the block, the traditional measure of routing capacity is no longer accurate. In this paper, we present an accurate measure of switch block routing capacity. Our new measure considers the exact positions of the switches inside a switch block. Experiments with a global router based on these ideas show an average improvement of 38% in the channel width required to route some benchmark circuits using a popular switch block, compared with an algorithm based on the traditional methods for congestion control.

1 Introduction

Field-programmable gate arrays (FPGA's) have been widely used in implementing Application Specific Integrated Circuits (ASIC's) since it first emerged in 1985 [3]. They offer a cheap and flexible solution for customized VLSI, providing fast manufacturing turnaround and low prototype costs. A typical symmetrical-array FPGA comprises a two-dimensional array of logic blocks interconnected by vertical and horizontal routing channels. See Figure 1 for an example of the architecture. The logic blocks are programmable cells which contain logic circuits that implement different logic functions. The routing channels consist of general routing resources which are used to connect the pins of the logic blocks. The switching network at the intersection of horizontal and vertical channels is referred to as a *switch block*. Each switch block has terminals on its four sides. Terminals on different sides of a switch block can be programmed to be electrically connected by internal switches. Thus nets can be routed through the switch block using some of these internal switches.

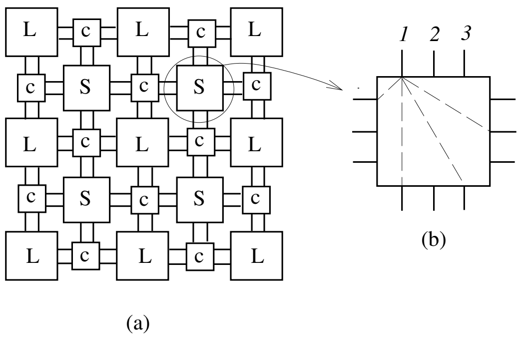

Figure 1: (a) The symmetrical-array FPGA model. (b) A switch block.

FPGA routing is a very complex combinatorial problem. In order to make it manageable, the routing problem is usually solved using the two-stage method of global routing followed by detailed routing. The goals of these two stages are, balancing the channel densities of all routing channels and assigning nets to specific tracks and internal switches, respectively. It has been shown by Trimberger and Chene [7] that the feasibility of FPGA design is constrained by routing resources more than by logic resources. Furthermore, previous work [6] [1] also has showed that routing delays, rather than logic block delays, dominate the performance of FPGA's. Thus there is a need to emphasize the importance of FPGA routing. The FPGA routing problem for symmetrical-array architecture was first studied by Brown *et al.* [2], in which a branch-and-bound method with pruning was used for routing. Like the classical methods, the global router was based on a graph search technique guided by channel density.

In this paper we propose an FPGA global routing algorithm using a new congestion metric. Our method is motivated by the intrinsic difference between the FPGA routing resources and those of the traditional ASIC routing. In traditional ASIC routing, wires can be routed in any available space within routing regions and thus channel density itself

*This work was partially supported by the Texas Advanced Research Program under Grant No. 003658459, by a DAC Design Automation Scholarship, and by a grant from AT&T Bell Laboratories.

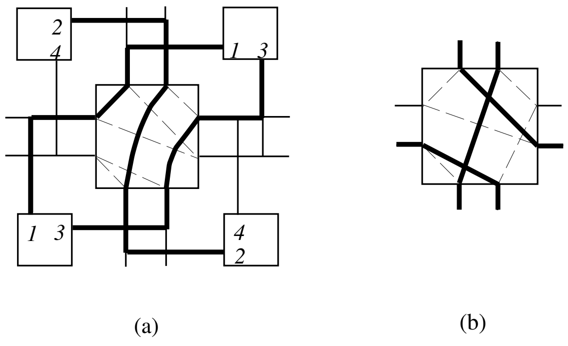

Figure 2: (a) An infeasible FPGA routing instance (though the channel densities do not violate the capacity constraints). (b) A switch block routing instance.

can capture feasibility information well. In contrast, the feasibility of routing in FPGA's is constrained not only by a set of fixed routing channels, but also by the physical architecture of a switch block. For example, in Figure 2, although no channel violates the channel capacity constraint of two, which is considered feasible in traditional ASIC global routing, there does not exist any FPGA routing solution by using the switch block shown in the figure. The reason is that the internal connections of the switch block are not flexible enough to afford four connections simultaneously. Thus the feasibility of FPGA routing is constrained more by the configuration of switch blocks than by that of channels. To precisely capture the FPGA routing nature, we need to consider the constraints induced by the structure of switch blocks.

In this paper we show a novel way to measure the congestion at individual switch blocks. As mentioned above, the number of available tracks is not an accurate estimate of the level of congestion. The internal architecture of the switch blocks places further constraints on the routability. Our method models the FPGA as a weighted graph. The weights on the edges are proportional to the congestion on the corresponding resources. We develop a technique that dynamically updates the weights based on the available resources. This weighting method is an extension of previous work by Thakur *et al.* [5] and Zhu *et al.* [9] for solving the switch block routing problem. We have tested our router on the five industrial benchmarks used in [2]. Experimental results show our global router consistently outperforms the traditional method. The classical router needs an average of 38% more channel width than the new router to route all nets on industrial benchmark circuits using a switch block similar to that used in Xilinx XC4000 series FPGA's.

The rest of this paper is organized as follows. In Section 2, we introduce some notation and define the problems that are addressed by this paper. In Section 3, we review some of the work in [5], which paves the way for acquiring the congestion control mechanism. In Section 4, we propose the new graph-based global routing algorithm. The experimental results are reported in Section 5.

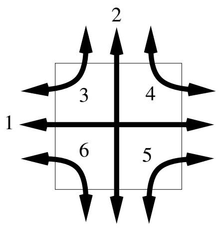

Figure 3: Six types of connections.

2 Problem Specifications

A switch block is a rectangular box with W_1 terminals on the left and right sides and W_2 on the top and bottom. Some pairs of terminals, on different sides of the box, may have programmable switches and the switches can be programmed to be connected or disconnected. Moreover, these switches are electrically non-interacting, unless they share a terminal. The specification of a switch block gives a list of such terminal pairs.

As mentioned earlier, nets can be routed through a switch block by programming some switches to be "on". To characterize such local routes, we say a *connection* is established in the switch block between two terminals, on different sides of the switch block, if the switch between those two terminals is programmed to connect them. Connections can be of six types as shown in Figure 3. The connection labeled i, $1 \leq i \leq 6$, in Figure 3, is said to be of *Type i*.

A *routing requirement vector* (*rrv* for short) $\vec{n}$ is a six-tuple $(n_1, n_2, n_3, n_4, n_5, n_6)$ where $0 \leq n_1 \leq W_1$, $0 \leq n_2 \leq W_2$, and $0 \leq n_3, n_4, n_5, n_6 \leq \min\{W_1, W_2\}$. For a given switch block and an *rrv*, a *routing* is a set of connections which are electrically non-interacting such that there are n_i of Type i connections, for $i \in \{1, \cdots, 6\}$. Note that a set of connections are electrically non-interacting if and only if the terminals on any two paths are distinct. An *rrv* $\vec{n}$ is said to be *routable* on a switch block S if there exists a routing for $\vec{n}$ on S. For example, in Figure 2(a), a switch block and a routing for the *rrv* $(0, 1, 1, 0, 1, 0)$ on this switch block are shown. The *rrv* $(0, 1, 1, 0, 1, 1)$ is not routable on the same switch block.

We first define the following problem.

The *Switch Block Routing Problem* (*SBRP*): Given a switch block S and an *rrv* $\vec{n}$, is $\vec{n}$ routable on S?

For convenience, we often refer to the problem as simply SBRP with $\vec{n}$, omitting the input S. The problem is defined on single switch blocks in isolation. Our algorithm uses solutions to it as subroutines.

A *global route* for a net is an assignment of a sequence of channels and switch blocks to the net so that all the terminals of the net are connected. We assume that no jogs are used within switch blocks. The problem addressed in this paper is stated below.

FPGA Global Routing Problem: Given an FPGA architecture and a set of multiterminal nets, find a global route for each net.

3 The Switch Block Routing Problem

Our global router is based on a novel congestion metric of switch blocks. The measure is formally introduced in the next section. The idea is to route the nets one at a time. After each net has been routed, we shall update certain costs of using switch blocks. The cost of a switch block is a function of the current usage of the switch block and of the potential for further incremental use of the block. To estimate how many more nets can be routed through the block, we show how to do some preprocessing and store a list of maximally routable *rrv*'s. [1] Then, we shall use some metric for the distances between these maximally routable *rrv*'s and the *rrv* corresponding to the currently routed nets to determine the cost associated with a switch block.

The first subsection shows how to solve the SBRP using an integer linear program (ILP). The second subsection uses this solution to compute a list of maximally routable vectors.

3.1 The Routing Problem

Consider an SBRP with the *rrv* $(n_1, \ldots, n_6)$ and the switch block S. We formulate the problem as an Integer Linear Programming problem (ILP). We write an ILP for the corresponding SBRP. We have two sets of inequalities. The first set of inequalities is used to ensure that every terminal is used at most once. The second set of six inequalities, with the objective function, are used to ensure that the routing generated by the solution to the ILP routes as many of the connections specified by the *rrv*.

Label the terminals as $1, 2, \ldots, 2(W_1 + W_2)$ starting from the lower most terminal on the left side and proceeding clockwise. The programmable switches are specified by *sets* containing pairs of the terminals they connect. The terminals of a given connection come from different sides, as stated before. Let $N = \{\{i,j\}|$ there exists a programmable switch between terminals i and $j\}$. Let $t_1 = \{1, 2, \ldots, W_1\}$, $t_2 = \{W_1+1, \ldots, W_1+W_2\}$, $t_3 = \{W_1+W_2+1, \ldots, 2W_1+W_2\}$, $t_4 = \{2W_1+W_2+1, \ldots, 2(W_1+W_2)\}$. These sets identify the terminals of each of the four sides of the switch block. Define a variable $x_{\{i,j\}}$ for each programmable switch $\{i,j\} \in N$. This is a decision variable that is chosen to be 1, if the corresponding connection is chosen for the routing, else it is 0. The number of variables $= |N|$ and number of constraints $= |N| + 2(W_1 + W_2) + 6$. Figure 4 shows the ILP and Theorem 1 states the correctness of the formulation.

Theorem 1 *The problem ILP has a solution with objective value $\sum_{i=1}^{6} n_i$ if and only if the rrv $(n_1, \ldots, n_6)$ is routable on S.*

3.2 Minimal Dominating Set

Using the solution for the SBRP, above, we develop an algorithm to precompute a set of maximally routable vectors for S. For this subsection, *fix* a switch block S. Consider solving SBRP on S for various *rrv*'s. Using our algorithm in section 3.1, an instance of integer programming problem is solved for each *rrv* and thus the set of routable *rrv*'s with

Problem ILP.

$$
\begin{array}{rcl}
\max \sum\limits_{\{i,j\}\in N} x_{\{i,j\}} & & \\
\sum\limits_{\substack{j \\ \{i,j\}\in N}} x_{\{i,j\}} & \leq & 1, \\
\text{for each } i = 1, \quad \ldots, & & 2(W_1+W_2) \\
\sum\limits_{\{i,j\}\in N, i\in t_1, j\in t_3} x_{\{i,j\}} & \leq & n_1 \\
\sum\limits_{\{i,j\}\in N, i\in t_2, j\in t_4} x_{\{i,j\}} & \leq & n_2 \\
\sum\limits_{\{i,j\}\in N, i\in t_1, j\in t_2} x_{\{i,j\}} & \leq & n_3 \\
\sum\limits_{\{i,j\}\in N, i\in t_2, j\in t_3} x_{\{i,j\}} & \leq & n_4 \\
\sum\limits_{\{i,j\}\in N, i\in t_3, j\in t_4} x_{\{i,j\}} & \leq & n_5 \\
\sum\limits_{\{i,j\}\in N, i\in t_4, j\in t_1} x_{\{i,j\}} & \leq & n_6 \\
x_{\{i,j\}} \in \{0,1\} & \textit{and} & \{i,j\} \in N
\end{array}
$$

Figure 4: ILP Formulation for Switch Block.

respect to S can be computed. Since the set of maximally routable *rrv*'s well characterizes the feasibility condition of routing on the switch block, as mentioned earlier, it then can be applied to guide a global router.

We also describe a pre-computation on S so that, following this pre-computation, we get a list of maximally routable *rrv*'s. For a given S, a set of *rrv*'s are identified during the pre-computation (this involves solving several integer programs). Following this computation, the routability of an *rrv* $\vec{n}$ as well as the potential for routing further nets through S can be quickly determined by comparing $\vec{n}$ with this set of *rrv*'s; both the computation of this set and the comparison of a given *rrv* with the *rrv*'s in this set is now described. First consider solving SBRP.

An *rrv* $(n_1, \ldots, n_6)$ is said to *dominate* another *rrv* $(m_1, \ldots, m_6)$ if and only if $n_i \geq m_i, i = 1, \ldots, 6$. It is a simple observation that any *rrv* $\vec{m}$ is routable if another *rrv* $\vec{n}$ is routable on S and $\vec{n}$ dominates $\vec{m}$. Intuitively, we wish to compute the set of all *rrv*'s which dominate all the routable *rrv*'s for S. We formalize this below.

A set D of *rrv*'s is called a *dominating set* for a switch block S, if for an *rrv* $\vec{v}$, $\vec{v}$ is routable on S if and only if there exists an *rrv* $\vec{w} \in D$ such that $\vec{w}$ dominates $\vec{v}$. A dominating set D for S is called *minimal* if $\forall \vec{v}, \vec{w} \in D$ neither $\vec{v}$ dominates $\vec{w}$ nor $\vec{w}$ dominates $\vec{v}$.

Let $W = \min\{W_1, W_2\}$. Let V be the set of *rrv*'s for S. Define $L(\alpha) = \{\vec{v} \in V | \sum_{i=1}^{6} v_i = \alpha\}$. An *rrv* $\vec{w}$ is a *child* of *rrv* $\vec{v} \in L(\alpha)$ if $\vec{w} \in L(\alpha - 1)$ and $\vec{w}$ differs from $\vec{v}$ in exactly

[1] Informally, an *rrv* $\vec{n}$ is maximally routable for a switch block S if, once the nets corresponding to $\vec{n}$ are routed on S, no further nets can be routed through S.

one component. A *parent* is similarly defined.

We describe an algorithm to compute the minimal dominating set for a given switch block S. Our algorithm proceeds in levels $1 \cdots W_1 + W_2$. At level β, the set of rrv's in $L(\beta)$ is considered. In particular, only those rrv's in $L(\beta)$, all of whose children in $L(\beta - 1)$ are routable, are considered. For each such rrv, using the integer programming approach in Section 3.1, it is determined if the rrv is routable. All the rrv's that were considered in level $\beta - 1$ which have the property that none of their parents in level β are routable, are output. Note that it is sufficient to stop the algorithm after level $W_1 + W_2$, since in succeeding levels, the rrv's require more tracks than are available, to be routed. It is easy to see that the set of output elements of our algorithm is the minimal dominating set.

Computing the minimal dominating set D for S completes the pre-computation. Following this, consider solving SBRP with rrv $\vec{v}$. Clearly, $\vec{v}$ is routable if and only if there exists some rrv in D which dominates $\vec{v}$. This can be checked quickly by successively performing binary search on the components of the six-tuples in a straightforward manner. Note in particular that no integer programming problem need be solved. Thus precomputing the exact routing for each of these rrv's obviates the need to do any expensive computations while doing the global routing, i.e., we compute the minimal dominating set off-line.

4 Global Routing Algorithm

In this section, we use the precomputed minimal dominating set of rrv's from the previous section to define a new congestion metric in switch blocks. It will be used to model the FPGA as a weighted graph. We first introduce two definitions. We define the *switch block density* of a switch block S, denoted by $\vec{d_S}$, as a vector $(m_1, m_2, \ldots, m_6)$, $m_i \geq 0$, $1 \leq i \leq 6$, where m_i is the number of Type i connections currently routed through S. We define $C_{\vec{d_S}} = \{\vec{v} \in D | \ \vec{v}$ dominates $\vec{d_S}$, D is the minimal dominating set of S. $\}$. Since the feasibility condition with respect to a switch block S can be characterized by its minimal dominating set, we can model congestion as a function of $\vec{d_S}$ and $C_{\vec{d_S}}$. The global routing algorithm is based on a graph search technique guided by the congestion information associated with switch blocks. The router assigns higher costs to route nets through congested areas of the FPGA to balance the net distribution among channels.

4.1 Modeling the FPGA

Before we can apply the graph search technique to FPGA routing, we first need to model the FPGA as a graph such that the graph topology can represent the FPGA architecture. Figure 5 illustrates the FPGA modeling. As shown in Figure 5, each logic block or connection block is represented by a node and each routing channel is modeled as an edge called a *channel edge*. We use six edges and four nodes to model the six possible types of nets routing through a switch block. These six edges are referred to as *switch edges*. See Figure 5(b) and 5(c) for the modeling. Paths in the graph represent routes on the FPGA, and vice versa. Weights associated with edges represent congestion information.

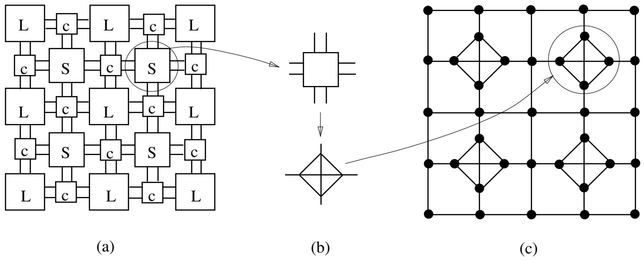

Figure 5: The FPGA graph modeling. (a) A symmetrical-array FPGA architecture. (b) The switch block modeling. (c) The FPGA modeling.

4.2 The Global Routing Algorithm

The global router is based on a modified Dijkstra's shortest path algorithm [4]. Unlike the classical ASIC global router which is guided by channel density, the FPGA global router is guided by switch block density. The main goal is to evenly distribute the nets among channels so that the channel width required to route all nets is minimized. The algorithm does the routing net by net. For the net being routed currently, we prefer to route it along uncongested routing regions. With this in mind, we design the following *cost function* with respect to a switch block S to guide the routing of the current net:

$$\alpha(\vec{d_S}, C_{\vec{d_S}}) = \max_{\vec{n} \in C_{\vec{d_S}}} \left(\sum_{i=1}^{6} a/b^{(n_i - m_i)}\right), \qquad (1)$$

where a and b are positive constants, and $\vec{d_S} = (m_1, \ldots, m_6)$. We call $n_i - m_i$ as the *slack* with respect to the i-th components of $\vec{n}$ and $\vec{d_S}$.

The whole routing procedure is illustrated in Figure 6. Given an FPGA F, we first construct a graph G to model F. Initially, $C_{\vec{d_S}}$ is set to the precomputed minimal dominating set D of S and the weights of all switch edges on F are computed from the cost function using zero switch block density, i.e., $\vec{d_S} = (0, 0, 0, 0, 0, 0)$. See Figure 6(a) for the initial configuration. The router assigns zero weights to all channel edges on F and these weights will never be changed throughout the routing, i.e., the router is guided by switch block density alone. After a net is routed, both $\vec{d_s}$ and $C_{\vec{d_S}}$ need to be updated to reflect the additional congestion resulting from the routing of the net. Therefore weights associated with the switch edges on the route are recomputed using the updated $\vec{d_S}$ and $C_{\vec{d_S}}$, and the cost function shown above. See Figure 6(b) and 6(c) for the update. In Figure 6(c), those rrv's which no longer dominate $\vec{d_S}$ are removed from $C_{\vec{d_S}}$ during the update. The process continues as routing proceeds. See Figure 6(d). Notice that the cardinality of $C_{\vec{d_S}}$ monotonically decreases during the process. We let $C_{\vec{d_S}}$ resume the initial configuration, i.e., $C_{\vec{d_S}} \leftarrow D$, when its cardinality equals zero. See Figure 6(e). The algorithm is shown in Figure 7.

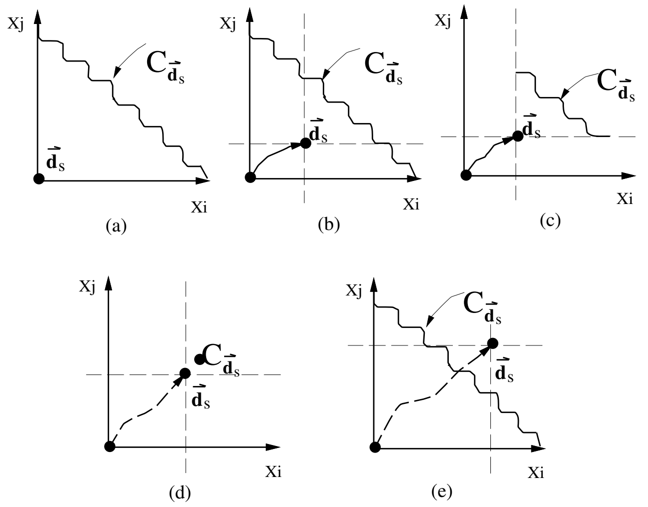

Figure 6: Dynamically update congestion information during routing, illustrated in a 2-D plane with axes X_i and X_j. (a) The initial stage. (b)(c) Update $\vec{d_S}$ and $C_{\vec{d_S}}$. (d) Only one vector remains in $C_{\vec{d_S}}$. (e) Resume the initial $C_{\vec{d_S}}$ when none in $C_{\vec{d_S}}$ dominates $\vec{d_S}$.

Algorithm: FPGA_Global_Routing$(F, S, D, \mathcal{N})$
Input: F - FPGA architecture;
S - switch block specification;
D - the minimal dominating set of S;
$\mathcal{N}$ - netlist of 2-terminal nets.
Output: Global routing for $\mathcal{N}$ on F.
begin
$C_{\vec{d_S}} \leftarrow D$ and $\vec{d_S} \leftarrow (0,0,0,0,0,0)$;
Construct graph G to model F;
Assign initial weights to edges in G;
for each net $n \in \mathcal{N}$
find a minimum cost route for n in G;
update weights of the switch edges on the route, $\vec{d_S}$, and $C_{\vec{d_S}}$ accordingly;
endfor
end

Figure 7: Global Routing Algorithm.

5 Experimental Results

We used the popular ILP solver, *lp_solve*, to obtain the minimal dominating sets for different switch blocks. We implemented the FPGA global router in C on a SUN SPARC 1 station. Two criteria are used to evaluate the quality of FPGA routing solutions: the channel width required to route 100% of nets and the percentage of routing regions which do not violate the capacity constraints for a given channel width W.

We tested our router on five industrial benchmark circuits used in [2]. Table 1 gives the names of circuits, size of FPGA's (i.e., number of logic modules in the FPGA's) , number of equivalent two-terminal nets in the circuit, and the function of each circuit. For the purpose of comparison, we also implemented a global router based on the same shortest path algorithm [4] using the traditional measure of channel density as a congestion control parameter. All benchmark circuits were routed by this method also and using the same net ordering. Contrary to the method used by our new router, it assigned zero weights to all switch edges while the weights of channel edges were dynamically updated by the following cost function during the routing:

$$\beta_i(\vec{d}) \quad = \quad a/b^{W-d_i}, \qquad (2)$$

where a and b are positive constants and $\vec{d} = (d_1, \ldots, d_k)$, is the current *channel density vector*. Here, d_i is the density in channel i of the FPGA and k is the number of channels of the FPGA. Hence $W - d_i$ denotes the *slack* with respect to channel i.

For FPGA's, the capacity of a channel is the size of the corresponding side of a switch block, W. In the experiment, we used the parameters $a = 1000$ and $b = 2$. The switch block used was similar to that of Xilinx XC4000 series FPGA's [8]. The *switch block flexibility* F_s is three. By switch block flexibility (F_s) we mean the number of programmable switches between a terminal and the others. Figure 8 illustrates the switch block architecture. Table 2 gives the comparison results of both classical and our routers for these benchmarks. The benchmark circuits in Table 1 were routed on the architecture using both, the traditional and our new, algorithms. At the end of global routing, a switch block S is called *feasible* if there exists an *rrv* $\vec{v} \in C_{\vec{d_S}}$ such that $\vec{v}$ dominates the corresponding $\vec{d_S}$. Table 2 lists the channel widths required for routing all the nets such that all switch blocks are feasible. It shows that our algorithm outperforms the old algorithms on all circuits and the classical router (old router) needs an average of 38% more channel width to route all nets than our router (new router) using the switch block shown in Figure 8. We also compared the percentage of feasible switch blocks for specified W's for the benchmarks. Figure 9 shows the experimental results for the DMA benchmark circuit. The percentage of feasible switch blocks (represented by the vertical axis) is plotted as a function of the channel width (represented by the horizontal axis). This is done for both algorithms. Though not presented here, the results of all the other circuits are similar. These results show that our router consistently outperforms the classical router.

6 Acknowledgments

The authors would like to thank Professor Stephen Brown of University of Toronto for providing us with the benchmark circuits.

References

[1] N. Bhat and D. Hill, "Routable Technology Mapping for LUT FPGA's," *Proc. Intl. Conf. Computer-Aided Design*, pp. 95–98, 1992.

Circuit	FPGA size	# 2-pin nets	Type
BUSC	12×13	392	Bus Cntl
DMA	16×18	771	DMA Cntl
BNRE	21×22	1257	Logic/Data
DFSM	22×23	1422	State Mach.
Z03	26×27	2135	Mult.

Table 1: Benchmark Circuit Descriptions.

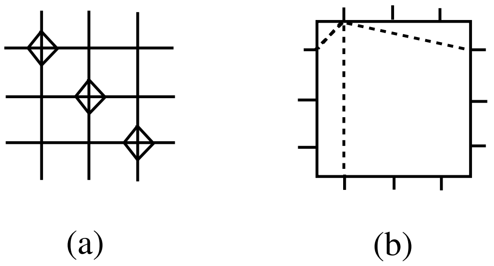

Figure 8: (a) The Xilinx XC4000-type switch block architecture ($F_s = 3$). (b) The corresponding switch block model.

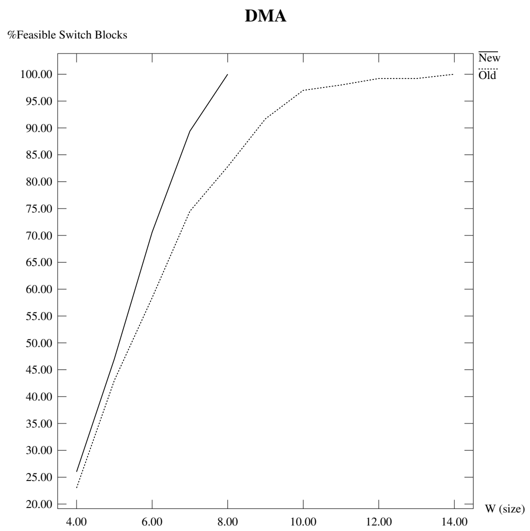

Figure 9: Experimental results for the circuit DMA.

Circuit	Old	New
BUSC	10	9
DMA	14	8
BNRE	14	11
DFSM	12	9
Z03	16	11
Total	66	48
Factor	1.38	1.00

Table 2: Channel width required for routing all nets keeping all switch blocks feasible.

[2] S. Brown, J. Ross, and Z.G. Vranesic, "A Detailed Router for Field-Programmable Gate Arrays," *IEEE Trans. Computer-Aided Design*, vol. 11, pp. 620–627, 1992.

[3] W. Carter *et al.*, "A User Programmable Reconfigurable Gate Array," *Proc. 1986 Custom Integrated Circuits Conference*, May 1986, pp. 233–235, 1986.

[4] E. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik*, 1:269–271, 1959.

[5] S. Thakur, D.F. Wong, and S. Muthukrishnan, "Algorithms for FPGA Switch Module Routing," *Proc. Euro-DAC*, to appear, 1994.

[6] S. Trimberger, *ed.*, *Field-Programmable Gate Array Technology*, Kluwer Academic Publishers, 1994.

[7] S. Trimberger and M. Chene, "Placement-Based Partitioning for Lookup-Table-Based FPGA's," *Proc. Intl. Conf. Computer-Aided Design*, pp. 91–94, 1992.

[8] Xilinx Inc., *The Programmable Logic Data Book,* 1994.

[9] K. Zhu, D.F. Wong, and Y.-W. Chang, "Switch Module Design with Application to Two-Dimensional Segmentation Design," *Proc. Intl. Conf. Computer-Aided Design*, pp. 481–486, 1993.