Algorithms for an FPGA Switch Module Routing Problem with Application to Global Routing

Shashidhar Thakur, *Member, IEEE,* Yao-Wen Chang, *Associate Member, IEEE,* D. F. Wong, and S. Muthukrishnan

***Abstract*— We consider a switch module routing problem for symmetrical-array field-programmable gate arrays (FPGA's). This problem was first introduced in [21]. They used it to evaluate the routability properties of switch modules which they proposed. Only an approximation algorithm for the problem was proposed by them. We give an optimal algorithm for the problem based on integer linear programming (ILP). Experiments show that this formulation leads to fast and efficient solutions to practical-sized problems. We then propose a precomputation that eliminates the need to use ILP on-line. We also identify special cases of this problem that reduce to problems for whom efficient algorithms are known. Thus, the switch module routing problem can be solved in polynomial time for these special cases. Using our solution to the switch module routing problem, we propose a new metric to estimate the congestion in each switch module in the FPGA. We demonstrate the use of this metric in a global router. A comparison with a global router guided by the density of the routing channels shows that our metric leads to far superior global and detailed routing solutions.**

***Index Terms*—Field-programmable gate array, global routing.**

I. Introduction

In the symmetrical-array FPGA architecture [1], [8], [20], routing resources consist of horizontal and vertical channels and their intersecting areas. The layout in such an architecture is shown in Fig. 1. An intersecting area of horizontal and vertical channels is referred to as a switch module. A net can change its routing direction via a switch module, and such a direction change requires going through at least one programmable switch inside the switch module. Researchers have shown in [2], [18], and [19] that the feasibility of FPGA design is most constrained by routing resources, and circuit performance in FPGA's is most limited by routing delays. Thus, switch-module design and routing are of significant importance in the design and use of FPGA's.

Due to the area constraints of switch modules and delay constraints of routing, the number of switches which can be put inside a switch module is usually limited. On the other hand, fewer switches in a switch module would reduce routability. Thus, this presents a problem of designing switch modules to maximize the routability under area and delay constraints. An experimental evaluation of the effect of varying different parameters, like switch-module and connection-module flexibilities, on the amount of routing resources needed to complete routing was reported in [3]. This provides an empirical way to choose a routing architecture. In contrast, Zhu *et al.* in [21] presented an algorithm for switch-module design that generated designs, given a distribution for the nets, that provide good routability. They did it for more general switch-module architectures than [3]. In order to evaluate their designs, they introduced a switch-module-routing problem, which was the key problem for analyzing the routability of a switch module with respect to various routing instances in the provided distribution. This switch-module-routing problem is addressed in this paper.

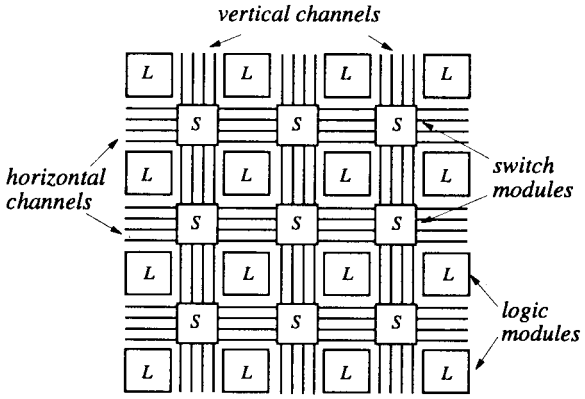


Fig. 1. Symmetrical array architecture.

Informally, the switch-module-routing problem is described as follows. The input is a switch-module description and a sextuple specifying the number of nets that have to pass through the switch module in the six possible directions through the switch module, namely, the four directions that require a 90-degree turn and the two that pass through. The problem is to determine a configuration of the switches in the switch module that allows the specified number of routings. A more formal definition will be given in the next section.

A network flow based algorithm was developed in [21] for the switch-module-routing problem. But the algorithm was approximate in the sense that it overestimated routability. In this paper, we present an optimal algorithm for the problem, based on integer linear programming (ILP). Although the

Manuscript received January 6, 1995; revised November 3, 1996. This work was supported in part by the Texas Advanced Research Program under Grant 003658459 and a DAC design automation scholarship. Preliminary versions of the work reported in this paper were presented at EURO-DAC '94 [17] and ICCAD '94 [5]. This paper was recommended by Associate Editor, M. Sarrafzadeh.

S. Thakur is with Synopsys Inc., Mountain View, CA 94043 USA.

Y.-W. Chang is with the Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan 300.

D. F. Wong is with Department of Computer Sciences, University of Texas, Austin, TX 78712 USA.

S. Muthukrishnan is with Lucent Technologies, Murray Hill, NJ 07974 USA.

Publisher Item Identifier S 0278-0070(97)01280-3.

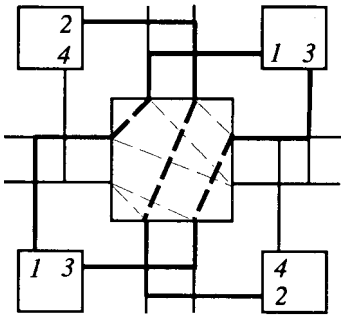


Fig. 2. An infeasible FPGA routing instance.

algorithm, in the worst case, does not run in polynomial time, experimental results consistently show that our algorithm is very efficient for practical-sized switch modules. For example, running times for all the 20 × 20 switch modules we considered averaged about 0.25 s of central processing unit (CPU) time. We further improve this approach by proposing a method that avoids having to solve the integer programming problems when actually solving the switch-module-routing problem. This is done by performing some preprocessing on the given switch module. We also identify interesting special cases of the switch-module-routing problem which can be solved optimally in polynomial time. This is achieved by reducing them to instances of bipartite-matching problems and network-flow problems.

Some previous work on FPGA routing [3], [4] suggested that it was a sensible goal for global routers to balance channel densities in all the channels of the FPGA. However, in FPGA's, the physical architecture of the switch modules constrains the routing more than channel capacity, as illustrated by the following example.

Example 1: For the switch module and the net specification in Fig. 2, let us suppose the global route for all four nets uses the shown switch module. The density of each channel is two and hence does not exceed the capacity of the channels. Thus, this would be a valid global route if one were concerned about channel density only. Nevertheless, given this global route, the switches available in the switch module do not permit a valid detailed route of all the nets. The thick lines show a feasible detailed route for three of the nets, which is the largest possible number of nets that can be routed through this switch module.

Based on our optimal solution for the switch-module-routing problem, we give a way of estimating congestion at individual switch modules in an FPGA. We propose a novel metric for quantifying the congestion level at each switch module in the FPGA. This can be used to generate global-routing paths which avoid heavily congested switch modules. We developed a global router based on this congestion metric. The router was able to route benchmark circuits, consistently using smaller routing resources as compared to a channel-density-guided global router (22% less channel width required for routing completion on an average using Xilinx XC4000-like switch modules on the CGE [4] and SEGA [14] benchmarks).

The rest of the paper is organized as follows. In Section II, we introduce the notation we shall use. Section III gives an ILP-based solution to the switch-module-routing problem defined formally in the next section. Section IV explores some special cases that can be solved efficiently and optimally. Section V shows how preprocessing can be used to avoid use of ILP at run time. Section VI shows the use of this theory in the development of a global router. Finally, Section VII shows the experimental data.

II. Definitions and Problem Specification

A *switch module* is a $W_1 \times W_2$ rectangular box with W_1 terminals on the left and right faces and W_2 terminals on the top and bottom faces. Within a switch module, various terminals are interconnected in some manner dependent on the module. A switch module can be one of two types, namely, a *switch matrix* or a *switch block*.

A *switch matrix* is a rectangular grid of W_1 *horizontal* tracks and W_2 *vertical* tracks. These tracks are electrically noninteracting. The horizontal tracks are numbered top to bottom and the vertical tracks left to right. A switch matrix comprises two types of switches, namely, *crossing switches* and *separating switches*. These switches are utilized in establishing connections between the tracks. Crossing switches are found at the intersection of a horizontal track and a vertical track. A crossing switch between a horizontal track i and a vertical track j has the following property. When on, it connects tracks i and j electrically. When off, these two tracks are electrically noninteracting. Separating switches are found anywhere along a track, subject to the constraint that each horizontal or vertical track has at most one separating switch. A separating switch on track i, when off, splits track i into two electrically noninteracting tracks. When on, track i becomes a single electrical track. A switch matrix M is the specification of the placement of crossing switches and separating switches on a given $W_1 \times W_2$ grid. An example switch matrix is shown in Fig. 3(b).

A *switch block* is a rectangular box with W_1 terminals on the left and right faces and W_2 on the top and bottom. Some pairs of terminals on different faces of the box may have programmable electrical links, i.e., these pairs can be programmed to be connected or disconnected. Moreover, these links are electrically noninteracting, unless they share a terminal. The specification of the switch block gives a list of such terminal pairs. An example switch block is shown in Fig. 3(a).

Henceforth, a *connection* is an electrical path in the switch module between two terminals on different faces of the switch module. Connections can be of six types as shown in Fig. 3(c). The connection labeled i, $1 \leq i \leq 6$, in Fig. 3(c) is said to be of *Type* i. Type 1 and Type 2 connections are called *straight* connections and Types 3, 4, 5, and 6 are called *bent* connections. For a switch matrix, it is additionally required that at most one switch be found on the electrical path comprising the connection. Thus, only straight connections can use a separating switch. For a switch block, connections have to be chosen from the programmable links specified.

A *routing requirement vector* (rrv) n is a sextuple $(n_1, n_2, n_3, n_4, n_5, n_6)$ 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

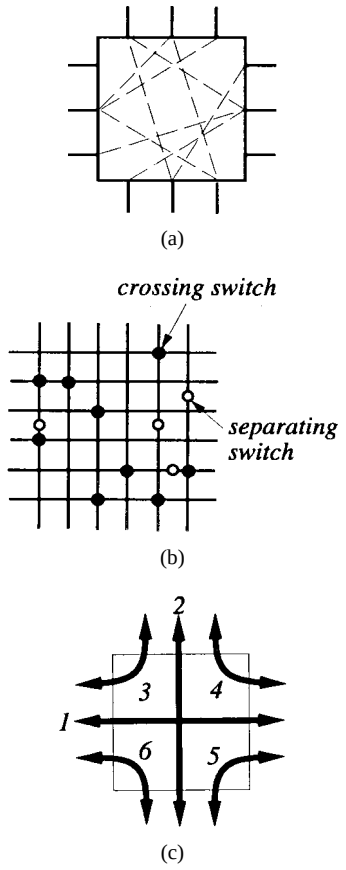


Fig. 3. Models for switch module. (a) Switch Block. (b) Switch Matrix. (c) Six types of connections.

module and rrv, a *routing* is a set of connections which are electrically noninteracting such that there are n_i of Type i connections, for $i \in \{1, \cdots, 6\}$. Note that a set of connections are electrically noninteracting only if the terminals on any two paths are distinct. In case of the switch matrices, for the set of connections to be electrically noninteracting, it is additionally required that the paths be disjoint, that is, no two paths share parts of a track. For switch matrices, notice the following role of separating switches. By setting the separating switch on the horizontal track i to off, the track to the left is electrically disconnected from the track to the right. Therefore, these two segments can be part of two different connections in any routing. An rrv n is said to be *routable* on a switch module S, if there exists a routing for n on S.

Example 2: In Fig. 4 a switch matrix and the routing for the $rrv(1,1,1,0,0,1)$ on this switch matrix are shown. The $rrv(0,0,2,0,0,0)$ is not routable on the same switch matrix as only the two crossing switches on vertical track 2 can be used for a Type 3 routing, and both cannot be used simultaneously.

We consider the following problems.

Routing Decision Problem (RDP): Given a switch module S (either a switch matrix or a switch block) and an rrv n, is n routable on S?

Routing Solution Problem (RSP): Given a switch module S (either a switch matrix or a switch block) and an rrv n, determine a routing for n on S, if any.

For convenience, we often refer to these problems as simply RDP with n or RSP with n, omitting the input S.

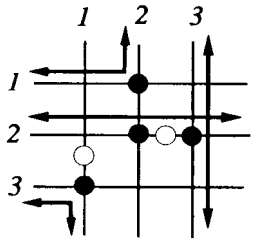


Fig. 4. Example of routing.

III. Integer Programming Formulation

In this section we solve the RDP using an ILP. The solution to the RSP is obtained from the solution to the ILP. We show our formulations for switch matrices and switch blocks separately.

A. Switch Matrix

Consider the RDP with the $rrv(n_1, \cdots, n_6)$ and switch matrix M. We formulate this problem as an ILP. In the resultant ILP, there are two main sets of constraints. The first set contains at most two constraints for each horizontal or vertical track. For each horizontal track one constraint ensures that the segment of the track to the left of the separating switch, if any, is part of at most one connection. The other constraint ensures this for the segment to the right of the separating switch. Similarly, at most two constraints are generated for each vertical track. Note that if a track does not contain a separating switch, then only one constraint is generated for this track. A set of constraints and an objective function are generated to ensure that a maximum number of connections specified by the rrv are routed in the solution of the ILP. We introduce some notation to succinctly describe the ILP.

Let S_1, S_2, S_3, and S_4 be four *constant* matrices defined as follows:

$$(S_1)_{ij} = \begin{cases} 1, & \text{if a crossing switch is found between horizontal track } i \text{ and vertical track } j \text{ such that a separating switch, if any, in this horizontal track is to the right of this crossing switch} \\ 0, & \text{otherwise} \end{cases}$$

$i = 1, \cdots, W_1, j = 1, \cdots, W_2$. S_2 is similarly constructed as an indicator matrix of the crossing switches to the right of separating switches in the horizontal tracks. Likewise, S_3 (S_4) is an indicator matrix of the crossing switches above (below) the separating switches in the vertical tracks.

Four $W_1 \times W_2$ *variable* matrices are as follows:

$$(X_k)_{ij}, = \begin{cases} (x_k)_{ij}, & \text{if } M \text{ has a crossing switch between horizontal track } i \text{ and vertical track } j \text{ and this can be used to achieve a connection of Type } k \\ 0, & \text{otherwise} \end{cases}$$

$k = 3, 4, 5, 6, i = 1, \cdots, W_1, j = 1, \cdots, W_2$. Variable $(x_k)_{ij}$ is an indicator variable that indicates if the switch between horizontal track i and vertical track j is utilized in a connection

Problem ILP1.

$$\max \quad e_1^T r + e_2^T c + \sum_{k=3}^{6} e_1^T X_k e_2$$

$$(S_1 \otimes \sum_{i=3}^{6} X_i) e_2 + r \leq e_1 \tag{1}$$

$$(S_2 \otimes \sum_{i=3}^{6} X_i) e_2 + r \leq e_1 \tag{2}$$

$$(S_3 \otimes \sum_{i=3}^{6} X_i)^T e_1 + c \leq e_2 \tag{3}$$

$$(S_4 \otimes \sum_{i=3}^{6} X_i)^T e_1 + c \leq e_2 \tag{4}$$

$$e_1^T r \leq n_1 \tag{5}$$

$$e_2^T c \leq n_2 \tag{6}$$

$$e_1^T X_k e_2 \leq n_k, \quad k = 3, \ldots, 6 \tag{7}$$

$$r \in \{0,1\}^{W_1}, c \in \{0,1\}^{W_2}, X_k \in \{0,1\}^{W_1 \times W_2}, \quad k = 3, \ldots, 6$$

Fig. 5. ILP formulation for switch matrix.

of Type k. Note that not every switch can be used for every type of connection. For example, if a crossing switch is above a separating switch for that column, then the switch cannot be used to realize a connection of Type 6. In Fig. 3(b), the crossing switch in row 2, column 1 can only route connections of the Types 3 and 4 and not of Types 5 and 6. The one in row 4, column 1 can route connections of Types 5 and 6 and not of Types 3 and 4.

Define the binary operator $\otimes$ on matrices as the component-wise multiplication, i.e., $(A \otimes B)_{ij} = A_{ij} B_{ij}$. Thus, each entry in $A \otimes B$ is the product of the corresponding entries in A and B.

Define a *variable* column vector r of dimension W_1 as

$$r_i = \begin{cases} 1, & \text{if horizontal track } i \text{ is used in a connection of Type 1} \\ 0, & \text{otherwise} \end{cases}$$

for $i = 1, \cdots, W_1$.

Define a *variable* column vector c of dimension W_2 as

$$c_j = \begin{cases} 1, & \text{if vertical track } j \text{ is used in a connection of Type 2} \\ 0, & \text{otherwise} \end{cases}$$

for $j = 1, \cdots, W_2$.

Let e_1 and e_2 be two constant column vectors of dimensions W_1 and W_2, respectively, with all components one. The integer programming formulation is shown in Fig. 7. In Fig. 5, inequality (1) ensures that the track to the left of the separating switch on each row, if any, are part of at most one connection. Inequality (2) does the same for the track to the right of the separating switch. Similarly, inequalities (3) and (4) ensure that the tracks above and below the separating switch on each column, if any, are part of at most one connection. The objective function, together with the last three inequalities, guarantees that the $rrv(n_1, \cdots, n_6)$ is routable if the maximum value of the objective function is $\sum_{i=1}^{6} n_i$. This will be shown in Theorem 1. The number of variables $\leq W_1 + W_2 + 4$ (number of switches) $\leq 4W_1W_2 + W_1 + W_2$ and the number of constraints $\leq 2(W_1 + W_2) + 6 + W_1 + W_2 + 4$ (number of switches) $\leq 4W_1W_2 + 3(W_1 + W_2) + 6$.

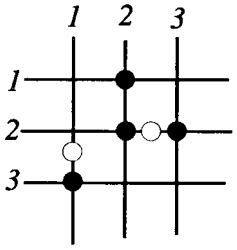


Fig. 6. Switch matrix.

Theorem 1: The problem ILP1 has a solution with objective value $\sum_{i=1}^{6} n_i$ if and only if the $rrv(n_1, \cdots, n_6)$ is routable on M.

Proof Sketch: If ILP1 has a solution with objective value $\sum_{i=1}^{6} n_i$, then constraints (5)–(7) are satisfied with equality. Since the problem has a solution, the "on" variables give a way to route $(n_1, \cdots, n_6)$ using the convention established before for naming variables. Constraints (1)–(4) ensure that the routing thus generated is valid.

Similarly, if $(n_1, \cdots, n_6)$ is routable, then there exists an assignment to variables such that the constraints (1)–(4) are satisfied and (5)–(7) are satisfied with equality. Hence, the value of the objective function is $\sum_{i=1}^{6} n_i$. □

Example 3: Consider the switch matrix in Fig. 6. Fig. 7 shows a set of important constraints in the corresponding ILP.

B. Switch Block

Consider an RDP with $rrv(n_1, \cdots, n_6)$ and switch block B. We write an ILP for the corresponding RDP. We have two sets

$$
\begin{aligned}
(x_3)_{12} + (x_4)_{12} + (x_5)_{12} + (x_6)_{12} + r_1 \le 1 \qquad & r_1 + r_2 + r_3 \le n_1 \\
(x_3)_{22} + (x_6)_{22} + r_2 \le 1 \qquad & c_1 + c_2 + c_3 \le n_2 \\
(x_4)_{23} + (x_5)_{23} + r_2 \le 1 \qquad & (x_3)_{12} + (x_3)_{22} \le n_3 \\
(x_5)_{31} + (x_6)_{31} + r_3 \le 1 \qquad & (x_4)_{12} + (x_4)_{23} \le n_4 \\
(x_5)_{31} + (x_6)_{31} + c_1 \le 1 \qquad & (x_5)_{12} + (x_5)_{23} + (x_5)_{31} \le n_5 \\
(x_3)_{12} + (x_4)_{12} + (x_5)_{12} + (x_6)_{12} + (x_3)_{22} + (x_6)_{22} + c_2 \le 1 \qquad & (x_6)_{12} + (x_6)_{22} + (x_6)_{31} \le n_6 \\
(x_4)_{23} + (x_5)_{23} + c_3 \le 1 &
\end{aligned}
$$

Fig. 7. Example of ILP1 (important constraints).

Problem ILP2.

$$
\begin{aligned}
\max \quad & \sum_{\{i,j\} \in N} x_{\{i,j\}} & & \\
& \sum_{\substack{j \\ \{i,j\}}} x_{\{i,j\}} \le 1, \quad \text{for each } i = 1, \ldots, 2(W_1 + W_2) & & (8) \\
& \sum_{\{i,j\} \in N, i \in t_1, j \in t_3} x_{\{i,j\}} \le n_1 & & (9) \\
& \sum_{\{i,j\} \in N, i \in t_2, j \in t_4} x_{\{i,j\}} \le n_2 & & (10) \\
& \sum_{\{i,j\} \in N, i \in t_1, j \in t_2} x_{\{i,j\}} \le n_3 & & (11) \\
& \sum_{\{i,j\} \in N, i \in t_2, j \in t_3} x_{\{i,j\}} \le n_4 & & (12) \\
& \sum_{\{i,j\} \in N, i \in t_3, j \in t_4} x_{\{i,j\}} \le n_5 & & (13) \\
& \sum_{\{i,j\} \in N, i \in t_4, j \in t_1} x_{\{i,j\}} \le n_6 & & (14) \\
x_{\{i,j\}} \in \{0,1\} \quad & \{i,j\} \in N & &
\end{aligned}
$$

Fig. 8. ILP formulation for switch block.

of inequalities. One set of inequalities is used to ensure that every terminal is used at most once. A 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 $1, 2, \cdots, 2(W_1 + W_2)$ starting from the lower most terminal on the left face and proceeding clockwise. The programmable links are specified by *sets* containing pairs of the terminals they connect. The terminals of a given connection come from different faces, as stated before. Let $N = \{\{i,j\}|$ there exists a programmable link between terminals $\{i,j\}\}$. Let $t_1 = \{1, 2, \cdots, W_1\}$, $t_2 = \{W_1+1, \cdots, W_1+W_2\}$, $t_3 = \{W_1+W_2+1, \cdots, 2W_1+W_2\}$, and $t_4 = \{2W_1 + W_2 + 1, \cdots, 2(W_1 + W_2)\}$. These sets identify the terminals of each of the four faces of the switch block. Define a variable $x_{\{i,j\}}$ for each programmable link $\{i,j\} \in N$. This is a decision variable that is chosen to be one, if the corresponding connection is chosen for the routing, else it is zero. The integer linear program is shown in Fig. 14. The number of variables $= |N|$ and number of constraints $= |N| + 2(W_1 + W_2) + 6$.

Theorem 2: The problem ILP2 has a solution with objective value $\sum_{i=1}^{6} n_i$ if and only if the $rrv(n_1, \cdots, n_6)$ is routable on B.

Proof Sketch: If ILP2 has a solution with objective value $\sum_{i=1}^{6} n_i$, then constraints (9)–(14), shown in Fig. 8, are satisfied with equality. Since the problem has a solution, the "on" variables give a way to route $(n_1, \cdots, n_6)$ using the convention established before for naming variables. The first constraint ensures that the routing thus generated is valid, i.e., each terminal is used in at most one connection.

Similarly, if $(n_1, \cdots, n_6)$ is routable, then there exists an assignment to variables such that the first constraint is satisfied and (9)–(14) are satisfied with equality. Hence, the value of the objective function is $\sum_{i=1}^{6} n_i$. □

Example 4: A switch block and the corresponding set of important constraints are shown in Figs. 9 and 10, respectively.

IV. Special Cases

Since the integer-programming problem is NP-complete [7], polynomial time algorithms are not known for RDP and RSP

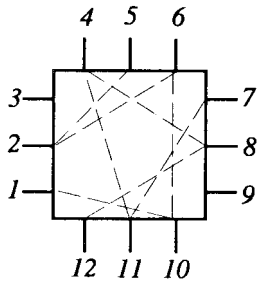


Fig. 9. Switch block.

$$x_{\{2,5\}} + x_{\{2,6\}} \le 1 \qquad x_{\{4,11\}} + x_{\{6,10\}} \le n_2$$
$$x_{\{4,11\}} + x_{\{4,8\}} \le 1 \qquad x_{\{2,5\}} + x_{\{2,6\}} \le n_3$$
$$x_{\{6,2\}} + x_{\{6,10\}} \le 1 \qquad x_{\{8,4\}} \le n_4$$
$$x_{\{8,4\}} + x_{\{8,12\}} \le 1 \qquad x_{\{7,11\}} + x_{\{8,12\}} \le n_5$$
$$x_{\{10,1\}} + x_{\{10,6\}} \le 1 \qquad x_{\{10,1\}} \le n_6$$
$$x_{\{11,4\}} + x_{\{11,7\}} \le 1$$

Fig. 10. Example of ILP2 (important constraints).

using the approach in Section III. In this section we identify several interesting classes of switch modules for which RDP and RSP can be solved in polynomial time.

Again, for convenience, the cases of switch blocks and switch matrices are considered separately. In what follows, we consider solving the RDP. The solutions to the corresponding RSP's are directly obtained from the proposed solutions to the RDP.

Define a *generic* rrv to be a sextuple in which each component is either a number or a special symbol "$*$." Any generic rrv represents the class of all rrv's which differ only in the components marked "$*$."

Example 5: The vector $(*,0,0,0,0,*)$ is a generic rrv. The vector $(*,0,0,0,0,*)$, represents the class containing all rrv's which have zeros in components 2–5. They may have zeros in components 1 and 6 but that is not necessary. Examples of rrv's in this class include $(10,0,0,0,0,6)$ and $(0,0,0,0,0,6)$.

In what follows, RDP (or RSP) with generic rvv stands for the problem of RDP on any rrv in the class of rrv's represented by v.

A. Switch Matrix

Case A—No Separating Switches: Suppose that the given switch matrix M contains no separating switches. We characterize the complexity of routing on M in terms of the complexity of the *bipartite-matching* problem. The bipartite-matching problem is to determine if a given bipartite graph has a matching of size k [12].

Let $\mathcal{P}1$ and $\mathcal{P}2$ be two problems. We denote $\mathcal{P}1 \Rightarrow \mathcal{P}2$ if $\mathcal{P}2$ reduces to $\mathcal{P}1$, that is, an efficient algorithm for problem $\mathcal{P}1$ yields an efficient algorithm for $\mathcal{P}2$.[1]

Lemma 1: RDP with $(0,0,*,0,0,0) \Rightarrow$ RDP with $(*,*,*,0,0,0)$.

Proof: Consider the instance of RDP on M with $rrv(n_1,n_2,n_3,0,0,0)$. If $n_1+n_3 > W_1$ or $n_2+n_3 > W_2$ the problem is trivially infeasible. Otherwise $(n_1,n_2,n_3,0,0,0)$ is routable on M if and only if $(0,0,n_3,0,0,0)$ is routable on M. This is because, given a routing for $(0,0,n_3,0,0,0)$, we can generate a routing for $(n_1,n_2,n_3,0,0,0)$ by utilizing n_1 horizontal tracks and n_2 vertical tracks disjoint from the connections in the routing of $(0,0,n_3,0,0,0)$. □

Lemma 2: RDP with $(*,*,*,0,0,0) \Rightarrow$ RDP with $(*,*,*,*,*,*)$.

Proof: We claim that $rrv(n_1,n_2,n_3,n_4,n_5,n_6)$ is routable on a switch matrix M if $rrv(n_1,n_2,n_3+n_4+n_5+n_6,0,0,0)$ is routable. We observe that any connection from left to top, in the absence of separating switches, renders the corresponding horizontal and vertical tracks unusable for further connections. Indeed this observation holds for all bent connections. Therefore, a left to top connection can be replaced by any other bent connection without causing any conflicts with the other connections in routing. In particular, if $n_3+n_4+n_5+n_6$ of Type 3 connections are possible then we can always replace these with n_i connections of Type $i, i=3,\cdots,6$. Therefore, if $(n_1,n_2,n_3+n_4+n_5+n_6,0,0,0)$ is routable, so is $(n_1,n_2,n_3,n_4,n_5,n_6)$. □

Lemma 3: RDP with $(0,0,*,0,0,0) \Leftrightarrow$ bipartite-matching problem.

Proof: Consider the instance of RDP with $rrv(0,0,n_3,0,0,0)$. We construct a bipartite graph as follows. Let $V = U \cup W$ be the vertex set where U has a vertex for each terminal on the left face of the switch module and W one for each terminal on the top. Thus, $|U| = W_1, |W| = W_2$. For every $u_i \in U, w_j \in W$ let $(u_i,w_j) \in E$ if and only if there is a crossing switch between the horizontal track i and vertical track j. Then, we claim that, $G=(V,E)$ has a bipartite matching of size n_3 if and only if the $rrv(0,0,n_3,0,0,0)$ is routable.

If the rrv is routable then n_3 crossing switches have been used in the routing. Choose the corresponding edges in E to give a matching of size n_3. Routability criterion ensure that no row or column is used more than once.

If there is a bipartite matching of size n_3 in G then choose the switches corresponding to the edges in the matching to route n_3 connections from the left to the top in the switch module.

To prove the reduction in the other direction, given a bipartite graph $G=(V,E), V=U\cup W, U\cap W=\phi$, construct a $|U| \times |W|$ switching matrix M as follows: place a crossing switch between horizontal track i and vertical track j if and only if $(u_i,w_j) \in E$. We claim that G has a matching of size k if and only if the $rrv(0,0,k,0,0,0)$ is routable on M. The proof is along the lines sketched above; it is omitted here. □

Example 6: A switch matrix with the bipartite graph in the transformation of RDP with $(0,0,*,0,0,0)$ is shown in Fig. 11. For any integer k, the $rrv(0,0,k,0,0,0)$ can be

[1] Formally, we say $\mathcal{P}1 \Rightarrow \mathcal{P}2$ if an instance of problem $\mathcal{P}2$ can be reduced to an instance of problem $\mathcal{P}1$ in time $O(W_1+W_2+|T|)$ where T is the set of crossing switches in M, and M is the switch matrix in one of the problems $\mathcal{P}1$ or $\mathcal{P}2$.

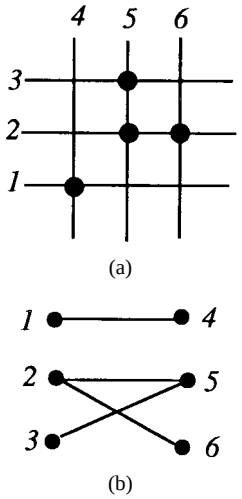


Fig. 11. No separating switches. (a) Switch matrix. (b) Bipartite graph.

routed on the switch matrix in Fig. 11(a) if and only if the graph in Fig. 11(b) has a matching of size k. It is easy to see that any $k \leq 3$ yields a routable rrv.

Now we are ready to prove the following theorem.

Theorem 3: RDP with $(*,*,*,*,*,*) \Leftrightarrow$ bipartite-matching problem.

Proof: Trivially, RDP with $(*,*,*,*,*,*) \Rightarrow$ RDP with $(*,*,*,0,0,0) \Rightarrow$ RDP with $(0,0,*,0,0,0)$. Combined with Lemmas 1, 2, and 3, this yields the theorem. □

The bipartite-matching problem can be solved in time $O(\sqrt{|V|}|E|)$ for a bipartite graph $G(V,E)$ [12]. From Theorem 3, it follows that RDP for a switch matrix M with no separating switches can be solved in time $O(\sqrt{(W_1+W_2)}|T|)$, where $|T|$ is the number of crossing switches in M. In fact, Theorem 3 implies something stronger: any algorithm for solving RDP on M which is faster than time $O(\sqrt{(W_1+W_2)}|T|)$ immediately yields an algorithm for the bipartite-matching problem which is faster than $O(\sqrt{|V|}|E|)$. Note that the existence of such an algorithm for the bipartite-matching problem is a long-standing open problem. Therefore, improving the time bound of $O(\sqrt{(W_1+W_2)}|T|)$, for routing on M with no separating switches, is an extremely hard problem.

Let M_1 be a switch matrix without separating switches such that the corresponding bipartite graph (see proof of Lemma 3) has a perfect matching. Let M_2 be any other switch matrix without separating switches. The following corollary of Theorem 3 is easy to see.

Corollary 1: An $rrvn$ is routable on M_2 only if it can be routed on M_1.

This corollary asserts that a switch matrix M_1 is the most "powerful" in the class of switch matrices without separating switches. This means that, given W_1, W_2, any rrv that can be routed on some switch matrix without separating switches can be routed on M_1. Thus, if the number of crossing switches is taken as a measure of complexity, then designing a switch matrix, for which the corresponding bipartite graph does not have a perfect matching, is, in general, a waste of resource. This fact can be used in the design of a good switch matrix without separating switches.

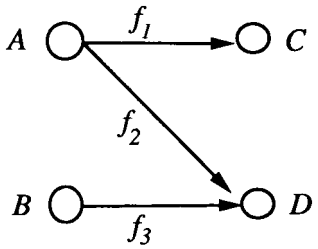


Fig. 12. Noninterfering network.

Case B—Without Separating Switches in Horizontal or Vertical Tracks: Consider RDP or RSP with $(*,0,*,*,*,*)$. Assume that in the switch matrix for these problems, the horizontal tracks do not contain any separating switches. Similar results hold for the case where vertical tracks do not have separating switches. Under this condition, it is easy to see using the technique in the proof of Lemma 2 that $(n_1,0,n_3,n_4,n_5,n_6)$ is routable if and only if $(n_1,0,n_3+n_4,0,0,n_5+n_6)$ is routable. The RDP with generic $rrv(*,0,*,0,0,*)$ is known to be solvable using unit-capacity network flows [21]. It follows that under the given condition, RDP or RSP with $(n_1,0,n_3+n_4,0,0,n_5+n_6)$ is solvable as well, using unit-network capacity network flows.

Case C—Class of Problems Solvable by Network Flows: Consider the following problem which we call the *noninterfering network-flow* problem, shown in Fig. 12.

Consider a directed network with four blocks of nodes, namely, A, B, C, and D. In addition there exist special nodes s_1, s_2 and t_1, t_2, respectively, the pair of source nodes and the pair of sink nodes. Arcs between nodes in the blocks, if any, exist between nodes in block A and C or A and D, or between nodes in block B and D. In particular, there are no arcs between nodes in the same block. The source s_1 (s_2) is connected to each node in A (B). Every node in C (D) is connected to the sink t_1 (t_2). Each arc has capacity one. The *noninterfering network-flow problem* is the following. Given such a network, and integers f_1, f_2, and f_3, does there exist a feasible flow such that source s_1 supplies a flow of f_1+f_2, source s_2 supplies a flow of f_3, sink t_1 receives a flow of f_1, and sink t_2 receives a flow of f_2+f_3? It is easy to see that such a flow exists if and only if there is a matching between the vertex sets $A \cup B$ and $C \cup D$ such that there exist exactly f_1 arcs between nodes in A and C, exactly f_2 arcs between nodes in A and D, and exactly f_3 arcs between nodes in B and D.

Following are two categories of RDP's which can be solved using a transformation to the noninterfering network-flow problem. In what follows, the switch matrix is assumed to have the following property. Each horizontal and vertical track of the matrix has precisely one separating switch.

1) RDP's with generic rrv in which the components corresponding to any three bent connections are marked "$*$," and the remaining components are zero. For example, RDP with $(0,0,*,0,*,*)$.
2) RDP's with rrv's in which the components corresponding to any two bent connections which do not share a face of the switch matrix are marked "$*$," and the component of any one straight connection is marked "$*$."

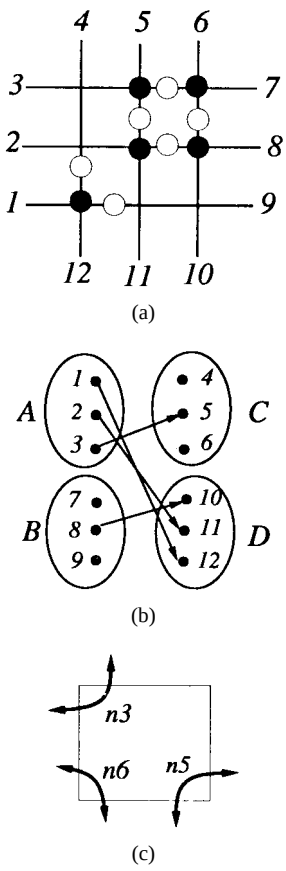


Fig. 13. Example of a transformation into the noninterfering network-flow problem. (a) Switch matrix. (b) Network. (c) Requirements.

The remaining components are zero. For example, RDP with $(*, 0, *, 0, *, 0)$.

We now sketch the transformations from problems listed above to noninterfering network-flow problems.

Consider an example of a problem in Category 2 above, for example, RDP with $(0, 0, n_3, 0, n_5, n_6)$. We create a node for every terminal of the switch matrix. The nodes of the left face form block A, those on the right form B, those on the top form C, and those on the bottom form D. For a crossing switch which is found between the horizontal track i and vertical track j, create an edge from the node in $A(B)$ to the one in $C(D)$ corresponding to the terminals i and j. It is crucial to note that since each horizontal and vertical track has precisely one separating switch, a crossing switch can be utilized in precisely one bent connection. It is now easy to observe that $(0, 0, n_3, 0, n_5, n_6)$ is routable if and only if there is a matching between the vertex sets $A \cup B$ and $C \cup D$ such that there exist exactly f_1 arcs between nodes in A and C, exactly f_2 arcs between nodes in A and D, and exactly f_3 arcs between nodes in B and D. Thus, RDP with $(0, 0, n_3, 0, n_5, n_6)$ is transformed to the noninterfering network-flow problem.

Example 7: Fig. 13 gives an example of this transformation. The sources and sinks are omitted for clarity.

A similar construction suffices for transforming a problem in Category 2 above to the noninterfering network-flow problem.

Using standard techniques for computing the max-flow in networks [9], the noninterfering network-flow problem on a network $G(V, E)$ can be solved in time $O(|V||E| \log |V|)$ [16]. Therefore, problems in Categories 1 and 2 above can be solved in time $O((W_1 + W_2)|T| \log(W_1 + W_2))$.

B. Switch Block

The problem of routing in switch blocks is, in some sense, simpler than that on switch matrices. This is because connections can interfere with each other if and only if they share a terminal. In the case of a switch matrix, they could additionally interfere if the connections shared a part of a track.

We show a few special cases of routing on switch blocks that have polynomial-time algorithms. The explanations are similar to the corresponding switch matrix cases, and we just give brief ideas about the algorithms or reductions.

Case A—Problems Solvable by Flows in Noninterfering Networks: The concept of noninterfering networks has been introduced earlier in Section IV-A, Case C. The category of rrv's that can be routed on switch blocks using these network-flow techniques is the same as those enumerated in Section IV-A, Case C.

The only difference in the transformation is that only arcs corresponding to relevant programmable links are considered. For example, for the RDP with $(0, 0, n_3, 0, n_5, n_6)$, only arcs corresponding to left to top, right to bottom, and left to bottom programmable links are drawn.

Example 8: An illustration of the above transformation is shown in Fig. 14. The sources and sinks are omitted for clarity.

Case B—Problems Solvable by Single Source Network Flows: Consider the case in which the nonzero routing requirements in the rrv share a common face of the switch block. An example is RDP with generic $rrv(*, 0, *, 0, 0, *)$. RDP with such rrv's can be transformed to a single source network flow problem. We show how to do this for the RDP with $rrv(n_1, 0, n_3, 0, 0, n_6)$. There is one node for each terminal. Call the block of nodes corresponding to the left, top, right, and bottom faces A, B, C, and D, respectively. There are four special nodes, a source s and three sinks t_1, t_2, and t_3. For each node in A, there is an arc to a node in B if there is a programmable link between the corresponding terminals. Similarly, there are arcs from the nodes in A to those in C and D. There is an arc from s to every node in A and from each node in B, C, and D to t_1, t_2, and t_3, respectively. All arcs have capacity one. The problem now is that given such a network, is there a feasible flow where s supplies a flow of $n_1 + n_3 + n_6$ and t_1, t_2, and t_3 receive flows of n_1, n_3, and n_6, respectively? This can be solved by network-flow algorithms in time $O((W_1 + W_2)|N| \log(W_1 + W_2))$ where N is the set of programmable links as in Section III-B [16].

Example 9: An example of this transformation is shown in Fig. 15. The sources and sinks are omitted for clarity.

V. Minimal Dominating Set

For this section, *fix* a switch module S. Consider solving either RDP or RSP on S for various rrv's. Using our algorithm in Section III, an instance of integer programming problem is solved for each rrv. In this section, we describe a precomputation on S so that following this precomputation, either

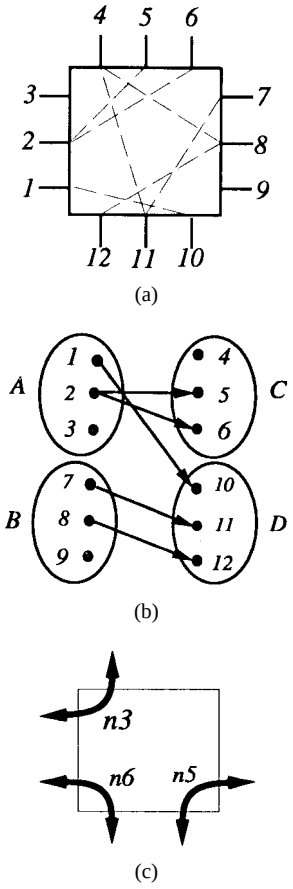


Fig. 14. Example noninterfering network-flow transformation for switch block. (a) Switch block. (b) Network. (c) Requirements.

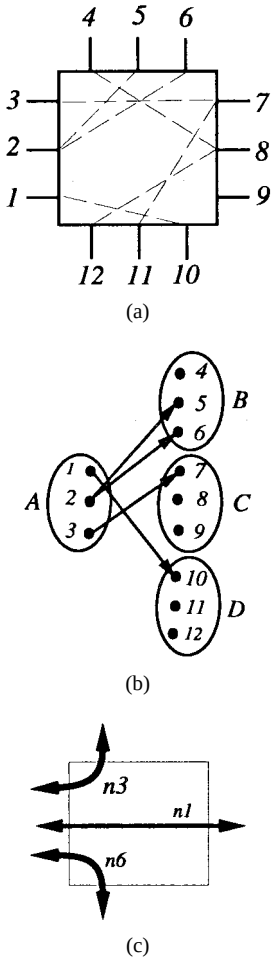


Fig. 15. Example of a transformation into single source network-flow problem. (a) Switch block. (b) Network. (c) Requirements.

RDP or RSP on S can be solved for any given rrv without resorting to the integer programming problem. For a given S, a set of routing requirement vectors are identified during the precomputation. This involves solving several integer programs. Following this computation, RDP or RSP on any given rrv can be solved fast by comparing it 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 are now described. First consider solving RDP.

An $rrv(n_1, \cdots, n_6)$ is said to *dominate* another $rrv(m_1, \cdots, m_6)$ if and only if $n_i \geq m_i,\ i = 1, \cdots, 6$ and for some i, $n_i > m_i$. It is a simple observation that any $rrv\ m$ is routable if another $rrv\ n$ is routable on S, and n dominates 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 R of rrv's is called a *dominating set* for a switch module S, if for an $rrv\ v$, v is routable on S if and only if either $v \in R$, or there exists an $rrv\ w \in R$ such that w dominates v. A dominating set R for S is called *minimal* if $\forall v, w \in R$ neither v dominates w nor w dominates v. The following property is crucial.

Lemma 4: The minimal dominating set for a switch module S is unique.

Proof: Suppose for contradiction that R, T are two distinct minimal dominating sets for S. Consider the case when $R \not\subset T$. In this case there exists a v such that $v \in R$ and $v \notin T$. Since T is a dominating set, there exists a $w \in T$ that dominates v. Since R is a dominating set as well, either $w \in R$, or for some $u \in R$, u dominates w. In either case, there exists an rrv in R that dominates v. This contradicts the assumption that R is minimal. Similarly it can be shown that if $R \subset T$ then T cannot be minimal. □

Observe that the set of routable rrv's for S is partially ordered under dominance relation. An $rrv\ v$ is called a *top element* if it is routable and there exists no other rrv that dominates v. The following lemma is the key in computing the minimal dominating set for S.

Lemma 5: Let $\Gamma = \{v |\ v$ is a top element $\}$. Then Γ is the minimal dominating set.

Proof: Γ is a dominating set since for any rrv which is not a top element, there exists a top element that dominates it. Also, if v and w are two top elements, v does not dominate w, and w does not dominate v. Therefore, Γ is minimal. By Lemma 4, Γ is the unique minimal dominating set. □

Let $W = \min\{W_1, W_2\}$. Let V be the set of rrv's for S. Define $L(\alpha) = \{v \in V | \sum_{i=1}^{6} v_i = \alpha\}$. An $rrv\ w$ is a *child* of $rrv\ v \in L(\alpha)$ if $w \in L(\alpha - 1)$, and if w differs from v in exactly one component. The $rrv\ v$ is called a *parent* of w. For $1 \leq j \leq 6$, the jth parent of w is the only parent of w that differs from it in the jth component. Thus, each rrv has

```
Algorithm: Minimal_Dominating_Set(S)
Input: S--switch-block specification.
Output: The minimal dominating set of S.

1   Insert (0,0,0,0,0,0) in L(0);
2   for i = 0 to W1 + W2 - 1
3       for each v ∈ L(i)
4           found = false;
5           for j = 1 to 6
6               if j^th parent of v is routable
7                   insert j^th parent in L(i+1);
8                   found = true;
9           if (not found) Output v;
10  Output each element of L(W1 + W2).
```

Fig. 16. Algorithm for computing the dominating set.

up to six parents and up to six children. Note that the rrv's $(0,0,0,0,0,0)$ and (W_1, W_2, W, W, W, W) have no children and no parents respectively.

We describe an algorithm to compute the minimal dominating set for a given switch module 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 III, 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 as top elements. Note that it is sufficient to stop the algorithm after level $W_1 + W_2$, since in succeeding levels, the rrv's satisfy the trivial infeasibility condition from Section II. From Lemma 5, it is easy to see that the set of top elements in the output of our algorithm is the minimal dominating set. The pseudocode is shown in Algorithm Minimal_Dominating_Set (see Fig. 16).

Computing the minimal dominating set R for S completes the precomputation. Following this, consider solving RDP with rrv v. Clearly, v is routable if and only if $v \in R$ or there exists some rrv in R which dominates v. This can be checked quickly by successively performing a binary search on the components of the sextuples in a straightforward manner. Note in particular that no integer programming problem needs to be solved.

To solve RSP, we modify the precomputation described above. Along with each rrv v determined to be in the minimal dominating set R, we determine and store the routing for v. Following this, RSP for any given rrv u can be solved fast. First determine if $u \in R$ or find an element w, if any, in R which dominates u. In the second case, it is easily seen that a routing for u, if any, can be generated from the routing for w if w exists. Again, no integer programming problem is solved in RSP.

To sum up, by precomputing the minimal dominating set R of S off-line, the need to solve an integer programming problem while solving RDP or RSP on-line is avoided.

VI. Global Routing

We now show how the minimal dominating set, whose computation has been described in the previous section, can be used in global routing. In this paper we shall limit ourselves to switch modules being switch blocks. A similar approach can be used for switch matrices. Our *demonstrative* algorithm will closely follow the maze-routing algorithm. A description of the maze-routing approach is given in [13]. We shall model the FPGA as a weighted graph. Paths in the graph will represent routes in the FPGA. The novelty of our approach lies in the way we compute the weights of the graph edges. For this we will propose a new metric that makes use of the minimal dominating set. This metric captures the constraints imposed by the limited switches available in the switch block. We assume that no jogs are used within switch blocks.

For simplicity, we shall assume that all switch modules in the FPGA are identical and that $W_1 = W_2 = W$. This is the case with most commercially available FPGA's. The techniques to be described can be easily generalized to avoid making these assumptions.

We first introduce two definitions. We define the *switch-block density* of a switch block S, denoted by d_S, as a vector $(m_1, m_2, \cdots, m_6)$, $m_i \geq 0, 1 \leq i \leq 6$, where m_i is the number of Type i connections currently routed through S. Let R denote the dominating set of each switch block in the FPGA. We define the set

$$D_S = \{v \in R \mid v \text{ dominates } d_S\}.$$

Thus, D_S is the set of rrv's in the dominating set R of S which dominate d_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 d_S and 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 routing channels. At the end of global routing, we say that a switch block S in the FPGA is *feasible* if d_S is routable on S.

A. 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. Fig. 17 illustrates the FPGA modeling. As shown in Fig. 17, each logic block or connection block is represented by a node, each routing channel is modeled as an edge called a *channel edge*, and each connection between a logic block and a connection block is modeled as an edge called a *connection 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 Fig. 17(b) and (c) for the modeling. Paths in the graph represent global routes on the FPGA and vice versa. Weights associated with edges represent congestion information. Henceforth, we shall denote the graph used to model an FPGA F by $G_F(V, E)$. The edge

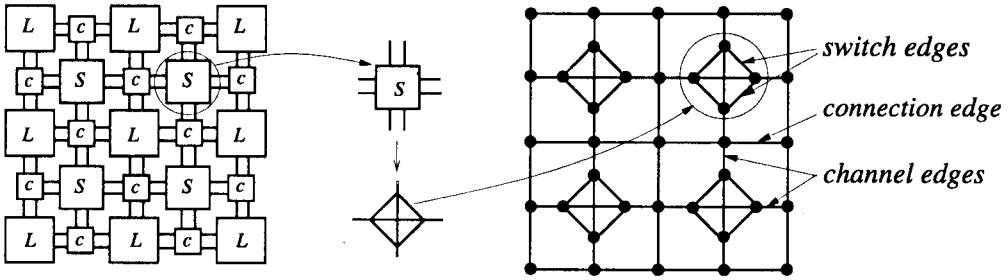


Fig. 17. The FPGA graph modeling. (a) A symmetrical-array FPGA architecture. (b) The switch-block modeling. (c) The FPGA modeling.

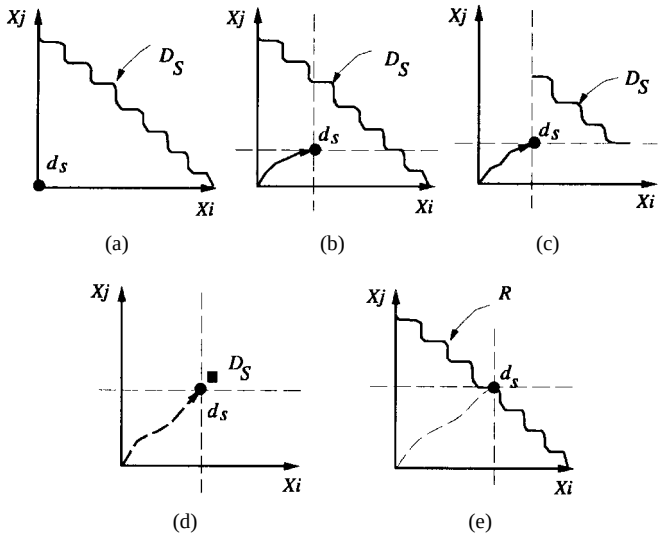


Fig. 18. Dynamically update congestion information during routing. An illustration in a two-dimensional plane with axes x_i and x_j. (a) The initial stage. (b) and (c) Update d_S and D_S. (d) Only one vector remains in D_S. (e) The state when $d_S \in R, D_S = \phi$.

Algorithm: *FPGA_Global_Routing*$(F, S, R, \mathcal{N})$

Input: F--FPGA architecture.
S--switch-block specification.
R--the minimal dominating set of S.
$\mathcal{N}$--netlist of 2-pin nets.

Output: The set of global routes P for $\mathcal{N}$ on F.

```
1  D_S ← R; d_S ← (0,0,0,0,0,0); P ← ∅;
2  Construct graph G_F to model F;
3  Assign initial weights to edges in G;
4  for each net N ∈ 𝒩
5      Find a minimum cost route r for N in G;
6      P ← P ∪ {r};
7      Update weights of the switch edges on the route,
           d_S, and D_S accordingly;
8      exit when there exists an infeasible switch block
9  Output P.
```

Fig. 19. Global-Routing Algorithm.

set E is partitioned into $\mathcal{C}$, the set of channel edges, $\mathcal{N}$, the set of connection edges, and $\mathcal{S}$, the set of switch edges.

B. The Global Routing Algorithm

The global router is based on a modified Dijkstra's shortest path algorithm [6]. Unlike the traditional global router which is guided by channel density, our FPGA global router is guided by switch-block density. The main goal is to evenly distribute the nets among routing 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. For a switch edge $e \in \mathcal{S}$, denote the switch block corresponding to e by S_e. Similarly, for a channel edge $e \in \mathcal{C}$, denote the routing channel corresponding to e by C_e.

The cost function $\alpha : E \to \Re^+ \cup \{0\}$ that guides the global routing is defined by

$$\alpha(e) = \begin{cases} \max_{n \in D_{S_e}} \left(\sum_{i=1}^{6} 1/c^{(n_i - m_i)}\right), & e \in \mathcal{S} \\ 0, & e \in \mathcal{C} \cup \mathcal{N} \end{cases}$$

where $c > 1$ is a constant. This cost function is used to weight each of edges in $G_F(V, E)$.

The whole routing procedure is illustrated in Fig. 18. Given an FPGA F, we first construct the graph G_F to model F. Initially, $d_S = (0, 0, 0, 0, 0, 0)$ for every switch block S in the FPGA. The cost of every edge in G_F is computed using the function α above. See Fig. 18(a) for the initial configuration. After a net is routed, d_S and D_S need to be updated to reflect the additional congestion resulting from the routing of the net. The weights associated with the edges on the route are recomputed using the updated d_S and D_S and the cost function shown above. See Fig. 18(b) and (c) for an illustration of the update. In Fig. 18(c), those rrv's which no longer dominate d_S are removed from D_S during the update. The process continues as routing proceeds. Notice that the cardinality of D_S monotonically decreases during the process. We assign a high cost to the switch edges corresponding to the switch block S when the set D_S is empty. Essentially, at this stage, $d_S \in R$, and hence, no more nets can be routed through S. This is graphically shown in Fig. 18(e). The last step ensures that a saturated switch block gets low priority while routing further nets. Algorithm FPGA_Global_Routing summarizes the process (see Fig. 19).

TABLE I
RUNNING TIMES FOR ILP METHOD

Module name	W	Maximum time	Average time	#Variables	#Constraints	#Nonnuls
module40	20	0.3	0.205	97	70	388
module50	20	0.5	0.220	115	70	460
module80	20	1.5	0.258	177	70	708
nm40	20	0.3	0.217	120	66	480
nm50	20	0.6	0.236	140	66	560
nm80	20	1.3	0.289	220	66	880

In contrast, the classical channel-density-based router will assign weights to the graph edges based on the following metric. A description of such a cost function is given in [11]. The cost function $\beta : E \rightarrow \Re^{+} \cup \{0\}$ that guides such a method is defined by

$$\beta(e) = \begin{cases} 1/c^{W-d_{C_e}}, & e \in \mathcal{C} \\ 0, & e \in \mathcal{S} \cup \mathcal{N} \end{cases}$$

where $c > 1$ is a constant, and d_{C_e} is the density in the channel C_e, corresponding to the channel edge e of the FPGA. The overall strategy is quite similar to the one in Algorithm FPGA_Global_Routing. The difference is that update steps need to update the values of d_{C_e} for each channel edge e along the newly routed path.

TABLE II
COMPARISON WITH APPROXIMATE ALGORITHM

		% of routable vectors	
Module name	W	Approximate algorithm	Exact algorithm
module40	20	65	58
module50	20	70	64
module80	20	94	82
nm40	20	99	76
nm50	20	99	83
nm80	20	99	90
Average	20	87.7	75.5

VII. EXPERIMENTAL RESULTS

Our experimental results fall into two parts. In Section VII-A, we demonstrate the improvement in solutions to the RDP and RSP. In Section VII-B, we show the effects of the two metrics α and β on routing.

A. Results of Using Exact Solutions for RDP and RSP

We wrote programs that take in routing problems and switch-module descriptions and generate integer programming problems as described in Sections III-A and III-B. We used a popular integer linear programming code called *lp_solve* that uses branch-and-bound techniques combined with the simplex algorithm for linear programming to generate integer solutions. We ran the program on a Sun Sparc 1 workstation. We tested the sizes of the problems and running times for both switch-matrix and switch-block models. The results are tabulated in Table I where the second column gives the size of the switch module ($W = W_1 = W_2$), the third gives maximum observed running time, and the fourth column gives the average running time over 100 experiments. The last three columns give an idea about the size of the ILP. In all cases the RDP was being solved. The fast running time of our algorithms makes our approach an attractive one to use in practice for evaluating designs of switch modules as well as for the application to global routing.

We also compared the routabilities of several switch modules as computed by our exact algorithm with those obtained by the approximate algorithm in [21]. This is shown in Table II. All experiments used 100 rrv's on the switch modules. The extent of overestimation that results from an approximate algorithm justifies the use of our algorithms. The approximate algorithm was off by about 16%, on an average.

We tested the technique mentioned in Section V. We observed a dramatically small search-space size, i.e., the cardinality of the minimal dominating set. For example, it was observed that for a 10 $\times$ 10 switch-matrix design the cardinality of the minimal dominating set was 1254 which is just 0.12% of the possible 10^6 possible rrv's. For a 15 $\times$ 15 switch block of the type to be used in the routing in the next section, the cardinality of the dominating set was 1368. As explained before, a binary search could be used on this set of vectors to test for the routability of a specified rrv.

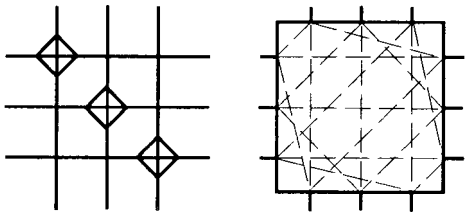
B. Routing Results

To explore the effects of the two congestion metrics α and β on routing, we implemented the global-routing algorithms described earlier and then integrated them into the CGE [4] and SEGA [14] detailed routers.

We tested the performance of the metrics on 14 industrial benchmark circuits used in [4] and [14]. As mentioned earlier, the new metric α uses switch-block capacity as a congestion control parameter while the traditional metric β is based on channel density. All benchmark circuits were first routed by the two global routers, one based on the metrics α and the other on β, using the same net ordering to obtain respective global routes. The global routes were

TABLE III
COMPARISON OF THE EFFECTS OF THE TWO METRICS α AND β ON ROUTING USING THE CGE ROUTER AND CIRCUITS. FROM LEFT TO RIGHT: NAMES OF CIRCUITS, SIZES OF FPGA (NUMBERS OF LOGIC MODULES IN THE FPGA'S), NUMBERS OF NETS IN THE CIRCUITS, NUMBERS OF EQUIVALENT 2-P-I-N CONNECTIONS, CHANNEL WIDTHS REQUIRED FOR GLOBALLY ROUTING ALL NETS KEEPING ALL SWITCH BLOCKS FEASIBLE USING THE GLOBAL-ROUTING ALGORITHMS PRESENTED EARLIER (GLOBAL ROUTING), AND CHANNEL WIDTHS REQUIRED FOR ROUTING COMPLETION USING THE CGE DETAILED ROUTER (DETAILED ROUTING)

CGE				Global routing (W)		Detailed routing (W)	
circuits	# Logic blocks	# Nets	# Connections	Metric α	Metric β	α	β
BNRE	22 × 21	352	1257	11	13	12	13
BUSC	13 × 12	151	392	9	11	10	11
DFSM	23 × 22	420	1422	11	14	12	15
DMA	18 × 16	213	771	10	15	11	15
Z03	27 × 26	608	2135	13	16	16	17
Total				54	69	61	71



(a) (b)

Fig. 20. (a) The switch-block architecture ($F_S = 3$). (b) The corresponding switch-block model.

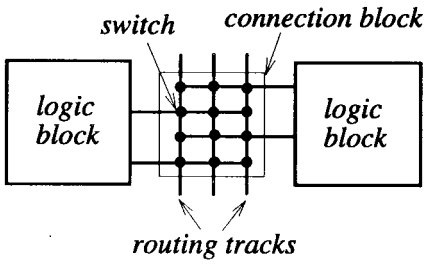


Fig. 21. The connection-block architecture ($F_C = W$, $W = 3$).

then fed into the CGE/SEGA detailed routers to determine final routing solutions. Notice that the most important concern in the experiment shall be the investigation of the effects of the two metrics. For the purpose of fair comparison, we kept our experimental factors simple. For instance, we used the shortest path-based algorithm to explore the effects, and no optimization such as rip-up and reroute was incorporated.

For FPGA's, the capacity of a channel is the size of the corresponding side of a switch block, W. In our experiments, we used the parameter $c = 2$. The FPGA architectures used in the routing based on the two metrics were identical. The switch block used was similar to that of Xilinx XC4000 series FPGA's [20]. We refer to the *flexibility* of a switch block S, denoted by F_S, as the number of programmable links connected to a terminal in S and that of a connection block, F_C, as the number of tracks that a logic-block pin can connect to [15]. For the architecture we used $F_S = 3$, and $F_C = W$. Figs. 20 and 21 illustrate the respective switch-block and connection-block architectures for the case $W = 3$.

We evaluated a metric based on the channel width W required for global- and detailed-routing completion by using the metric. Since smaller W implies the capability of routing a larger circuit on a given FPGA, a metric leading to a smaller W requirement for routing is desirable. As defined before, at the end of global routing, a switch block S is feasible if d_S is routable on S, i.e., if $d_S \in R$, or if there exists a $v \in R$ such that v dominates d_S. The columns "Global routing (W)" in Tables III and IV list the channel widths required for routing all the nets based on the metric α or β such that all switch blocks are feasible. The columns "Detailed routing (W)" give the channel widths required for routing completion, using the global routes generated from the corresponding metric. The results show that better global-routing topologies, in general, lead to better detailed-routing solutions, and the new metric α has better area performance than the traditional metric β. An average of 22% channel-width reductions on the 14 CGE/SEGA benchmarks is achieved. Fig. 22 shows the detailed-routing solution for the circuit example 2 with the parameters $W = 13$, $F_S = 3$, and $F_C = 13$, using the SEGA detailed router and the global routes generated by our new metric α.

VIII. CONCLUSIONS AND FUTURE WORK

In this paper, we described an integer programming approach to solving a routing problem on switch modules. The problem was originally proposed in [21] as an important part of their approach to switch-module design. Experimental results consistently showed that our algorithm was very efficient for practical-size switch modules. We also identified in this paper several special cases of the problem which reduce to well-known problems and to which polynomial-time algorithms are known.

The techniques proposed provide an efficient way of estimating congestion at switch modules which can be used in computing good global routes. We demonstrated the success of this scheme by showing that a substantial reduction of channel widths are required as compared to methods guided by channel capacity alone. We propose to extend this method to more general FPGA routing architectures, e.g., the one proposed in [10] and other global-routing approaches, e.g., the Steiner-tree formulation.

TABLE IV
COMPARISON OF THE EFFECTS OF THE TWO METRICS α AND β ON ROUTING USING THE SEGA ROUTER AND CIRCUITS. FROM LEFT TO RIGHT: SAME AS IN TABLE III, EXCEPT THAT THE SEGA ROUTER WAS USED FOR DETAILED ROUTING

SEGA				Global routing (W)		Detailed routing (W)	
circuits	# Logic blocks	# Nets	# Connections	Metric α	Metric β	α	β
9symml	11 × 10	79	259	8	9	9	9
alu2	15 × 13	153	512	9	12	10	13
alu4	19 × 17	255	852	12	17	14	18
apex7	12 × 10	115	301	11	14	11	14
example2	14 × 12	205	445	13	15	13	17
k2	22 × 20	404	1257	17	23	20	28
term1	10 × 9	88	203	9	11	10	12
too_large	14 × 14	186	520	11	15	12	15
vda	17 × 16	225	723	15	16	15	17
Total				105	132	114	143

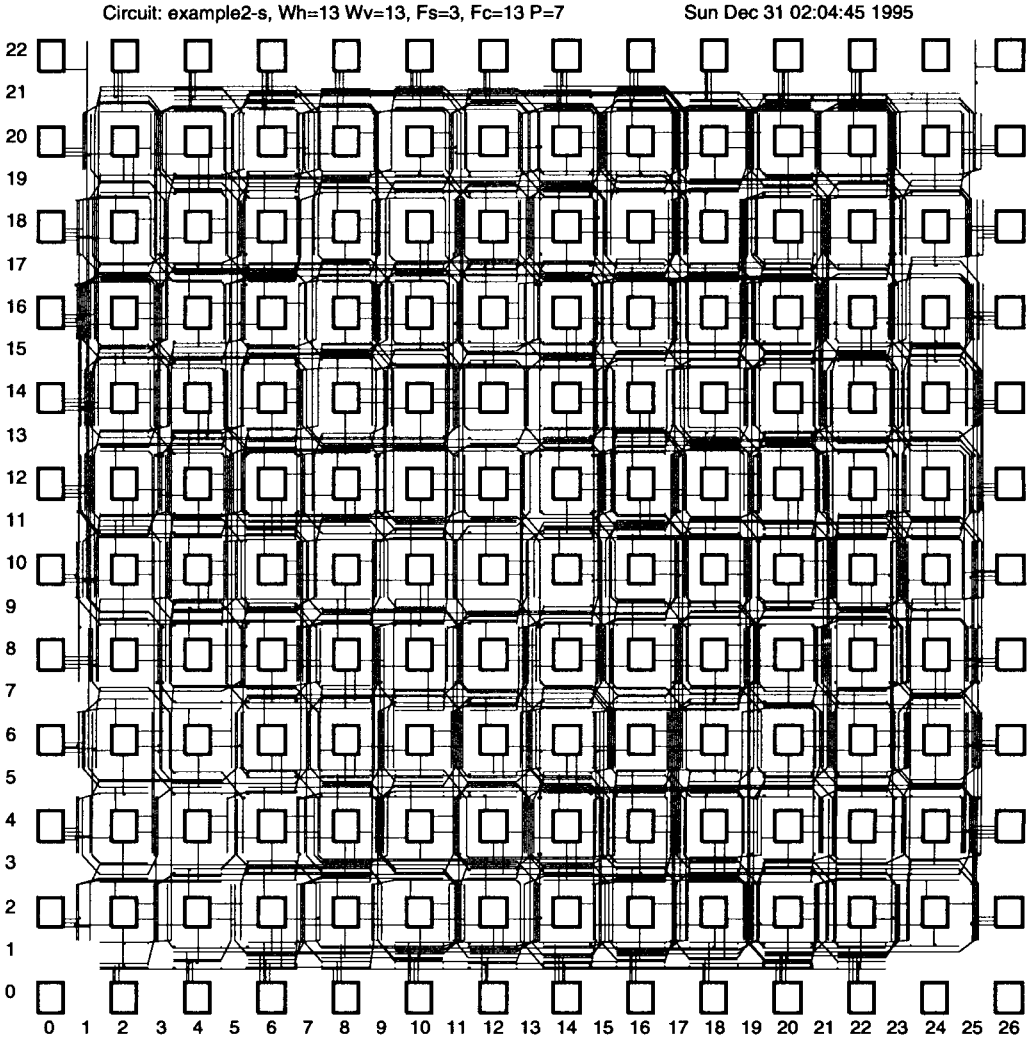


Fig. 22. The routing solution for the circuit example 2 with the parameters $W = 13$, $F_S = 3$, and $F_C = 13$, using the SEGA detailed router and the global routes generated by metric α.

The integer programming package we used was general, and we did not attempt to customize it to make use of the specific nature of the problem matrix. As can be seen from Table I, the problem matrix is quite sparse. Exploiting this would further speed up the solution process. Also, whether the routing problem RDP is NP-complete is still open.

ACKNOWLEDGMENT

The authors thank Prof. S. Brown of the University of Toronto for providing them with the CGE and SEGA packages. Also, the authors thank K. Zhu for helpful discussions on several topics in the paper and specifically on the formulation in Section III-B.

REFERENCES

[1] *Optimized Reconfigurable Cell Array (ORCA) Series Field-Programmable Gate Arrays*. AT&T Microelectronics, Advance Data Sheet, Feb., 1993.
[2] N. Bhat and D. Hill, "Routable technology mapping for LUT FPGA's," in *Proc. Int. Conf. Computer Design, VLSI Computers Processors*, 1992, pp. 95–98.
[3] S. Brown, R. J. Francis, J. Rose, and Z. G. Vranesic, *Field-Programmable Gate Arrays*. Norwell, MA: Kluwer, 1992.
[4] S. Brown, J. Rose, and Z. G. Vranesic, "A detailed router for field-programmable gate arrays," *IEEE Trans. Computer-Aided Design*, vol. 11, pp. 620–627, 1992.
[5] Y.-W. Chang, S. Thakur, K. Zhu, and D. F. Wong, "A new global routing algorithm for FPGA's," in *IEEE/ACM Proc. Int. Conf. Computer-Aided Design*, 1994, pp. 380–385.
[6] E. Dijkstra, "A note on two problems in connection with graphs," *Numer. Math.*, vol. 1, pp. 269–271, 1959.
[7] M. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA: Freeman, 1979.
[8] H.C. Hsieh, W. Carter, J. Ja, E. Cheung, S. Schreifels, C. Erickson, P. Freidin, L. Tinkey, and R. Kanaza, "Third-generation architecture boosts speed and density of field-programmable gate arrays," in *Proc. IEEE Custom Integrated Circuits Conf.*, 1990, pp. 31.2.1–31.2.7.
[9] T. C. Hu, *Integer Programming and Network Flows*. Reading, MA: Addison-Wesley, 1969.
[10] K. Kawana, H. Keida, M. Sakamoto, K. Shibata, and I. Moriyama, "An efficient logic block interconnect architecture for user-reprogrammable gate array," in *Proc. IEEE Custom Integrated Circuits Conf.*, 1990, pp. 31.3.1–31.3.4
[11] E. S. Kuh and M. Marek-Sadowska, "Global routing," in *Layout Design and Verification*, T. Ohtsuki, Ed. Amsterdam, The Netherlands: Elsevier, 1985.
[12] C. Papadimitriou and K. Steiglitz, *Combinatorial Optimization-Algorithms and Complexity*. Englewood Cliffs, NJ: Prentice-Hall, 1982.
[13] M. J. Lorenzetti and D. S. Baeder, "Routing," in *Physical Design Automation of VLSI System*, B. Preas and M. J. Lorenzetti, Eds. Menlo Park, CA: Benjamin-Cummings, 1988.
[14] G. Lemieux and S. Brown, "A detailed routing algorithm for allocating wire segments in field-programmable gate arrays," in *Proc. ACM/SIGDA Physical Design Workshop*, Lake Arrowhead, CA, 1993, pp. 215–226.
[15] J. Rose and S. Brown, "Flexibility of interconnection structures for field-programmable gate arrays," *IEEE J. Solid State Circuits*, vol. 26, no. 3, pp. 277–282, 1991.
[16] D. D. Sleator and R. E. Tarjan, "A data structure for dynamic trees," *J. Comput. Syst. Sci.*, vol. 26, pp. 362–391, 1983.
[17] S. Thakur, D. F. Wong, and S. Muthukrishnan, "Algorithms for FPGA switch module routing," in *Proc. European Design Automation Conf.*, 1994, pp. 265–270.
[18] S. Trimberger, Ed., *Field-Programmable Gate Array Technology*. Norwell, MA: Kluwer, 1994.
[19] S. Trimberger and M. Chene, "Placement-based partitioning for lookup-table-based FPGA's," in *Proc. Int. Conf. Computer Design, VLSI Computers Processors*, 1992, pp. 91–94.
[20] *The Programmable Logic Data Book*, Xilinx Inc., 1994.
[21] K. Zhu, D. F. Wong, and Y.-W. Chang, "Switch module design with application to two-dimensional segmentation design," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, 1993, pp. 481–486

Shashidhar Thakur (S'94–M'96) received the B.Tech. degree from the Indian Institute of Technology, Bombay, India, in 1990 and the M.S. and Ph.D. degrees from the University of Texas, Austin, in 1993 and 1996, respectively, all in computer science.

He was a Research Assistant in the Department of Computer Sciences, University of Texas, Austin, from 1992 to 1996. During the summers of 1991 and 1992, he was at the AT&T Bell Laboratories, Murray Hill, NJ, and during the summer of 1995, he was at Synopsys Inc. Presently, he is a Senior Research and Development Engineer at Synopsys Inc., Mountain View, CA. His research interests include algorithms, combinational optimization, logic synthesis, and physical design of VLSI circuits.

Yao-Wen Chang (S'95–A'96) received the B.S. degree in computer science and information engineering from National Taiwan University, Taiwan, R.O.C., in 1998 and the M.S. and Ph.D. degrees in computer science from the University of Texas, Austin, in 1993 and 1996, respectively.

He was a Second Lieutenant during his compulsory military service from 1998 to 1990, a Research Assistant at the Institute of Information Science, Academia Sinica, Taiwan, R.O.C., from 1990 to 1991, and a Teaching/Research Assistant in the Department of Computer Sciences, University of Texas, Austin, from 1992 to 1996. During the summer of 1994, he was with IBM J. Watson Research Center, Yorktown Heights, NY, in the VLSI group. Since 1996, he has been an Associate Professor in the Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan, R.O.C. His current research interests lie in design automation, architectures, and systems for VLSI and combinatorial optimization.

Dr. Chang received Best Paper Award at the 1995 IEEE International Conference on Computer Design (ICCD '95).

D. F. Wong received the B.Sc. degree in mathematics from the University of Toronto, Toronto, Canada, the M.S. degree in mathematics from the University of Illinois, Urbana-Champaign, and the Ph.D. degree in computer science from the University of Illinois, Urbana-Champaign, in 1987.

He is currently an Associate Professor of Computer Sciences at the University of Texas, Austin. He has published more than 120 technical papers. He is a coauthor of *Simulated Annealing for VLSI Design* (Norwell, MA: Kluwer, 1998). His main research interest is computer-aided design of VLSI.

Dr. Wong received Best Paper Awards at DAC-86 and ICCD-95 for his work on floorplan design and FPGA routing, respectively. He has served on the program committees of a number of VLSI CAD conferences (e.g., ICCAD, ED&TC, FPGA Symposium, and PD Symposium). He is an Editor of IEEE TRANSACTIONS ON COMPUTERS.

S. Muthukrishnan received the B.Tech. (Hons.) degree in computer science from the Indian Institute of Technology, Kharagpur, India, in 1989 and the M.S. and Ph.D. degrees from New York University, New York, in 1991 and 1994, respectively, all in computer science.

He was a Postdoctoral Visitor at DIMACS, Rutgers University, from 1994 to 1995 and on the faculty of University of Warwick, Coventry, UK, from 1995 to 1996. At present, he is a Member of the Technical Staff, Bell Laboratories, Lucent Technologies, Murray Hill, NJ.