Analysis of FPGA/FPIC Switch Modules

YAO-WEN CHANG
National Taiwan University
KAI ZHU
Q Design Automation
GUANG-MING WU
Nan-Hua University
D. F. WONG
University of Illinois at Urbana-Champaign
and
C. K. WONG
The Chinese University of Hong Kong

Switch modules are the most important component of the routing resources in FPGAs/FPICs. Previous works have shown that switch modules with higher routability result in better area performance for practical applications. We consider in this paper an FPGA/FPIC switch-module analysis problem: the inputs consist of a switch-module description and the number of nets required to be routed through the switch module; the question is to determine if there exists a feasible routing for the routing requirements on the switch module. As a fundamental problem for the analysis of switch modules, this problem is applicable to the design and routability evaluation of FPGA/FPIC switch modules and FPGA/FPIC routing. We present a network-flow-based approximation algorithm for this problem. Based on mathematical analyses, we show that this algorithm has provably good performance with the bounds 5 and 5/4 away from the optima for two types of switch modules, respectively. Extensive experiments show that this algorithm is highly accurate and runs very efficiently.

Categories and Subject Descriptors: B7.1 [**Integrated Circuits**]: Types and Design Styles—*Gate arrays*; B7.2 [**Integrated Circuits**]: Design Aids—*Placement and routing*; J.6 [**Computer Applications**]: Computer-Aided Engineering

General Terms: Algorithms, Design, Experimentation, Measurement, Performance

The work of Y.-W. Chang was partially supported by the National Science Council of Taiwan ROC under grant NSC 88-2215-E-009-064.

Authors' addresses: Y.-W. Chang, Graduate Institute of Electronics Engineering and Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan; email: ywchang@ee.ntu.edu.tw; K. Zhu, Q Design Automation; email: kaizhu@yahoo.com; G.-M. Wu, Department of Information Management, Nan-Hua University, Chiayi, Taiwan; email: gmwu@mail.nhu.edu.tw; D. F. Wong, Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801; email: mdfwong@uiuc.edu; C. K. Wong, Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, China; email: wongck@cse.cuhk.edu.hk.

Additional Key Words and Phrases: Computer-aided design of VLSI, layout, FPGA, FPIC, synthesis

1. INTRODUCTION

With their user-programmability, *field-programmable gate arrays* (*FPGAs*), the fastest growing segment in semiconductor industry during the past decade, have emerged as an unparalleled solution to the cost and time-to-market challenges by providing short turnaround time of designs with low risk/cost, allowing easy design changes. In general, an FPGA architecture consists of an array of logic modules which can be connected by general routing resources [Brown et al. 1992a]. Figures 1(a) and 1(b) illustrate two major FPGA architectures: symmetric-array-based and row-based FPGAs. The logic modules contain combinational and sequential circuits that implement logic functions. In a symmetric-array-based FPGA (see Figure 1(a)), the routing resources consist of horizontal and vertical channel and their intersection areas. The intersection area of a horizontal and a vertical channel is referred to as a *switch module*. In a row-based FPGA (see Figure 1(b)), an intersection area of a routing channel and a set of vertical segments can be viewed as a switch module. A net can change its routing direction via a switch module; this requires using at least one programmable switch inside the switch module. (See Figure 2 for a routing example.) A logic circuit is implemented in an FPGA by partitioning logic into individual logic modules and then interconnecting the modules by programming switches. A large circuit that cannot be accommodated into a single FPGA is divided into several parts; each part is realized by an FPGA, and these FPGAs are then interconnected by a *field-programmable interconnect chip* (*FPIC*) [Actel Corp. 1992; Aptix, Inc. 1992; Guo et al. 1992]. The routing architecture of an FPIC is similar to that of a symmetric-array-based FPGA, whereas the logic modules are replaced by pins (see Figure 3).

Previous works by Bhat and Hill [1992] and Trimberger and Chene [1992] have shown that the feasibility of FPGA design is constrained more by routing resources than by logic resources, and often routing delays, rather than logic-module delays, dominate the performance of FPGAs. Therefore it is desirable to facilitate routing in the design of FPGAs and FPICs. Switch modules are the most important component of the routing resources in FPGAs/FPICs. Studies by Brown et al. [1993], Chang et al. [1996a, 1996b], Rose and Brown [1991], Wu and Chang [1998, 1999], and Shvu et al. [2000] have shown that that the higher the routability of a switch module, the smaller the track count is needed to achieve 100% routing completion. The following crucial factors contribute to this phenomenon:

— Switch modules with higher routability increase the connectivity of routing components, and thus improve the overall routability of an FPGA.
— For practical applications, most connections are short; for example, about 60% (90%) of connections in the CGE [Brown et al. 1992b] and the SEGA [Lemieux and Brown 1993] benchmark circuits are routed through no more than two (five) switch modules, independent of FPGA sizes. (See

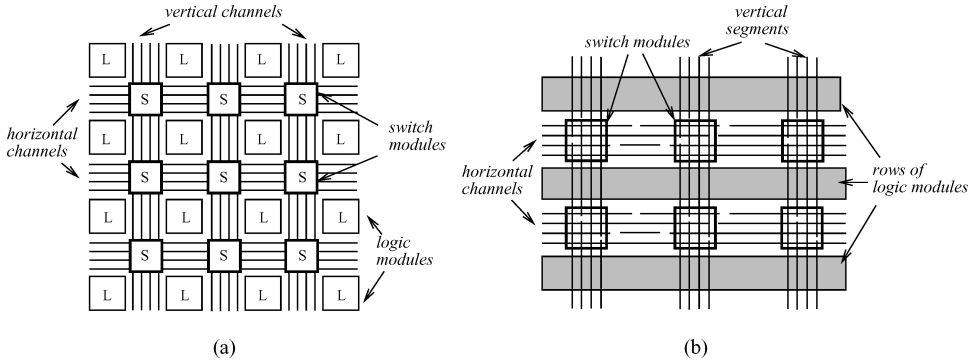

Fig. 1. Two major FPGA architectures. (a) Symmetric-array-based architecture. (b) Row-based architecture.

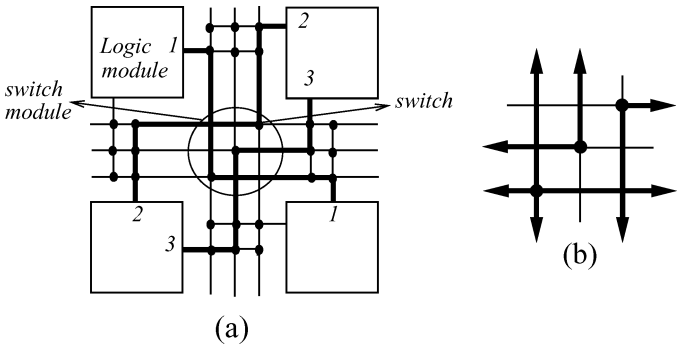

Fig. 2. (a) A routing example. (b) Switch-module routing.

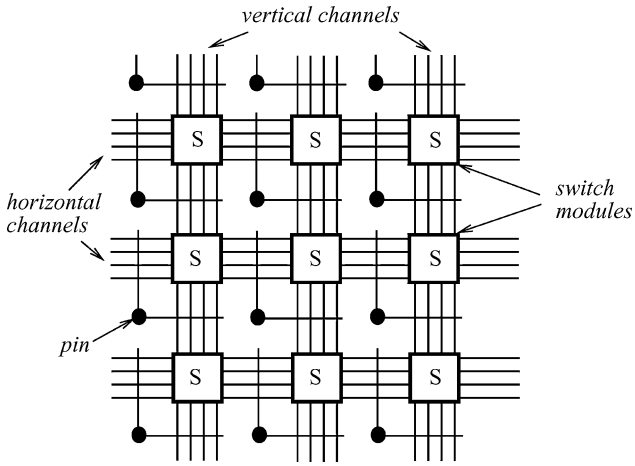

Fig. 3. The FPIC architecture.

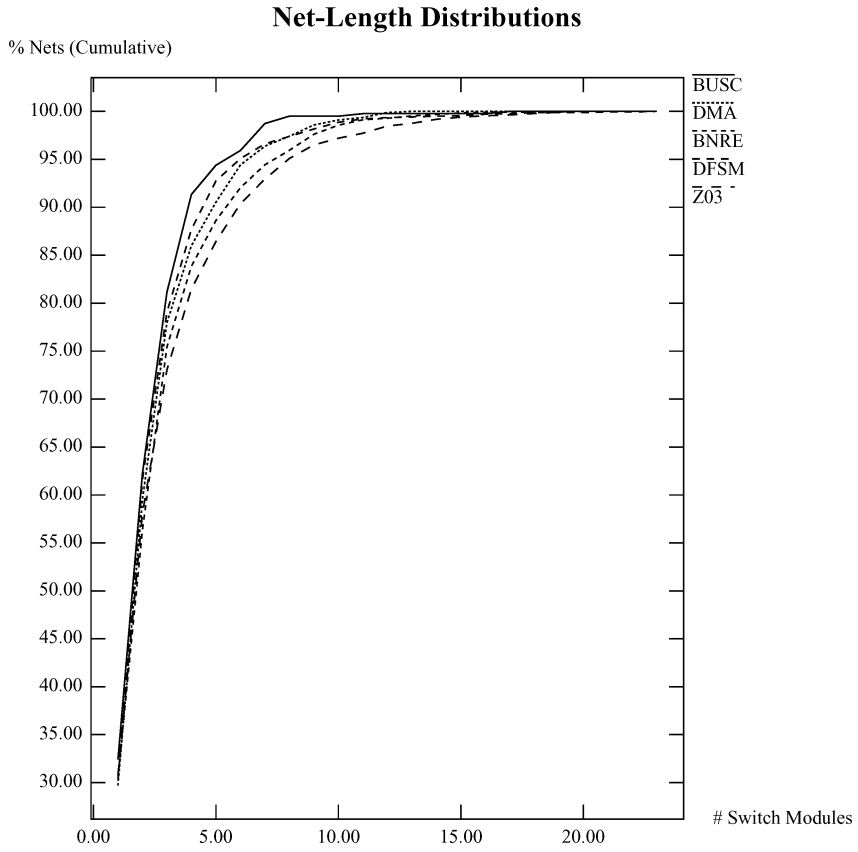

Fig. 4. Cumulative percentage of connections in the CGE benchmark circuits is plotted as a function of the number of switch modules passed. The FPGA sizes (number of logic modules) range from 12×13 (BUSC) to 26×27 (Z03).

Figure 4 for the distributions of net length for the CGE benchmark circuits, measured by the number of switch modules passed by a net.) Further, the switch modules used in current commercial FPGAs are homogeneous. Thus the routability of a single switch module plays an important role in overall FPGA routing.

— Most logic-module pins are logically equivalent [Trimberger 1994][1]; pin permutations combined with highly routable switch modules pave the way for optimizing routing.

Hence, increasing the routability of a switch module also improves the area performance of a router. Therefore, it is of significant importance to consider the analysis of switch modules.

The FPGA/FPIC programmable switches usually have high series resistance and parasitic capacitance, and consume a large amount of area. Due to the area and delay constraints, the number of switches that can be placed in a switch

[1]For example, the lookup-table and control inputs in a logic module are logically equivalent.

module is usually limited, implying limited routability. Therefore, there is a basic trade-off between routability and area/performance for switch-module architectures.

The effects of switch-module architectures on routing for the symmetric-array-based FPGAs with unit-length segments (i.e., every segment spans only one switch module) were first studied experimentally by Rose and Brown [1991]. A theoretical study of flexibility and routability was later presented based on a stochastic model [Brown et al. 1993]. The primary conclusion in both of the studies [Brown et al. 1993; Rose and Brown 1991] is that high pin-to-track connectivity together with relatively low switch-module connectivity is a better solution to the routability and area/performance trade-off. More recent studies based on FPGAs with segments of different lengths show that lower pin-to-track connectivity (e.g., one-quarter to one-half of the connectivity) combined with a high-quality switch-module architecture is sufficient for good area and routability trade-off [Betz et al. 1999; Betz and Rose 2000; Wilton 1997]. Therefore, the architecture of a switch module is of particular importance, due to a relatively small switch population in a switch module. Chang et al. [1996a, 1996b], analyzed three types of well-known switch modules; they showed theoretically and experimentally that switch modules with higher routing capacities[2] usually lead to better area performance, which confirms the findings by Brown et al. [1993] and Rose and Brown [1991]. Recent studies by Wu and Chang [1998, 1999] and Shyu et al. [2000] have shown that the routability of a single switch matrix plays a more important role when (1) the net density on a chip gets denser, and (2) the switch matrices become larger. Since denser applications and larger chips are trends of the commercial applications and products, the switch-matrix architectures would have even greater impact on FPGA/FPIC chip-level routability than they do now. Unlike the works by Brown et al. [1993], Chang et al. [1996a, 1996b], Rose and Brown [1991], and Wu and Chang [1998, 1999], which are based on real circuit designs and stochastic/analytical analyses, Wu et al. [1996] explored the routing behavior on the symmetric-array-based architecture using the *worst-case* scenario—they showed that it is NP-complete to determine whether a given global route can be mapped to a feasible detailed route in polynomial time; the worst cases occur when most nets are very long and are routed in some specially designed topologies, which rarely happen in practical applications [Lemieux et al. 1997]. Nevertheless, the work by Wu et al. [1996] still provides an important insight to the *worst-case* performance by using that routing architecture.

In this paper, we give more general switch-module models than that considered in Brown et al. [1993], Chang et al. [1996a, 1996b], Rose and Brown [1991], Shyu et al. [2000], Wu and Chang [1998, 1999], and Wu et al. [1996]) and study a routing-capacity analysis problem for switch modules. This problem is informally described as follows. The input consists of a switch-module description and the number of nets required to be routed through the switch module. The question is to determine if there exists a feasible routing for the

[2]Informally, the number of routable instance on a switch module. A formal definition is given in Section 2.

routing requirements on the switch module. A precise definition of the problem will be given in Section 2. As a fundamental problem to the analysis of FPGA/FPIC switch modules, this problem is applicable to the routability evaluation of FPGA/FPIC switch modules and thus the switch-module design for FPGAs/FPICs. Further, the analysis can also be used to compute the routing capacity of a switch module for developing a congestion metric defined on switch modules and thus help guide the global routing for FPGAs/FPICs [Chang et al. 1994, 1995a; Thakur et al. 1997].

To solve this analysis problem, Thakur et al. [1997] proposed an exact algorithm based on integer linear programming (ILP). However, since ILP is NP-complete [Garey and Johnson 1979], this ILP-based algorithm in the worst case is computationally expensive. Instead of resorting to the ILP, the works by Chang et al. [1996a, 1996b] and Wu and Chang [1998, 1999] solved the analysis problem on some special switch-module architectures.[3] Whether the *general* analysis problem can be solved in polynomial time is still open.

In this paper, we present an efficient network-flow-based approximation algorithm for analyzing the routing capacity of a single switch module. We show that the algorithm has provably good performance with the bounds 5 and 5/4 away from the optima for two types of switch modules, respectively. Extensive experiments show that the algorithm is highly accurate and runs very much faster than the ILP-based algorithm.

The remainder of this paper is organized as follows. Section 2 gives the formal problem formulation. Section 3 proposes an approximation algorithm for switch-module analysis. Section 4 presents the techniques for analyzing the performance of the approximation algorithm. Section 5 generalizes our approach to the cases where different net distributions and a more general routing model are considered. Experimental results are reported in Section 6.

2. PROBLEM FORMULATION

A *switch module* is a $W \times W$ square block with W terminals on each side of the block. There are two types of switch modules, *switch blocks* and *switch matrices* (see Figure 5). In a switch block (see Figure 5(a)), some pairs of terminals on different sides may be connected by programmable switches. In particular, connecting two terminals requires one and only one switch, and the switches are electrically *noninteracting*, unless they share a terminal. Switch blocks are used in symmetric-array FPGAs for connections between single-length lines or between double-length lines [Xilinx, Inc. 1996]. A switch matrix (see Figure 5(b)) consists of a grid of horizontal and vertical tracks. There are two types of switches in a switch matrix, *crossing switches* and *separating switches*. If a crossing switch at the intersection of a horizontal and a vertical tracks is "ON," the two tracks are connected; if it is "OFF," the tracks are not connected and thus are electrically noninteracting. If a separating switch on a track is

[3]More precisely, they solved the analysis problem on some switch *blocks* such as those used in the Lucent Technologies and Xilinx FPGAs [Lucent Technologies 1996; Xilinx, Inc. 1996] and the diagonal switch matrices [Wu and Chang 1998, 1999]. See Section 2 for the formulation of switch blocks and matrices.

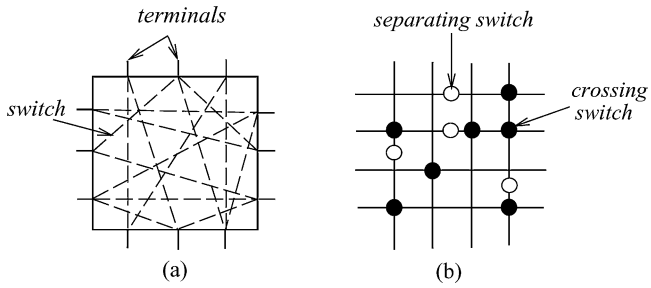

Fig. 5. (a) A switch block. (b) A switch matrix.

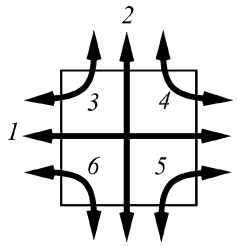

Fig. 6. Six types of connections.

"OFF," the track is split into two electrically noninteracting routing segments so that the terminals on opposite sides can be used independently; if it is "ON," the track becomes a single electrical track. In Figure 5(b), the crossing switches are represented by solid circles and the separating switches by hollow circles. Switch matrices are used in various symmetric-array FPGAs [Lucent Technologies 1996; Xilinx, Inc. 1996], row-based FPGAs [Actel Corp. 1996; Gamal et al. 1989; Marple and Cooke 1992], and FPICs [Actel Corp. 1992].

A *switch-module connection* is an electrical path between two terminals on different sides of a switch module. Switch-module connections can be of six types, each of which is characterized by two sides of a module, as shown in Figure 6. For example, type-6 connections connect terminals on the left and the bottom sides of a module. Type-1 and type-2 connections are *straight connections* whereas the others are *bent connections*. We first assume that at most one switch can be used—programmed to be "ON"—by a connection on a switch module (i.e., a net can use at most n switches if it passes through n switch modules); this restriction represents a suitable balance between routability and performance for a switch module, and is thus a reasonable assumption for the purpose of switch-module analysis. Extension to the case for using multiple switches for a switch-module connection is simple and will be discussed in Section 5.

A *routing requirement vector* (*RRV* for short) is a vector $\vec{n} = (n_1, n_2, \ldots, n_6)$, where n_i is the number of type-i connections required to be routed through a switch module, $0 \leq n_i \leq W$, $i = 1, 2, \ldots, 6$. A switch module M is *$\vec{n}$-routable* if n_i connections of type i, $1 \leq i \leq 6$, can be routed through M simultaneously, and each connection uses at most one switch. In this case, the RRV $\vec{n}$ is said to be *routable* on M. For example, the RRVs (0, 0, 1, 1, 1, 0) and (1, 1, 1, 0, 1, 0) are routable on the switch matrix shown in Figure 2, and their routing solutions

are shown in Figure 2(a) and Figure 2(b), respectively. We refer to the *routing capacity* of a switch module M as the number of distinct routable RRVs on M; that is, the routing capacity of M is the cardinality $|\{\vec{n}|\vec{n}$ is routable on $M\}|$. The *routability* of a switch module M is referred to as the *probability* that an RRV is routable on M.

In this paper, we consider the *switch-module analysis problem* (*SMAP*):

Input: A switch module M (could be a switch matrix or a switch block) and an RRV $\vec{n}$.

Question: Is $\vec{n}$ routable on M?

Note that connections in a switch block interfere with each other if and only if they share a terminal. In a switch matrix, however, they can also interfere with each other if they share a part of a track. Thus the feasibility conditions for switch blocks are "simpler" than those for switch matrices, and so is the SMAP associated with switch blocks.

3. THE NETWORK-FLOW ANALYZER

In this section, we first present a network-flow-based algorithm to solve a class of special SMAP's on the two types of switch modules, and then show how to apply the algorithm as a subroutine to *approximate* the general SMAP's. We will, in Section 4, mathematically analyze the performance of the approximation algorithm.

Consider the category of SMAPs with generic $\vec{n}$'s in which the components corresponding to any three types of connections which share one side of the switch module are nonzero and the remaining components are zero; for example, the SMAP with $\vec{n} = (n_1, 0, n_3, 0, 0, n_6)$, $(0, n_2, n_3, n_4, 0, 0)$, $(n_1, 0, 0, n_4, n_5, 0)$, or $(0, n_2, 0, 0, n_5, n_6)$. For each of these special SMAPs, we can reduce it to computing the maximum flow in a network. Given a switch module and an RRV $\vec{n}$, a network is constructed as follows. (Figure 7(d) shows the network for the switch matrix shown in Figure 7(a).) Pick one side of the switch module as the *sink side* and make the others *source sides*; in Figure 7(d), for instance, the right side is chosen as the sink side. Construct a sink vertex t for the sink side, three source vertices s_i's for the source sides, where i denotes the corresponding type of connections, and a supersource vertex s. Connect s to s_i by an edge (s, s_i) with a capacity equal to the corresponding routing requirement n_i. Label the terminals $1, 2, \ldots, 4W$, starting from the bottom-most terminal on the left side and proceeding clockwise (see Figure 7(a)). For a terminal labeled j, create a vertex v_j. Connect each source vertex s_i to the v_j's with respect to the same source side by a unit-capacity edge (s_i, v_j). Similarly, connect each v_j with respect to the sink side to t by a unit-capacity edge (v_j, t). For switch blocks, if there is a switch between two terminals k and l, one on a source side and the other on the sink side, construct a unit-capacity edge between v_k and v_l. For switch matrices, we need to consider the following cases for the construction of the edges between v_k and v_l:

(1) If one of terminals k and l is on a source side and the other on the sink side and they are on opposite sides of the same track (e.g., terminals 2 and 5 in

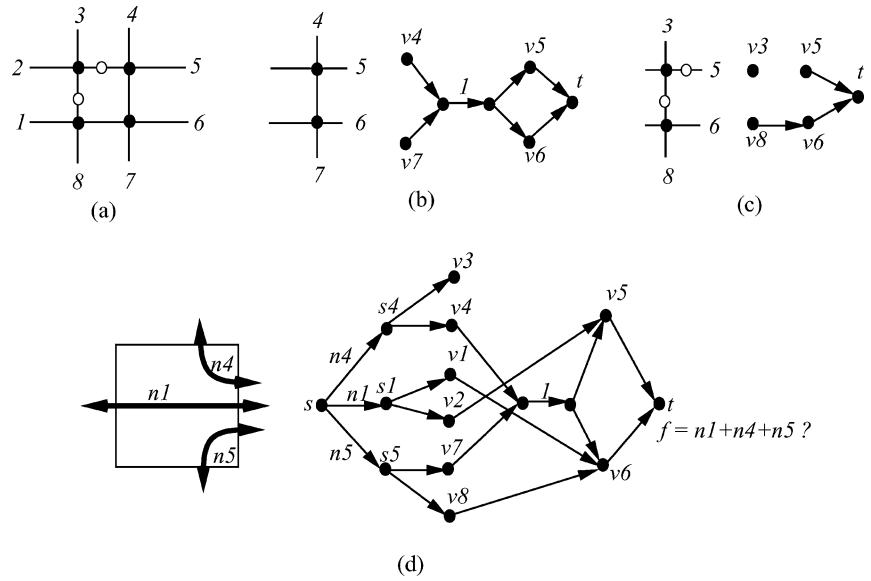

Fig. 7. Examples of network-flow construction. (a) A switch matrix instance. (b), (c) Network construction. (d) A complete network.

Figure 7(a)), construct a unit-capacity edge between v_k and v_l. The reason is that a connection between the two terminals can be established by programming the separating switch, if any, to be "ON," or can be immediately established if there is no separating switch on the track. In either case, the crossing switches on the track, if any, remain "OFF."

(2) If terminals k and l are both on source sides and are located on opposite sides of the same track (e.g., terminals 4 and 7 in Figure 7(a)), connect v_k and v_l as shown in Figure 7(b) if there is no separating switch on the track, or connect them as illustrated in Figure 7(c) if there is a separating switch on the track. Note that terminals 4 and 7 in Figure 7(a) electrically interfere with each other. Hence, we shall connect v_4 and v_7 to a unit-capacity edge before connecting them to other vertices. (See Figure 7(b).) Also, there is a separating switch on the bent connection between terminals 3 and 5. We need not to construct an edge between v_3 and v_5 because such a bent connection requires to use at least two switches, one crossing switch and one separating switch. (See Figure 7(c).) Note that, as mentioned earlier, at most one switch can be used for routing through a switch module.

The network-flow analyzer proceeds as follows. Pick one side of the switch module as the sink side and construct a flow network. Given such a flow network, we ask if there is a feasible flow where, for each i, s_i supplies a flow n_i and t receives a flow of $\sum_i n_i$. This problem can be solved by a network-flow algorithm in time $O(W(N+W)\log W)$, where N is the number of switches [Sleator and Tarjan 1983]. (Note that the number of vertices [edges] in the network is $O(W)$ $[O(N+W)]$.) If the answer is "NO," the switch module is not $\vec{n}$-routable; otherwise, pick another side of the switch module as the sink and repeat the above

Algorithm: Switch_Module_Analysis($M, \vec{n}$)
Input: M: A given switch module;
$\vec{n} = (n_1, n_2, \ldots, n_6)$: A routing requirement vector.
Output: *routable*.
/* Flag *routable* $= TRUE$ if $\vec{n}$ is routable on M; $FALSE$, otherwise. */

begin
1 **for** *sinkside* $\in \{left, top, right, bottom\}$ /* four sides of a switch module */
2 Pick *sinkside* as the sink side and the others as source sides, construct the flow network N for the corresponding three types, T_1, T_2, T_3, of connections;
3 Apply the maximum network-flow algorithm to compute the max flow f on N;
4 **if** $f < \sum_{i \in \{T_1, T_2, T_3\}} n_i$
5 **return** *routable* $\leftarrow FALSE$;
6 **return** *routable* $\leftarrow TRUE$
end

Fig. 8. Algorithm for switch-module analysis.

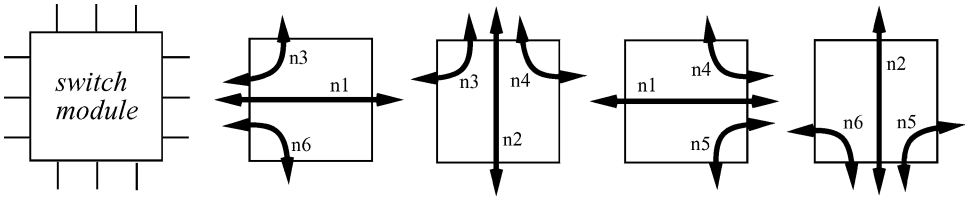

Fig. 9. The four networks used to analyze the six types of connections for a switch module.

procedure. Since there are four candidates for the sink, at most four networks need to be constructed to analyze *all six types* of connections. (See Figure 9.) If t can receive a flow of $\sum_i n_i$ in each of the four flow networks, the switch module is reported to be $\vec{n}$-routable. Figure 8 summarizes the network-flow-based algorithm for switch-module analysis.

Note that every vertex in the flow network, except s and t, has either in-degree or out-degree one, and every edge, except $\{(s, s_i)|i$ is a type of connections considered in the network$\}$, has a unit capacity. Hence the routing paths from s_i to t computed by a maximum network-flow algorithm are vertex-disjoint and represent legal routing paths. Since for each sink side, its corresponding flow network considers only three out of six types of connections simultaneously, the network-flow analyzer may fail to detect some unroutable RRVs. However, if an RRV $\vec{n}$ is routable on a switch module, the analyzer will always report the switch module to be $\vec{n}$-routable. Therefore, the analyzer will never underestimate the routability of the switch module.

For convenience, we refer to the network-flow-based approximation algorithm as the *flow analyzer* and an exact algorithm such as the ILP solver as an *exact analyzer*.

4. PERFORMANCE ANALYSIS

In this section, we analyze the performance bounds of the flow analyzer—a ratio of the sizes of two feasible sets obtained by the flow analyzer and by an exact

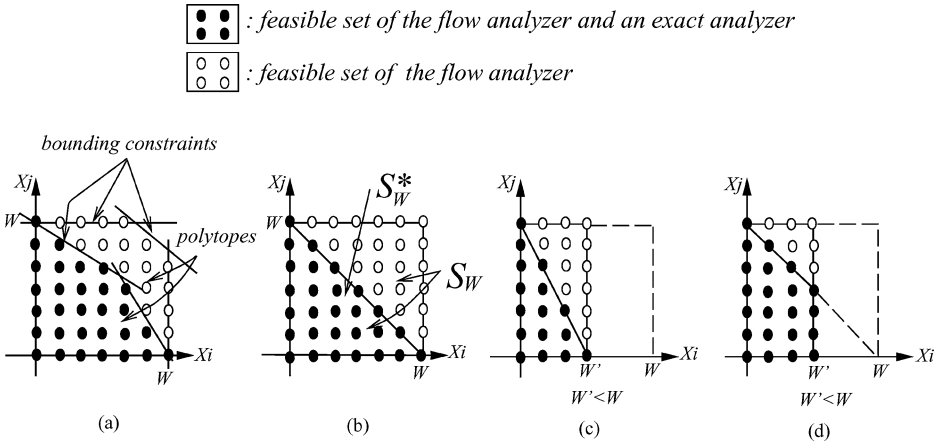

Fig. 10. The feasible sets with respect to the flow analyzer and an exact analyzer. (a) Two different feasible sets and their corresponding constraints. (b) An extreme instance. (c) The situation after scaling along the x_i axis for the instance in (b). (d) Intersection with the set $\{\vec{x}|x_i \leq W'\}$.

analyzer. The analysis procedure is described as follows. We first characterize the maximum feasible set for the flow analyzer and the minimum one for an exact analyzer for the case where there are W connections of each type. We then compute the sizes of these two sets by combinatorial counting techniques. It will be clear later that the ratio of the two sets is an increasing function of W. Hence, the ratio is upper bounded by the value computed from the case where $W \to \infty$. Then, we will generalize the bound to arbitrary cases. Our proofs show that the flow analyzer has the performance bounds 5 for switch matrices and 5/4 for switch blocks.

As mentioned earlier, the feasibility conditions for switch blocks are "simpler" than those for switch matrices. The routability-analysis techniques for switch matrices readily extend to switch blocks. We hence will focus on switch matrices and give only brief remarks on switch blocks.

4.1 Preliminaries

Let S_W and S^*_W be the *feasible sets* for those RRVs on a switch module M of size W with respect to the flow analyzer and an exact analyzer, respectively, in which the corresponding analyzer reports that M is $\vec{n}$-routable. We can think of an RRV as a nonnegative lattice point in an orthogonal six-dimensional (6-D) space with axes x_i, $1 \leq i \leq 6$. Because each component, n_i, $1 \leq i \leq 6$, of $\vec{n}$ satisfies the constraint $0 \leq n_i \leq W$, it is possible to bound S_W and S^*_W by a set of linear inequalities. Figure 10 illustrates a 2-D analogy. Each lattice point inside a feasible set, S_W or S^*_W, which is constrained by a set of inequalities, corresponds to a feasible solution with respect to the flow analyzer or the exact analyzer. For convenience, we will use $\vec{n} = (n_1, \ldots, n_6)$ to represent both a generic routing requirement *vector* and the corresponding 6-D lattice *point*. Its exact meaning will be clear from the context.

Let $\vec{n} = (n_1, \ldots, n_6)$ and $\vec{n^*} = (n^*_1, \ldots, n^*_6)$ be respective feasible lattice points with respect to the flow analyzer and an exact analyzer in analyzing

the routability of a switch module. Let $\tilde{n_i} = \max\{n_i | \vec{n} \in S_W\}$ and $\tilde{n_i^*} = \max\{n_i^* | \vec{n^*} \in S_W^*\}$, $1 \le i \le 6$.

LEMMA 1. *The following properties hold:*

(1) $S_W^* \subseteq S_W$;
(2) $\forall 1 \le i \le 6$, $\tilde{n_i} = \tilde{n_i^*}$;
(3) *for switch matrices,* $\tilde{n_i} = \tilde{n_i^*} = W$, $i = 1$ *or* 2.

PROOF.

(1) Since the flow analyzer considers only three out of six types of nets at a time, underestimation is impossible and thus S_W must contain S_W^*.
(2) Consider one component n_i, $1 \le i \le 6$, at a time by setting the other components equal to zero. The SMAP with a $\vec{n}$ that has only one nonzero component for an arbitrary switch module M is merely a special case of those discussed in the previous section; therefore, it can be solved by the flow analyzer. This implies that, for the SMAPs with the special $\vec{n}$'s on M, if the flow analyzer reports that M is $\vec{n}$-routable, so does an exact analyzer, and vice versa. Hence, $\tilde{n_i} = \tilde{n_i^*}$, $\forall 1 \le i \le 6$.
(3) It is obvious that type-1 and type-2 connections are noninteracting. Since there are W horizontal tracks and W vertical tracks, both $\tilde{n_i}$ and $\tilde{n_i^*}$ can be as large as W. □

4.2 Performance Bound

As mentioned earlier, we may evaluate the flow analyzer based on its performance bound; that is, we compute $|S_W|/|S_W^*|$. Let

$$\begin{aligned} I_W &= \{\vec{n} | n_1 + n_3 + n_6 \le W,\ n_2 + n_3 + n_4 \le W,\ n_1 + n_4 + n_5 \le W, \\ &\qquad n_2 + n_5 + n_6 \le W\}, \\ I_W^* &= \left\{ \vec{n} | \max\{n_1, n_2\} + \sum_{i=3}^{6} n_i \le W \right\}. \end{aligned}$$

We have the following lemma.

LEMMA 2 (CONTAINMENT PROPERTIES). *The following properties hold:*

(1) $S_W \subseteq I_W$;
(2) *for switch matrices,* $S_W^* \supseteq I_W^*$, *when* $\tilde{n_i} = W$, $1 \le i \le 6$.

PROOF.

(1) As mentioned earlier, the flow analyzer analyzes a switch module one side at a time. The total number of connections routed through each side must be bounded by the size of the switch module, W. Hence, if $\vec{n} \in S_W$, the constraints $n_1 + n_3 + n_6 \le W$, $n_2 + n_3 + n_4 \le W$, $n_1 + n_4 + n_5 \le W$,

and $n_2 + n_5 + n_6 \leq W$ for the four sides must be satisfied. This means $\vec{n} \in S_W \Rightarrow \vec{n} \in I_W$, and hence $S_W \subseteq I_W$.

(2) When $\tilde{n_i} = W$, $1 \leq i \leq 6$, $\tilde{n_i^*} = \tilde{n_i} = W$, by Lemma 1; that is, there exist W paths for each of the six types. We first consider the case where $n_1 = n_2 = 0$. Since there are W paths for each of the six types, it is easy to see that there always exist $W - n_i$ paths available for each of other types of bent connections after n_i type-i connections are routed, $i \in \{3, 4, 5, 6\}$. Thus, if $n_1 = n_2 = 0$ and $\sum_{i=3}^{6} n_i \leq W$, there must exist at least one feasible routing. For the case where $n_1 \neq 0$ or $n_2 \neq 0$, we observe that the number of tracks available for type-1 (type-2) connections is decreased by one after routing a bent connection. Thus there are $W - \sum_{i=3}^{6} n_i$ horizontal and $W - \sum_{i=3}^{6} n_i$ vertical tracks remaining after $\sum_{i=3}^{6} n_i$ bent connections are routed. Therefore, if $\max\{n_1, n_2\} \leq W - \sum_{i=3}^{6} n_i$, there must exist at least one feasible routing, that is, $\vec{n} \in I_W^* \Rightarrow \vec{n} \in S_W^*$. Hence $I_W^* \subseteq S_W^*$. □

By Lemma 2, we have $|S_W|/|S_W^*| \leq |I_W|/|I_W^*|$ for switch matrices when $\tilde{n_i} = W$, $1 \leq i \leq 6$. To compute the bound, we can compute $|I_W|/|I_W^*|$ instead. We have the monotone property of $|I_W|/|I_W^*|$.

LEMMA 3 (MONOTONE PROPERTY). *$|I_W|/|I_W^*|$ is a strictly increasing function of W, when $W \geq 0$.*

PROOF. Since

$$I_W^* = \left\{ \vec{n} | \max\{n_1, n_2\} + \sum_{i=3}^{6} n_i \leq W \right\},$$

we have

$$\sum_{i=3}^{6} n_i \leq W - \max\{n_1, n_2\}.$$

Consider the following two sets:

$$I_{W,k,1}^* = \{(n_1, n_2) | \max\{n_1, n_2\} = k,\ 0 \leq k \leq W\},$$

$$I_{W,k,2}^* = \left\{ (n_3, n_4, n_5, n_6) | \sum_{i=3}^{6} n_i \leq W - k,\ 0 \leq k \leq W \right\}.$$

We have

$$|I_{W,k,1}^*| = 2k + 1,\ 0 \leq k \leq W,$$

and

$$|I_{W,k,2}^*| = \left| \left\{ (n_3, n_4, n_5, n_6) | \sum_{i=3}^{6} n_i \leq W - k,\ 0 \leq k \leq W \right\} \right|$$

$$= \left| \left\{ (n_3, n_4, n_5, n_6, n_7) | \sum_{i=3}^{7} n_i = W - k,\ 0 \leq k, n_i \leq W \right\} \right|.$$

This is equivalent to counting the number of selections, with repetition, of size $W-k$ from a collection of size 5. Thus we have

$$|I^*_{W,k,2}| = \binom{W-k+4}{4}.$$

As a result,

$$\begin{aligned}
|I^*_W| &= \left|\left\{\vec{n} | \max\{n_1, n_2\} + \sum_{i=3}^{6} n_i \le W\right\}\right| \\
&= \sum_{k=0}^{W} |I^*_{W,k,1}||I^*_{W,k,2}| \\
&= \sum_{k=0}^{W} (2k+1)\binom{W+4-k}{4} \\
&= \sum_{k=1}^{W+1} k\binom{W+5-k}{4} + \sum_{k=1}^{W} k\binom{W+4-k}{4} \\
&= \sum_{k=0}^{W+5} \binom{k}{1}\binom{W+5-k}{4} + \sum_{k=0}^{W+4} \binom{k}{1}\binom{W+4-k}{4} \\
&= \binom{W+6}{6} + \binom{W+5}{6} \\
&= \frac{1}{6!}\vec{A}^* \cdot \vec{\omega},
\end{aligned}$$

where

$$\begin{aligned}
\vec{A}^* &= (2, 36, 260, 960, 1898, 1884, 720), \\
\vec{\omega} &= (W^6, W^5, W^4, W^3, W^2, W, 1).
\end{aligned}$$

Note that the identity

$$\sum_{k=0}^{l} \binom{l-k}{m}\binom{q+k}{n} = \binom{l+q+1}{m+n+1},\ n \ge q \ge 0,\ l, m, n, q \in \mathcal{Z}^+ \cup \{0\},$$

is an extension of *Vandermonde's convolution* [Graham at al. 1989]. Also, $\sum_{k=0}^{W+5}\binom{k}{1}\binom{W+5-k}{4}$ is derived from $\sum_{k=1}^{W+1} k\binom{W+5-k}{4}$ by adding several zero terms such as $\binom{0}{1}\binom{W+5}{4}$.

Similar to techniques used for computing $|I^*_W|$, to compute $|I_W|$, we can consider the values of n_1 and n_2 first and simplify the system of linear inequalities. This process continues for n_3 and n_5 (or n_4 and n_6) until all "basic" terms are found. (Alternatively, we can assume that $|I_W| = aW^6 + bW^5 + cW^4 + dW^3 + eW^2 + fW + g$ since the number of the lattice points in a six-dimensional space bounded by the size W in each axis is in the order of W^6. Since we have seven unknown variables, $a, b, c, \ldots, g$, we only need to find $|I_W|$'s for seven W's to obtain a system of seven equations to solve all the seven variables. [It is easy to write a program to compute $|I_W|$'s for small W's.] The assumed polynomial

function can then be verified by mathematical induction.) For conciseness, we just give the results as follows:

$$|I_W| = \begin{cases} \frac{1}{6!}\vec{A}_1 \cdot \vec{\omega}, & W = 2k, k \in \mathcal{Z}^+ \cup \{0\}; \\ \frac{1}{6!}\vec{A}_2 \cdot \vec{\omega}, & W = 2k+1, k \in \mathcal{Z}^+ \cup \{0\}, \end{cases}$$

where

$$\begin{aligned} \vec{A}_1 &= (10, 120, 595, 1560, 2320, 1920, 720), \\ \vec{A}_2 &= (10, 120, 595, 1560, 2320, 1920, 675). \end{aligned}$$

Since $|I_W|$ is an integer, we have

$$|I_W| = \left\lfloor \frac{1}{6!}\vec{A} \cdot \vec{\omega} \right\rfloor,$$

where $\vec{A} = \vec{A}_1$.

To show that $|I_W|/|I_W^*|$ is strictly increasing as W grows, observe that

$$\begin{aligned} \frac{|I_{W+1}|}{|I_{W+1}^*|} - \frac{|I_W|}{|I_W^*|} &= \frac{1}{|I_{W+1}^*||I_W^*|}(|I_{W+1}||I_W^*| - |I_W||I_{W+1}^*|) \\ &= \frac{1}{|I_{W+1}^*||I_W^*|}\left(\left\lfloor \frac{1}{6!}\vec{A} \cdot \vec{\omega'} \right\rfloor \left(\frac{1}{6!}\vec{A}^* \cdot \vec{\omega}\right) - \left\lfloor \frac{1}{6!}\vec{A} \cdot \vec{\omega} \right\rfloor \left(\frac{1}{6!}\vec{A}^* \cdot \vec{\omega'}\right)\right) \\ &\geq \frac{1}{|I_{W+1}^*||I_W^*|}\left(\left(\frac{1}{6!}\vec{A} \cdot \vec{\omega'} - 1\right)\left(\frac{1}{6!}\vec{A}^* \cdot \vec{\omega}\right)\right. \\ &\quad \left. - \left(\frac{1}{6!}\vec{A} \cdot \vec{\omega}\right)\left(\frac{1}{6!}\vec{A}^* \cdot \vec{\omega'}\right)\right) \\ &= \frac{10\vec{B} \cdot \vec{\omega''}}{(6!)^2|I_{W+1}^*||I_W^*|} \\ &> 0, \end{aligned}$$

where

$$\begin{aligned} \vec{B} &= (12, 342, 4311, 31608, 148806, 467172, 980034, 1332072, 4416957, \\ &\quad 446526, 55080), \\ \vec{\omega'} &= ((W+1)^6, (W+1)^5, (W+1)^4, (W+1)^3, (W+1)^2, (W+1), 1), \\ \vec{\omega''} &= (W^{10}, W^9, W^8, W^7, W^6, W^5, W^4, W^3, W^2, W, 1). \end{aligned}$$

Since $|I_W|/|I_W^*| < |I_{W+1}|/|I_{W+1}^*|$ when $W \geq 0$, $|I_W|/|I_W^*|$ is a strictly increasing function of W. □

Table I in Section 4.4 lists the cardinalities of I_W and I_W^*, computed by the closed forms shown above. The reader may check with the table for the monotone property of $|I_W|/|I_W^*|$.

THEOREM 1 (PERFORMANCE BOUND). *For switch matrices,* $|S_W|/|S_W^*| \leq 5$.

PROOF.

(i) Nondegenerate cases where $\tilde{n_i} = W, \forall 1 \leq i \leq 6$:
By Lemma 1(2), $\tilde{n_i^*} = \tilde{n_i} = W, 1 \leq i \leq 6$, and by Lemma 2, $S_W \subseteq I_W$ and

$S^*_W \supseteq I^*_W$. Since $|I_W|/|I^*_W|$ is monotonically increasing (Lemma 3), we have

$$\begin{aligned}\frac{|S_W|}{|S^*_W|} &\le \frac{|I_W|}{|I^*_W|} \\ &\le \lim_{W\to\infty} \frac{|I_W|}{|I^*_W|} \\ &\le \lim_{W\to\infty} \frac{\vec{A}\cdot\vec{\omega}}{\vec{A}^*\cdot\vec{\omega}} \\ &= 5,\end{aligned}$$

where

$$\begin{aligned}\vec{A}^* &= (2, 36, 260, 960, 1898, 1884, 720), \\ \vec{A} &= (10, 120, 595, 1560, 2320, 1920, 720), \\ \vec{\omega} &= (W^6, W^5, W^4, W^3, W^2, W, 1).\end{aligned}$$

(ii) Degenerate cases where there is some $\tilde{n_i} = W'$, $W' < W$:
Consider the following continuous sets. Let

$$\begin{aligned}P^*_W &= \left\{\vec{x} \,|\, \max\{x_1, x_2\} + \sum_{i=3}^{6} x_i \le W, x_i \in \Re^+ \cup \{0\}\right\}, \\ P_W &= \{\vec{x} | x_1 + x_3 + x_6 \le W,\ x_2 + x_3 + x_4 \le W,\ x_1 + x_4 + x_5 \le W, \\ &\quad x_2 + x_5 + x_6 \le W,\ x_i \in \Re^+ \cup \{0\}\}.\end{aligned}$$

P^*_W and P_W form two 6-D bounding polytopes for I^*_W and I_W, and are referred to as an *exact polytope* and a *flow polytope*, respectively; let their respective volumes be $V_{P^*_W}$ and V_{P_W}.

As $W \to \infty$, $|S_W|/|S^*_W|$ converges to $V_{P_W}/V_{P^*_W}$. Similarly to the claim given in Lemma 2 (Containment Properties) with the domain of nonnegative integers (I_W and I^*_W), we have the the following inequality with the domain of nonnegative real numbers:

$$\frac{|S_W|}{|S^*_W|} \le \frac{V_{P_W}}{V_{P^*_W}}.$$

V_{P_W} and $V_{P^*_W}$ can be interpreted as the sizes of the respective sets I_W and I^*_W, as $W \to \infty$. Therefore, $V_{P^*_W}$ and V_{P_W} are given by the leading terms of $|I^*_W|$ and $|I_W|$; we thus have $V_{P^*_W} = 2W^6/6!$ and $V_{P_W} = 10W^6/6! = W^6/72$. (Note that both $V_{P^*_W}$ and V_{P_W} can also be calculated by multiple integration as follows:

$$\begin{aligned}V_{P^*_W} &= \int_{\vec{x}\in P^*_W} d\vec{x} \\ &= W^6 \int_0^1 \int_0^{1-x_6} \int_0^{1-x_5-x_6} \int_0^{1-x_4-x_5-x_6} \\ &\quad \times (1 - x_3 - x_4 - x_5 - x_6)^2 \, dx_3 \, dx_4 \, dx_5 \, dx_6 \\ &= \frac{2W^6}{6!}\end{aligned}$$

and

$$\begin{aligned} V_{P_W} &= \int_{\vec{x}\in P_W} d\vec{x} \\ &= W^6 \int_0^1 \int_0^{1-x_6} \int_0^{1-x_5} \int_0^{\min(1-x_4,1-x_6)} \int_0^{\min(1-x_3-x_4,1-x_5-x_6)} \\ &\quad f(\vec{x})\, dx_2\, dx_3\, dx_4\, dx_5\, dx_6 \\ &= \frac{W^6}{72}, \end{aligned}$$

where $f(\vec{x}) = \min(1 - x_3 - x_6, 1 - x_4 - x_5)$.)

Since $\tilde{n_i} = W'$, $W' < W$, by Lemma 1(2), we have $\tilde{n_i} = \tilde{n_i^*} = W'$. Both the flow polytope and the exact polytope are scaled by W'/W along the x_i axis or cut by the plane $x_i = W'$; see Figures 10(c) and 10(d). In both cases, it is easy to see that $|S_W|/|S_W^*|$ is still bounded by the bound obtained in the nondegenerate case, that is, $|S_W|/|S_W^*| \leq 5$. For conciseness, we show only the case where both polytopes are scaled by W'/W, $W' > 0$, along some axis.

Let V'_{P_W} and $V'_{P_W^*}$ be degenerate volumes for the flow polytope and the exact polytope, respectively. We have

$$\begin{aligned} \frac{|S_W|}{|S_W^*|} &\leq \frac{V'_{P_W}}{V'_{P_W^*}} \\ &= \frac{(W'/W)V_{P_W}}{(W'/W)V_{P_W^*}} \\ &= \frac{W^6/72}{2W^6/6!} \\ &= 5. \end{aligned}$$

Note that if there are no feasible type-i connections for some i, $3 \leq i \leq 6$, neither S_W nor S_W^* contains a solution with $n_i > 0$, $3 \leq i \leq 6$, by Lemma 1(2); that is, there exists some W', $W' = 0$. Since, for switch matrices, by Lemma 1(3), $n_1(n_1^*)$ and $n_2(n_2^*)$ can be as large as W, they can at most degenerate to two 2-D polytopes. For example, when there are no crossing switches in a switch matrix, n_i and n_i^*, $3 \leq i \leq 6$, must be zero. This is a trivial best case that $|S_W|/|S_W^*| = (W + 1)^2/(W + 1)^2 = 1$. □

The theoretical performance bound 5 in Theorem 1 for switch matrices occurs when a switch matrix of size W, $W \to \infty$, contains no separating switches and has a set of W crossing switches on all vertical and horizontal tracks in the switch matrix (one of the W crossing switches on each vertical track or each horizontal track).

In current commercially available FPGAs, the sizes of switch modules are usually small, say $W \leq 30$. We can compute the performance bounds by the two closed forms $\vec{A}^* \cdot \omega$ and $\lfloor \vec{A} \cdot \omega \rfloor$ shown in the proof of Lemma 3. Our results

show that the bounds for $W = 10$, 20, and 30 are about 3.177, 3.862, and 4.174, respectively, which are better than the theoretical bound 5 for the case $W \to \infty$. See Table I in Section 4.4 for details. This is the worst-case performance of the flow analyzer. It will be seen in Section 6 that the flow analyzer has the empirical performance bounds 1.062 and 1.111 away from the optima for $W = 10$ and 20, respectively.

4.3 Remarks on the Results for Switch Blocks

As mentioned earlier, the techniques for the analyses associated with switch blocks are similar to those for switch matrices. We give the results in the following. Let

$$\begin{aligned} I_W &= \{\vec{n} | n_1 + n_3 + n_6 \le W,\ n_2 + n_3 + n_4 \le W,\ n_1 + n_4 + n_5 \le W, \\ &\quad n_2 + n_5 + n_6 \le W\}, \\ I_W^{*\prime} &= \{\vec{n} | \max\{n_1, n_2\} + \max\{n_3, n_5\} + \max\{n_4, n_6\} \le W, \\ &\quad n_1 + n_2 + \max\{n_3 + n_5, n_4 + n_6\} \le (2W - 1)\} \\ &\quad \cup \{(W, W, 0, 0, 0, 0),\ (0, 0, W, 0, W, 0),\ (0, 0, 0, W, 0, W)\}. \end{aligned}$$

Similar to Lemmas 2(2), we have the following:

LEMMA 2′ (CONTAINMENT PROPERTY). *For switch blocks,* $S_W^* \supseteq I_W^{*\prime}$, *when* $\tilde{n_i} = W$, $1 \le i \le 6$.

The closed form for the cardinality of $I_W^{*\prime}$, $W > 0$, can be obtained by similar techniques as those used in the proof for Lemma 3:

$$\begin{aligned} |I_W^{*\prime}| &= |\{\vec{n} | \max\{n_1, n_2\} + \max\{n_3, n_5\} + \max\{n_4, n_6\} \le W\}| \\ &\quad - \left| \bigcup_{i=1}^{W-1} \{(i, i, 0, W - i, 0, W - i)\} \right| - \left| \bigcup_{i=1}^{W-1} \{(i, i, W - i, 0, W - i, 0)\} \right| \\ &= \binom{W+6}{6} + 3\binom{W+5}{6} + 2\binom{W+4}{6} + \binom{W+3}{6} - 2(W - 1) \\ &= \frac{1}{6!}\vec{A}^{*\prime} \cdot \vec{\omega}, \end{aligned}$$

where

$$\begin{aligned} \vec{A}^{*\prime} &= (8, 96, 500, 1440, 2372, 624, 2160), \\ \vec{\omega} &= (W^6, W^5, W^4, W^3, W^2, W, 1). \end{aligned}$$

Note that $|I_W^{*\prime}| = 1$ if $W = 0$. We have

$$|I_W^{*\prime}| = \begin{cases} 1, & W = 0; \\ \frac{1}{6!}\vec{A}^{*\prime} \cdot \vec{\omega}, & W > 0. \end{cases}$$

By Lemma 3, we have

$$|I_W| = \left\lfloor \frac{1}{6!}\vec{A} \cdot \vec{\omega} \right\rfloor,$$

Table I. Cardinalities and Performance Bounds for Specified W's

W	Switch matrix			Switch block	
	$\lvert I_W \rvert$	$\lvert I_W^* \rvert$	$\lvert I_W \rvert / \lvert I_W^* \rvert$	$\lvert I_W^{*\prime} \rvert$	$\lvert I_W \rvert / \lvert I_W^{*\prime} \rvert$
1	10	8	1.250	10	1.000
2	56	35	1.600	50	1.120
3	214	112	1.911	186	1.151
4	641	294	2.181	547	1.172
5	1,620	672	2.411	1,364	1.188
6	3,616	1,386	2.609	3,014	1.200
7	7,340	2,640	2.781	6,072	1.209
8	13,825	4,719	2.930	11,371	1.216
9	24,510	8,008	3.061	20,070	1.222
10	41,336	13,013	3.177	33,730	1.226
15	334,680	93,024	3.598	270,476	1.238
20	1,573,121	407,330	3.863	1,266,227	1.243
25	5,377,190	1,330,056	4.043	4,319,358	1.245
30	14,905,856	3,570,952	4.175	11,959,494	1.247
35	35,622,150	8,334,768	4.274	28,560,010	1.248
40	76,215,041	17,511,879	4.353	61,075,531	1.248

where $\vec{A} = (10, 120, 595, 1560, 2320, 1920, 720)$. The following lemma and theorems can be proved easily by the closed forms $|I_W|$ and $|I_W^{*\prime}|$.

LEMMA 3′ (MONOTONE PROPERTY). *$|I_W|/|I_W^{*\prime}|$ is a strictly increasing function of W, when $W \geq 0$.*

THEOREM 1′ (PERFORMANCE BOUND). *For switch blocks, $|S_W|/|S_W^*| \leq 5/4$.*

4.4 Summary of Performance Analysis

Table I lists the cardinalities and performance bounds associated with I_W, I_W^*, and $I_W^{*\prime}$.

5. EXTENSIONS

Section 4 assumed that routing requirements are uniformly distributed, and at most one switch can be used for a connection. In this section, we give the performance bounds for the cases with different distributions and extend the flow-network construction for the relaxed routing model which allows more than one switch to be used for a connection. Note that the distributions may be obtained either from circuit designs or routability specifications.

5.1 Nonuniform Distribution

The distribution of RRVs is given by $p(\vec{n})$, the probability that the RRV is $\vec{n}$. The *means* of a probability density function (or a probability mass function in discrete cases) over S_W and S_W^* are $m_{p(\vec{n})}$ and $m_{p(\vec{n})}^*$, respectively. For continuous cases, if $V_{P_W} \neq 0$ and $V_{P_W^*} \neq 0$, we have

$$m_{p(\vec{n})} = \frac{\int_{\vec{n} \in P_W} p(\vec{n})\, d\vec{n}}{V_{P_W}}$$

and

$$m^*_{p(\vec{n})} = \frac{\int_{\vec{n}\in P^*_W} p(\vec{n})\, d\vec{n}}{V_{P^*_W}}.$$

Similarly, for discrete cases, if $|S_W| \neq 0$ and $|S^*_W| \neq 0$,

$$m_{p(\vec{n})} = \frac{\sum_{\vec{n}\in S_W} p(\vec{n})}{|S_W|}$$

and

$$m^*_{p(\vec{n})} = \frac{\sum_{\vec{n}\in S^*_W} p(\vec{n})}{|S^*_W|}.$$

Let $|T_W|$ and $|T^*_W|$ be the number of occurrences that the RRVs are in S_W and S^*_W, respectively, and $\Gamma = m_{p(\vec{n})}/m^*_{p(\vec{n})}$. We have the following theorem.

THEOREM 2. *Given an arbitrary distribution of RRVs, for switch matrices,* $|T_W|/|T^*_W| \leq 5\Gamma$ *if* $m^*_{p(\vec{n})} \neq 0$.

PROOF. We show only the continuous case below:

$$\begin{aligned}\frac{|T_W|}{|T^*_W|} &= \frac{\int_{\vec{n}\in P_W} p(\vec{n}) d\vec{n}}{\int_{\vec{n}\in P^*_W} p(\vec{n}) d\vec{n}} \\ &= \frac{m_{p(\vec{n})} V_{P_W}}{m^*_{p(\vec{n})} V_{P^*_W}} \\ &= \frac{V_{P_W}}{V_{P^*_W}}\Gamma \\ &\leq 5\Gamma,\end{aligned}$$

where $V_{P_W}/V_{P^*_W} = 5$, by Theorem 1. □

Similarly, we have the following theorem.

THEOREM 2′. *Given an arbitrary distribution of RRVs, for switch blocks,* $|T_W|/|T^*_W| \leq 5\Gamma/4$ *if* $m^*_{p(\vec{n})} \neq 0$.

Note that $\Gamma = 1$ for uniform distributions and thus the corresponding performance bounds are 5 and 5/4 for switch matrices and switch blocks, respectively. For geometric distributions (discrete cases), exponential distributions (continuous cases) $\Gamma < 1$, and hence the respective bounds are less than 5 and 5/4 for switch matrices and switch blocks.

5.2 Relaxed Routing Model

In this subsection, we extend the network-flow analyzer to a more general switch-module routing model. Specifically, each net can use at most one switch

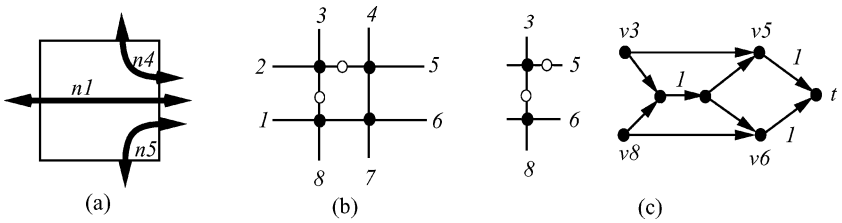

Fig. 11. Examples of network-flow construction based on the relaxed routing model. (a) Types of nets being considered. (b) A switch matrix instance. (c) Network construction for terminals 3 and 8.

for routing through a switch matrix (i.e., *1-switch routing*) in the previous discussion whereas a connection is considered *feasible* in the relaxed model as long as the connection contains no dogleg (jog). (In fact, jogs in a switch module rarely happen in practical FPGA/FPIC routing.) Hence, under the relaxed switch-matrix routing model, a connection can use up to three switches, one crossing and two separating switches. (Note that it is not allowed to use two crossing switches for a connection, since such a connection will introduce a jog inside a switch matrix.)

To analyze switch-module routability under the relaxed model, we need to modify the construction for the flow network shown in Figure 7 because the connections between terminals 3 and 6 or between terminals 8 and 5 become feasible now. Figure 11(c) illustrates the modification for the case where terminals k and l are both on source sides and are located on opposite sides of the same track with a separating switch between the terminals. (For other cases, the network construction remains the same and is thus not shown in Figure 11.) With the modification for the flow network construction, the network-flow-based analyzer described in Section 3 readily applies to the switch-module analysis under the relaxed routing model.

6. EXPERIMENTAL RESULTS

We did the following experiments:

(1) empirical performance of the flow analyzer on switch matrices;
(2) correlation between switch-matrix and chip-level routability, and the accuracy of the flow analyzer in evaluating the chip-level routability;
(3) Running times for computing all routable RRV's for a switch matrix (useful for routing [Chang et al. 1994]) by the flow analyzer and an exact analyzer.

The flow analyzer was implemented in C on a SUN SPARC 5 workstation. In Experiment (1), we measured the performance of the flow analyzer. We first applied the switch-matrix design algorithm in Zhu et al. [1993] to generate switch matrices. The sizes of the switch matrices are $W = 5$, 10, 15, and 20. For each W, we generated $5W$ switch matrices with the number of crossing switches N ranging from $2W$ to $7W - 1$, except the case for $W = 5$ where N ranges from $2W$ to $5W$ (since there are at most W^2 crossing switches in a switch matrix). (Figures 12, 13, and 14 show three switch matrices with $W = 20$ and $N = 80$ designed by a random generation, [Zhu et al. 1993; Chang et al.

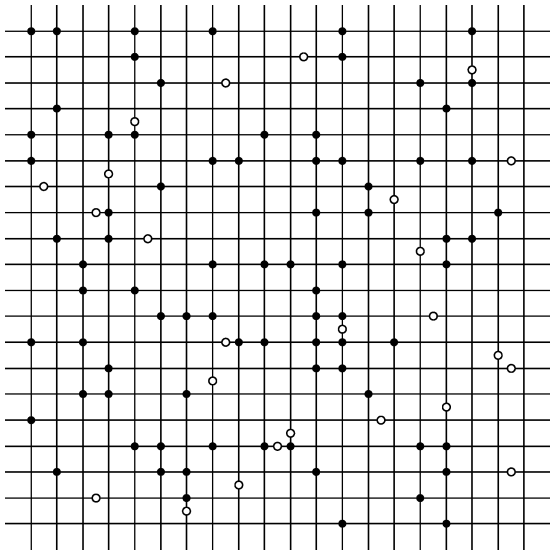

Fig. 12. A random switch matrix with $W = 20$ and $N = 80$.

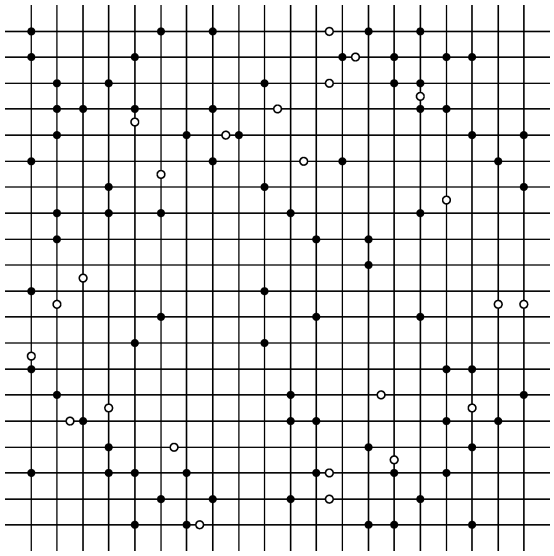

Fig. 13. A switch matrix with $W = 20$ and $N = 80$ designed by Zhu et al. [1993].

1995b], respectively.) For each switch matrix designed by Zhu et al. [1993], we analyzed its routability by the flow analyzer and the ILP analyzer used in Thakur et al. [1997] based on 100 randomly generated RRVs in I_W (see Section 4.2 for the definition of I_W); that is, trivially infeasible RRVs for both analyzers were not considered. The results are shown in Table II and Figure 15. Table II gives the average-, best-, and worst-case performance bounds for each W. In Figure 15, the percentage of routable $\vec{n}$'s, represented by the vertical

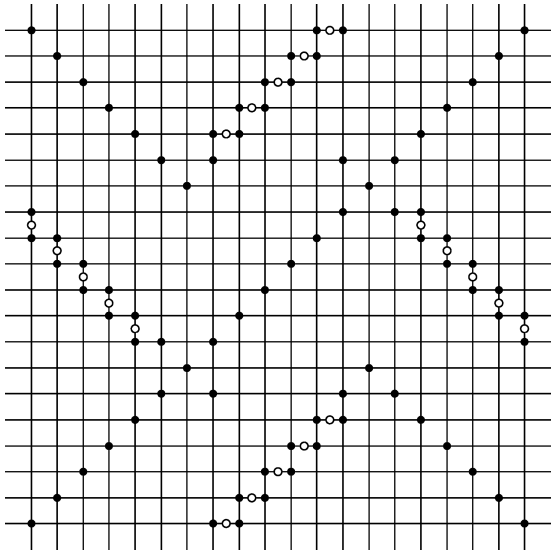

Fig. 14. A switch matrix with $W = 20$ and $N = 80$ designed by Chang et al. [1995b].

Table II. The Average-, Best-, and Worst-Case Performance Bounds of the Flow Analyzer

	Experimental performance bounds		
W	Average case	Best case	Worst case
5	1.024	1.000	1.091
10	1.062	1.000	1.139
15	1.091	1.000	1.184
20	1.111	1.000	1.216

axis, is plotted as a function of N, denoted by the horizontal axis, for the case where $W = 20$. They show that the flow analyzer has the respective average-, best-, and worst-case ($N = 57$) performance bounds 1.111, 1.000 (optimal), and 1.216 away from the optima. Note that they are much better than the theoretical bound, 3.863, for $W = 20$ (and the bound 5 for arbitrary W's). More significantly, as shown in Table III (and Figure 15), the flow analyzer ranked the routability of the switch matrices in very close to the same order as the ILP analyzer; the high fidelity makes the flow analyzer a reliable approximation analyzer.

In Experiment (2), we explored the correlation between switch-matrix and chip-level routability, and the accuracy of the flow analyzer in evaluating the chip-level routability. We first randomly generated connections on a 15×15 (number of logic modules) FPGA. For this purpose, we assumed that the number of pins on each logic module is unlimited (so that we could test denser circuits). We then used the detailed router developed by Wu and Chang [1998] to route the connections on the FPGA using the three types of 20×20 (number of terminals) switch matrices of various numbers of switches designed by the random generation [Zhu et al. 1993; Chang et al. 1995b]. As shown in Table IV, the

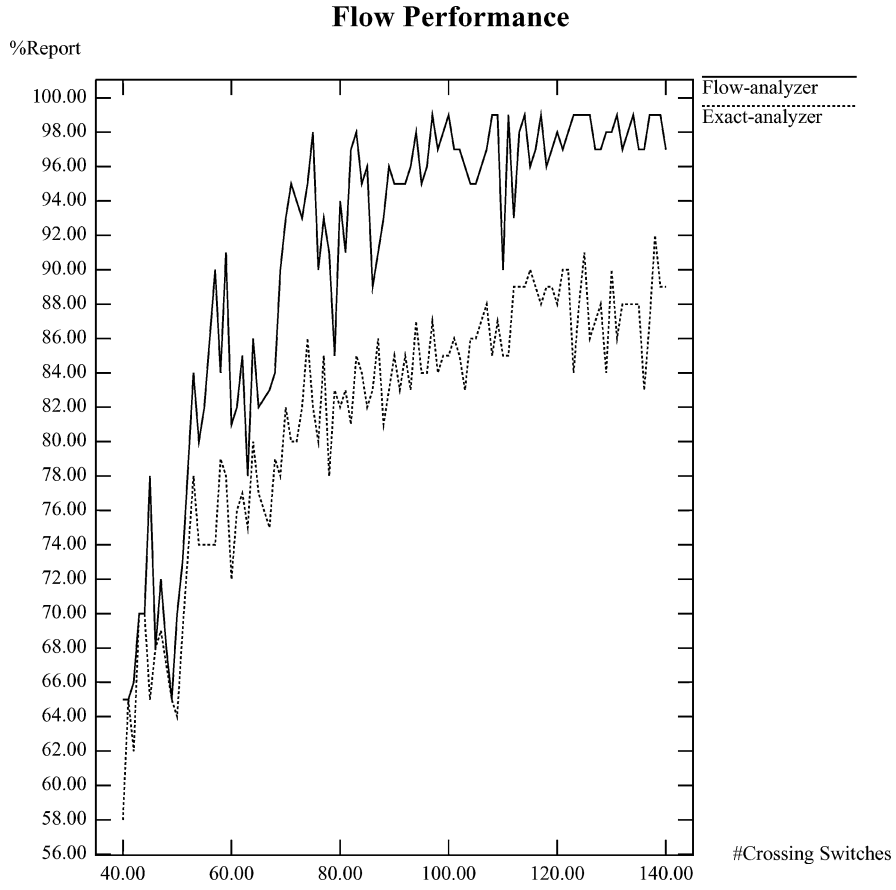

Fig. 15. Results for Experiment (1): performance of the flow analyzer.

Table III. Correlation Between the Results Reported by the ILP and the Flow Analyzers

# Crossing switches	Percentage of routable RRVs	
	Exact analyzer	Flow analyzer
40	58	65
50	64	70
60	72	81
70	82	93
80	82	94
90	84	95
100	86	99

switch matrices with higher routing capacity evaluated by the ILP analyzer and the flow analyzer typically result in better chip-level routability. Further, the ranking for the chip-level routability is typically the same as that reported by the flow and the ILP analyzers for the associated switch matrices (see Table III; the numbers reported by the flow analyzer are larger than those reported by an

Table IV. Percentage of Routable Connections on a 15 × 15 FPGA Using Three Types of 20 × 20 Switch Matrices with Various Numbers of Switches

# Crossing switches	Switch architecture	% of routable RRVs		Number of connections				
		ILP analyzer	Flow analyzer	200	300	400	600	800
40	Random	39	46	85.0%	69.3%	63.5%	43.7%	32.6%
	Zhu et al. [1993]	58	65	98.0%	92.0%	70.0%	50.2%	38.5%
	Chang et al. [1995b]	76	79	99.5%	91.0%	72.3%	56.8%	44.4%
60	Random	57	65	86.0%	72.0%	65.3%	45.5%	35.8%
	Zhu et al. [1993]	72	81	98.5%	94.0%	72.3%	51.3%	39.1%
	Chang et al. [1995b]	85	94	100.0%	98.3%	79.8%	61.2%	49.8%
80	Random	74	81	90.0%	79.7%	66.5%	47.8%	36.9%
	Zhu et al. [1993]	82	94	100.0%	96.3%	73.0%	51.7%	40.6%
	Chang et al. [1995b]	93	99	100.0%	100.0%	83.5%	66.0%	50.3%
100	Random	77	82	97.0%	88.0%	71.0%	48.7%	39.6%
	Zhu et al. [1993]	86	99	100.0%	98.0%	73.8%	52.3%	42.4%
	Chang et al. [1995b]	94	100	100.0%	100.0%	84.5%	66.3%	50.9%

Table V. Running Times Required for Computing All Routable $\bar{n}$'s on a Corresponding Switch Matrix

Switch matrix	W	N	Running time		Running-time ratio
			ILP analyzer	Flow analyzer	time(ILP)/time(flow)
Module5-25	5	25	15 min	1 min	15
Module10-50	10	50	161 min	7 min	23
Module15-75	15	75	2088 min	28 min	75
Module20-100	20	100	>28000 min	97 min	>288

exact analyzer because the flow analyzer overestimated the routability). This result reveals that switch-module routability is highly correlated to that of the chip-level routability, and the flow analyzer has the high fidelity in evaluating the FPGA chip-level routability.

In Experiment (3), we measured the run time required to compute the routing capacity of a designed switch matrix. This is applicable to FPGA routing [Chang et al. 1994]. The number of crossing switches N in each switch matrix was $5W$. We used the algorithm presented in Chang et al. [1994] to obtain all routable RRVs on a switch matrix; it computed $O(W^6)$ SMAPs with various RRVs. Table V lists the run times for the ILP and the flow analyzers. It shows that the flow analyzer runs much more efficiently than the ILP analyzer.

7. CONCLUDING REMARKS

We have shown that the network-flow-based routability analyzer is efficient and highly accurate. It has provably good performance with constant bounds away from the optima for two types of switch modules. However, whether the analysis problem SMAP is NP-complete is still open, and future research involves further investigation of the SMAP.

To explore the routability of an FPGA/FPIC chip, we adopted in this paper the bottom-up approach by considering a single switch module first. The methodology is mainly motivated by the golden rule "optimize the common cases" [Hennessy and Patterson 1996], which is the key to contemporary computer designs. As mentioned in the introduction, for *real* applications, most connections

are short (i.e., *the common cases*), independent of the sizes of FPGAs/FPICs. Therefore, it is of particular importance to consider the architecture of a single switch module. Future work lies in the exploration of the routing capacity for multiple switch modules in series and the interaction of switch modules and pin-to-track connection ones [Betz and Rose 2000].

ACKNOWLEDGMENTS

The authors would like to thank Dr. Glenn Lai for many helpful suggestions, Shashidhar Thakur for providing the ILP analyzer, and anonymous reviewers for very constructive comments.

REFERENCES

ACTEL CORP. 1996. *FPGA Data Book and Design Guide*. Actel Corp., Sunnyvale, CA.

APTIX, INC. 1992. *FPIC AX1024D*. Preliminary Data Sheet, Aptix, Inc., San Jose, CA.

BETZ, V., ROSE, J., AND A. MARQUARDT 1999. *Architecture and CAD for Deep-Submicron FPGAs*. Kluwer Academic Publishers, Boston, MA.

BETZ, V. AND ROSE, J. 2000. Automatic generation of FPGA rotuing architectures from high-level descriptions. In *Proceedings of ACM/SIGDA International Symposium on Field Programmable Gate Arrays* (Monterey, CA, Feb.) 175–184.

BHAT, N. AND HILL, D. 1992. Routable technology mapping for LUT FPGAs. In *Proceedings of IEEE International Conference on Computer Design, VLSI in Computers and Processors* (Cambridge, MA, Oct.). 95–98.

BROWN, S. D., FRANCIS, R. J., ROSE, J., AND VRANESIC, Z. 1992a. *Field-Programmable Gate Arrays*. Kluwer Academic Publishers, Boston, MA.

BROWN, S. D., ROSE, J., AND VRANESIC, Z. 1992b. A detailed router for field-programmable gate arrays. *IEEE Trans. Comput.-Aided Des. Integ. Circ. Syst. 11*, 5 (May), 620–627.

BROWN, S. D., ROSE, J., AND VRANESIC, Z. G. 1993. A stochastic model to predict the routability of field-programmable gate arrays. *IEEE Trans. Comput.-Aided Des. 12*, 12 (Dec.), 1827–1838.

CHANG, Y.-W., THAKUR, S., ZHU, K., AND WONG, D. F. 1994. A new global routing algorithm for FPGAs. In *Proceedings of IEEE/ACM International Conference on Computer-Aided Design* (San Jose, CA, Nov.). 356–361.

CHANG, Y.-W., WONG, D. F., AND WONG, C. K. 1995a. FPGA global routing based on a new congestion metric. In *Proceedings of IEEE International Conference on Computer Design* (Austin, TX, Oct.). 372–378.

CHANG, Y.-W., WONG, D. F., AND WONG, C. K. 1995b. Design and analysis of FPGA/FPIC switch modules. In *Proceedings of IEEE International Conference on Computer Design, VLSI in Computers and Processors* (Austin, TX, Oct.). 394–401.

CHANG, Y.-W., WONG, D. F., AND WONG, C. K. 1996a. Universal switch modules for FPGA design. *ACM Trans. Des. Automat. Electron. Syst. 1*, 1 (Jan.), 80–101.

CHANG, Y.-W., WONG, D. F., AND WONG, C. K. 1996b. Universal switch-module design for symmetric-array FPGAs. In *Proceedings of ACM/SIGDA International Symposium on Field Programmable Gate Arrays* (Monterey, CA, Feb.). 80–86.

GAMAL, A. E., GREENE, J., REYNERI, J., ROGOYSKI, E., ELAYAT, K. A., AND MOHSEN, A. 1989. An architecture for electrically configurable gate arrays. *IEEE J. Solid-State Circ. 24*, 2, 394–398.

GAREY, M. AND JOHNSON, D. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, San Francisco, CA.

GRAHAM, R. L., KNUTH, D. E., AND PATASHNIK, O. 1989. *Concrete Mathematics*. Addison-Wesley, Reading, MA.

GUO ET AL. 1992. A 1024 pin universal interconnect array with routing architecture. In *Proceedings of IEEE Custom Integrated Circuits Conference* (San Diego, CA, April). pp. 4.5.1–4.5.4.

HENNESSY, J. L. AND PATTERSON, D. A. 1996. *Computer Architecture: A Quantitative Approach*, 2nd ed. Morgan Kaufmann, San Mateo, CA.

LEMIEUX, G. AND BROWN, S. D. 1993. A detailed router for allocating wire segments in field-programmable gate arrays. In *Proceedings of ACM/SIGDA Physical Design Workshop* (Lake Arrow, CA, April). 215–226.

LEMIEUX, G., BROWN, S. D., AND VRANESIC, D. 1997. On two-step routing for FPGAs. In *Proceedings of ACM International Symposium on Physical Design* (Napa Valley, CA, April 14–16). 60–66.

LUCENT TECHNOLOGIES 1996. *Field-Programmable Gate Arrays Data Book*. Lucent Technologies, Murray Hill, NJ.

MARPLE, D. AND COOKE, L. 1992. An MPGA compatible FPGA architecture. In *Proceedings of ACM International Symposium on Field-Programmable Gate Arrays* (Monterey, CA, Feb.). 39–44.

ROSE, J. AND BROWN, S. 1991. Flexibility of interconnection structures for field-programmable gate arrays. *IEEE J. Solid-State Circ. 26*, 3, 277–282.

SHYU, M., CHANG, Y.-D., WU, G.-M. AND CHANG, Y.-W. 2000. Generic universal switch blocks. *IEEE Trans. Comput. 49*, 4 (April), 348-359.

SLEATOR, D. D. AND TARJAN, R. E. 1983. A data structure for dynamic trees. *J. Comput. Syst. Sci. 24*, 362–391.

THAKUR, S., CHANG, Y.-W., WONG, D. F., AND MUTHUKRISHNAN, S. 1997. Algorithms for an FPGA switch module routing problem with application to global routing. *IEEE Trans. Comput.-Aided Des. Integr. Circ. Syst. 16*, 1 (Jan.), 32–46.

TRIMBERGER, S. 1994. *Field-Programmable Gate Array Technologies*. Kluwer Academic Publishers, Boston, MA.

TRIMBERGER, S. AND CHENE, M. 1992. Placement-based partitioning for lookup-table-based FPGA's. In *Proceedings of IEEE International Conference Computer Design, VLSI in Computers and Processors* (Cambridge, MA, Oct.). pp. 91–94.

WILTON, S. 1997. Architecture and Algorithms for Field-Programmable Gate Arrays with embedded Memories. Ph.D. dissertation, University of Toronto, Toronto, Ont. Canada.

WU, G.-M. AND CHANG, Y.-W. 1998. Switch-matrix architecture and routing for FPDs. In *Proceedings of ACM International Symposium on Physical Design* (Monterey, CA, April 14–16). 158–163.

WU, G.-M. AND CHANG, Y.-W. 1999. Quasi-universal switch matrices for FPD design. *IEEE Trans. Comput. 48*, 10 (Oct.). 1107–1122.

WU, Y.-L., TSUKIYAMA, S., AND MAREK-SADOWSKA, M. 1996. Graph based analysis of 2-D FPGA routing. *IEEE Trans. Comput.-Aided Des. Integr. Circ. Syst. 15*, 1 (Jan.), 33–44.

XILINX, INC. 1996. *The Programmable Logic Data Book*. Xilinx, Inc., San Jose, CA.

ZHU, K., WONG, D. F., AND CHANG, Y.-W. 1993. Switch module design with application to two-dimensional segmentation design. In *Proceedings of IEEE/ACM International Conference on Computer-Aided Design* (San Jose, CA, Nov. 7–11). pp. 481–486.

Received August 2000; revised April 2002; accepted September 2002