A Graph-Theoretic Sufficient Condition for FPGA Switch-Module Routability *

Yao-Wen Chang[1], *D. F. Wong*[2], *and C. K. Wong*[3]

[1] *Dept. of Electrical Engineering & Graduate Institute of Electronic Engineering, National Taiwan University, Taiwan*
[2] *Dept. of Electrical and Computer Engineering, University of Illinois, Urbana, IL 61801, USA*
[3] *Dept. of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong, China*

Abstract

We consider in this paper an FPGA 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 to the analysis of switch modules, this problem is applicable to the routability evaluation of FPGA switch modules, the switch-module design for FPGA's, and FPGA routing. We present a graph-theoretic sufficient condition for the routability analysis problem. The implications of the condition are as follows: (1) there exist classes of polynomial-time approximation algorithms for the analysis problem; (2) there exist classes of switch-module architectures on which the analysis problem can be solved in polynomial time.

1 Introduction

In general, an FPGA architecture consists of an array of logic modules which can be connected by general routing resources [4, 2]. Figures 1(a) and (b) illustrate two major FPGA architectures: symmetric-array-based and row-based FPGA's. 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 channels and their intersection areas. The intersection area of a horizontal and a vertical channels 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. A logic circuit is implemented in an FPGA by partitioning logic into individual logic modules and then interconnecting the modules by programming switches.

The works by [3, 20] 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 FPGA's. Therefore it is desirable to facilitate routing in the design of FPGA's. Switch modules are the most important component of the routing resources in FPGA's. Studies by [6, 10, 11, 13,

*This work was supported in part by the National Science Council of Taiwan ROC by Grant No. NSC 88-2215-E-009-064.

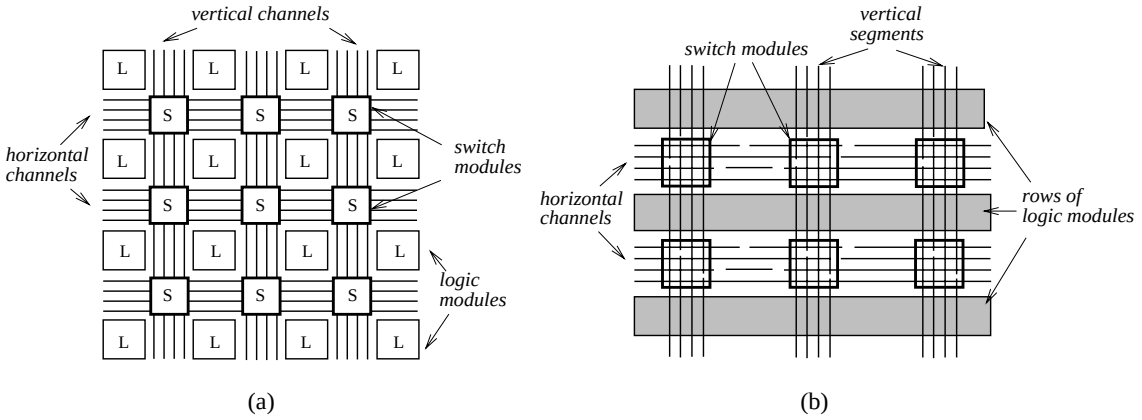

Figure 1: Two major FPGA architectures. (a) Symmetric-array-based architecture. (b) Row-based architecture.

17, 21, 22] have shown that the higher the routability of the switch modules, 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 nets are short; for example, about 60% (90%) of nets in the CGE [5], the SEGA [15], and the VPR [2] benchmark circuits are routed through no more than two (five) switch modules, independent of FPGA sizes. Further, the switch modules used in current commercial FPGA's are usually homogeneous. Thus the routability of a single switch module plays an important role in overall FPGA routing.
- Most logic-module pins are logically equivalent; 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 switch-module analysis.

The FPGA 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 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 FPGA's were first studied experimentally by Rose and Brown [17]. A theoretical study of flexibility and routability was later presented based on a stochastic model [6]. The primary conclusion in both of the studies in [6, 17] 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. 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. in [10] first presented the concept of *universal switch modules* and developed a class of universal switch modules. By definition, a universal switch module can accommodate any set of two-pin connections on a switch module as long as the number of connections routing through each side of the module does not exceed its size (the number of terminals on a side). Fan et al. in [12, 13] extended the universal switch modules to

hyper-universal switch modules for multi-terminal nets based on a graph-theoretic formulation. Wu and Chang in [22] studied another type of switch-module architecture (called *switch matrices*), showed the non-existence of the universality for switch matrices, and presented *quasi-universal switch matrices* which have the routability asymptotically close to the universal switch modules. All of those works showed theoretically and experimentally that switch modules with higher routability usually lead to better area performance, which confirms the earlier findings by [6, 17].

In this paper we consider more general switch-module models and study a routability 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 routing requirements on the switch module. A precise definition of the problem will be given in the next section. As a fundamental problem to the analysis of FPGA switch modules, this problem is applicable to

- routability evaluation of FPGA switch modules [7, 8, 19, 25],
- switch-module design for FPGA's [9, 12, 13, 25], and
- FPGA routing [7, 8, 19, 23].

To solve this analysis problem, Thakur *et al.* [19] proposed an exact algorithm based on integer linear programming (ILP). However, since ILP is NP-complete [14], this ILP-based algorithm in the worst case is computationally expensive. Instead of resorting to the ILP, Zhu *et al.* [25] and Chang et al. [11] presented a network-flow-based approximation algorithm for the analysis problem; the algorithm is approximate and tends to overestimate routability. Whether the *general* analysis problem can be solved in polynomial time is still open.

In this paper, we present a graph-theoretic sufficient condition for the analysis problem. The implications of this condition are as follows: (1) there exist classes of efficient approximation algorithms for the analysis problem; (2) there exist classes of switch-module architectures on which the analysis problem can be solved efficiently.

The organization of the rest of this paper is as follows. Section 2 gives the formal problem definition. Section 3 presents the sufficient condition for switch-module routability and its implications. Finally, the concluding remarks are given in Section 4.

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 2). In a switch block (see Figure 2(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 *non-interacting*, unless they share a terminal. Switch blocks are used in symmetric-array-based FPGA's for connections between single-length lines or between double-length lines [24]. A switch matrix (see Figure 2(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 non-interacting. If a separating switch on a track is "OFF," the track is split into two electrically non-interacting

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 2(b), the crossing switches are represented by solid circles and the separating switches by hollow circles. Switch matrices are used in various symmetric-array FPGA's [24] and row-based FPGA's [1, 16].

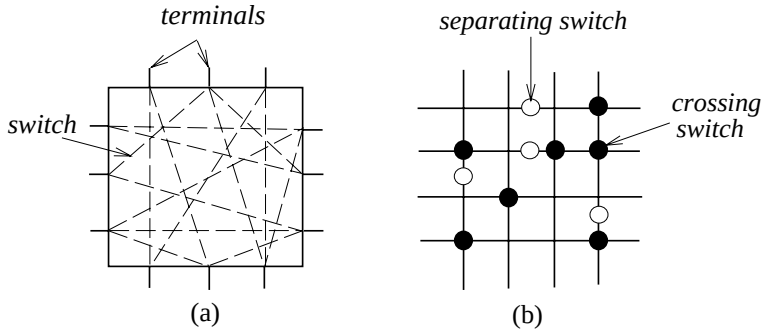

Figure 2: (a) A switch block. (b) A switch matrix.

A *connection* is an electrical path between two terminals on different sides of a switch module. Connections can be of six types, each of which is characterized by two sides of a module, as shown in Figure 3(a). 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 assume that at most one switch can be used, i.e., programmed to be "ON," by a connection. 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. Based on this assumption, for switch matrices, only straight connections can use separating switches.

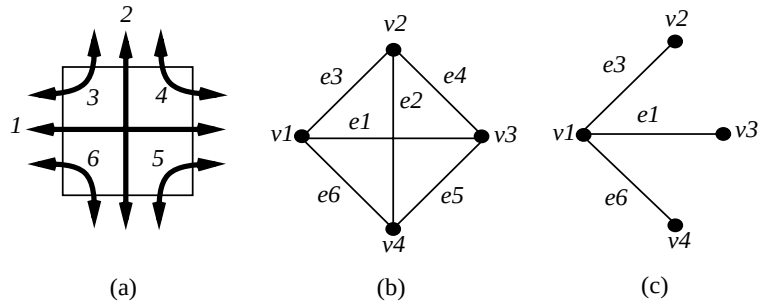

Figure 3: (a) Six types of connections. (b) $G_{\vec{n}}$, $\vec{n} = (n_1, \ldots, n_6), n_i \neq 0$, $1 \leq i \leq 6$. (c) $G_{\vec{n}}$, where $\vec{n} = (n_1, 0, n_3, 0, 0, n_6)$, $n_1, n_3, n_6 \neq 0$.

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 RRV $(0, 1, 0, 1, 1, 1)$ is routable on the switch matrix shown in Figure 4(a) by programming the switches 1, 2, 3, and 7 to be ON, and a routing solution is illustrated by the thick lines; on the other hand, the RRV $(0, 0, 2, 1, 0, 0)$ is not routable on the switch matrix. For the switch block shown in Figure 4(b),the

RRV $(2,1,0,0,0,0)$ is routable, but the RRV $(0,0,2,0,0,0)$ is not.

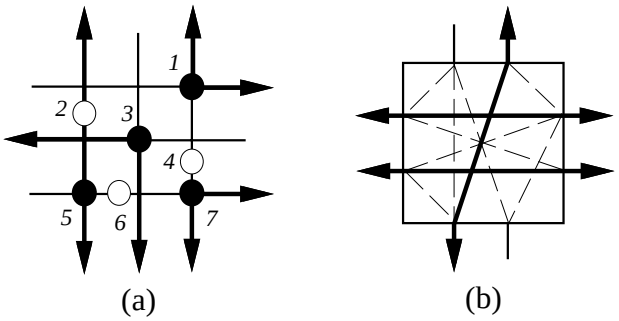

Figure 4: (a) A switch-matrix routing. (b) A switch-block routing.

A *routing requirement graph* (*RRG*) $G_{\vec{n}}$ is an undirected graph with respect to the RRV $\vec{n}$. Each node in $G_{\vec{n}}$ corresponds to one of the four sides of a switch module. An undirected edge between side a and side b means that the corresponding component of $\vec{n}$ characterized by sides a and b is non-zero. For instance, Figures 3(b) and (c) give two instances of RRG's for $\vec{n} = (n_1, \ldots, n_6), n_i \neq 0$, $1 \leq i \leq 6$, and $\vec{n} = (n_1, 0, n_3, 0, 0, n_6)$, $n_1, n_3, n_6 \neq 0$, respectively.

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?

3 A Sufficient Condition for the SMAP

In this section, we first consider four classes of network-flow-based techniques for switch-module analysis. Based on these techniques, we then induce a graph-theoretic sufficient condition for the SMAP.

3.1 Network-Flow-Based Algorithms

Consider the following four categories of SMAP's which can be solved in polynomial time using transformations to the maximum network-flow problems. In each category, at least three out of the six components of the RRV $\vec{n}$ are zero.

1. SMAP's 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 non-zero, and the remaining components are zero, i.e., the three edges in the corresponding $G_{\vec{n}}$ share one vertex. For example, as shown in Figure 5(a), SMAP with $\vec{n} = (n_1, 0, n_3, 0, 0, n_6)$, $(0, n_2, n_3, n_4, 0, 0)$, etc.

2. SMAP's with $\vec{n}$'s in which the edges in $G_{\vec{n}}$ corresponding to any two types of bent connections do not share a vertex, and there exists one and only one edge corresponding to one type of straight connections. For example, SMAP with $\vec{n} = (n_1, 0, n_3, 0, n_5, 0)$, etc. (See Figure 5(b).)

3. SMAP's with $\vec{n}$'s in which there are at most three edges in $G_{\vec{n}}$ corresponding to any three types of bent connections. For example, SMAP with $\vec{n} = (0, 0, n_3, n_4, 0n_6)$, etc. (See Figure 5(c).)

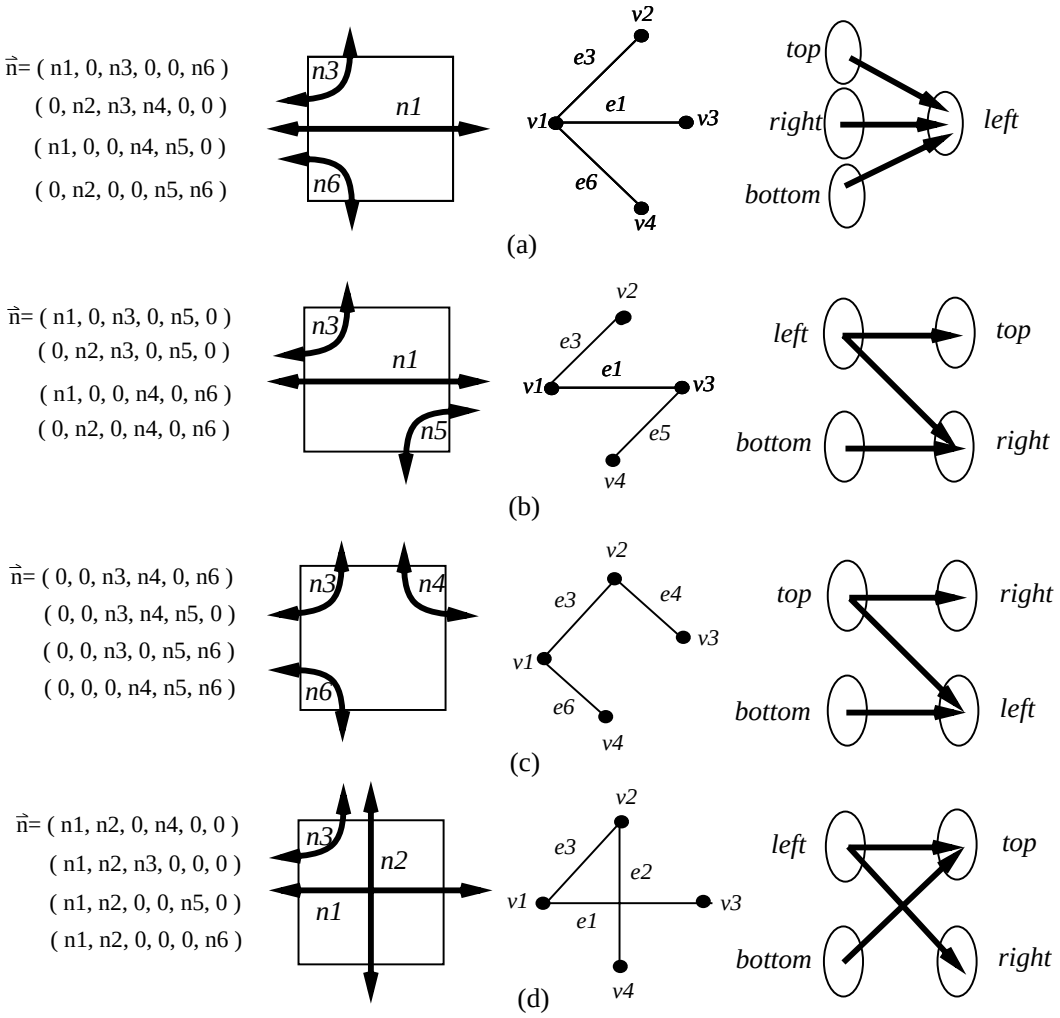

Figure 5: (a) α-network. (b) β-network. (c) γ-network. (d) δ-network. From left to right: RRV's, representative types of nets, $G_{\vec{n}}$, and outline of the network-flow construction.

4. SMAP's with $\vec{n}$'s in which there exist at most one edge in $G_{\vec{n}}$ corresponding to one type of bent connections, and two edges corresponding to the two types of straight connections. For example, SMAP with $\vec{n} = (n_1, n_2, 0, n_4, 0, 0)$, etc. (See Figure 5(d).)

For each of the four categories of SMAP's, we can reduce it to computing the maximum flow in a network. We refer to these networks corresponding to the cases in categories 1, 2, 3, and 4 as α-, β-, γ-, and δ-*networks*, respectively. See Figure 5 for these cases and their classification. The α-networks were first proposed in [25]. Given a switch module and an RRV $\vec{n}$, a network is constructed as follows. (Figure 6(d) shows the network for the switch matrix shown in Figure 6(a).) Pick one side of the switch module as the *sink side* and make the others *source sides*; in Figure 6(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, ..., $4W$,

starting from the bottom-most terminal on the left side and proceeding clockwise (see Figure 6(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 Figure 6(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 6(a)), connect v_k and v_l as shown in Figure 6(b) if there is no separating switch on the track, or connect them as illustrated in Figure 6(c) if there is a separating switch on the track. Note that terminals 4 and 7 in Figure 6(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 6(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 6(c).) Note that, as mentioned earlier, at most one switch can be used for routing through a switch module.

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 [18]. 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.

The construction of β-networks and γ-networks was presented in [19]. We generalize this approach to the construction of δ-networks. An example of the construction of a δ-network is shown in Figures 6(a) and (e). Unlike that of α-networks, the construction of β-, γ-, and δ-networks requires two sides as sources and the others as sinks. For the δ-network example shown in 6(e), its corresponding network-flow problem is as follows: Given a network, is there a feasible flow where source s_{13} supplies a flow of $n_1 + n_3$, source s_2 supplies a flow of n_2, sink t_{23} receives a flow of $n_2 + n_3$, and sink t_1 receives a flow of n_1? The problems associated with β- and γ-networks are similarly defined.

3.2 The Sufficient Condition and Its Implications

Note that the $G_{\vec{n}}$'s with respect to the above four categories of SMAP's are acyclic (see Figure 5). Since there are only four vertices in an RRG, it is easy to see that the four $G_{\vec{n}}$'s contain all possible acyclic RRG's. This leads to the following results:

Lemma 1 *$\forall \vec{m}\ \exists \vec{n}$, $G_{\vec{m}}$ is acyclic $\Longrightarrow$ $G_{\vec{m}}$ is a subgraph of some $G_{\vec{n}}$, where $\vec{n}$ is associated with an α-, β-, γ-, or δ-network.*

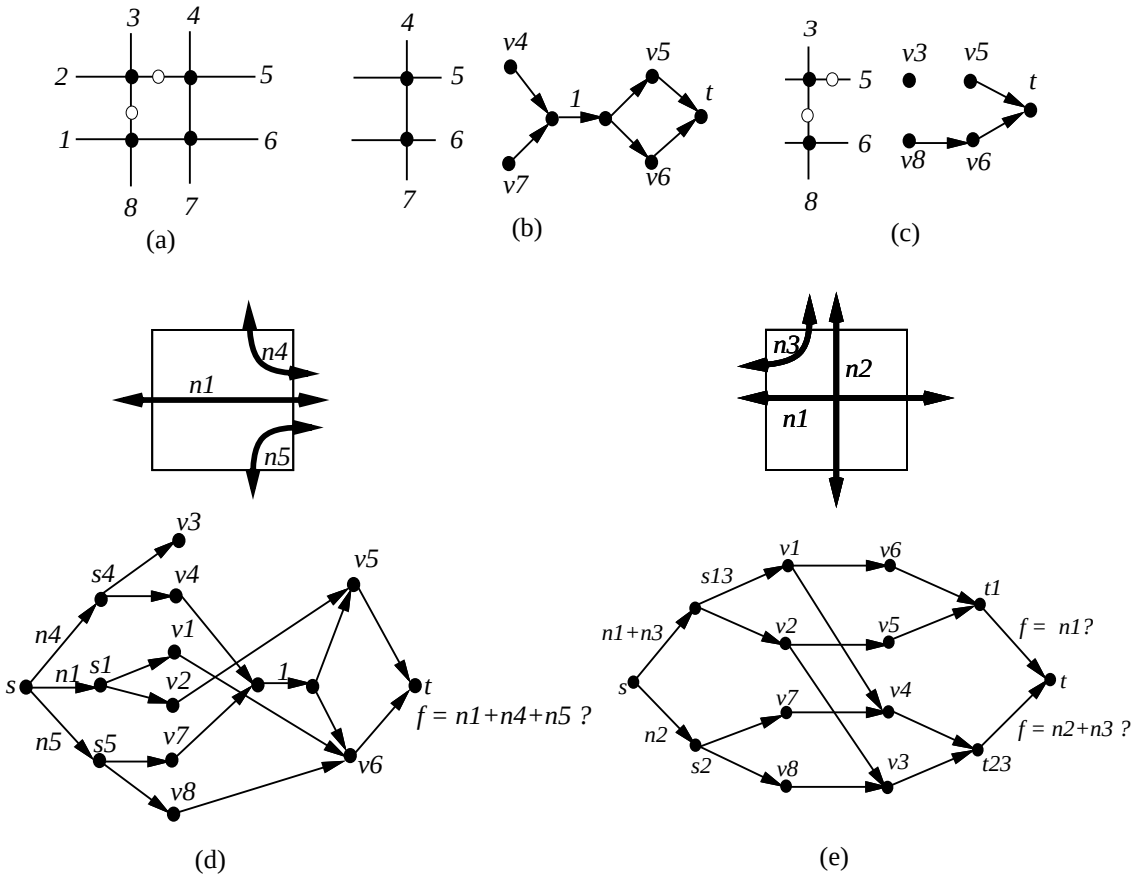

Figure 6: Examples of network-flow construction. (a) A switch matrix instance. (b)(c) Construction of an α-network. (d) A corresponding α-network. (e) A corresponding δ-network.

Theorem 1 *The SMAP with $\vec{n}$ on a switch module of size W and with N switches can be solved in $O(W(N+W)\log W)$ time if $G_{\vec{n}}$ is acyclic.*

Proof: Immediately from Lemma 1 and the above discussion. If $G_{\vec{n}}$ is acyclic, the SMAP with $\vec{n}$ can be solved by reducing it to computing the maximum flow in an α-, β-, γ-, or δ-network in $O(W(N+W)\log W)$ time [18]. □

Based on Theorem 1, we can obtain a class of approximation algorithms for the SMAP. Given a switch module M and an RRV $\vec{n}$, for example, we can pick one side of M as the sink and construct an α-network. If there exists no feasible flow on the network, M is not $\vec{n}$-routable; otherwise, pick another side of M as the sink and repeat the above 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 7.) Similarly, for β- and δ-networks, we also need at most four iterations to analyze all six types of nets routed through a switch module. Therefore, α-, β-, and δ-networks can be used to analyze switch-module routability.

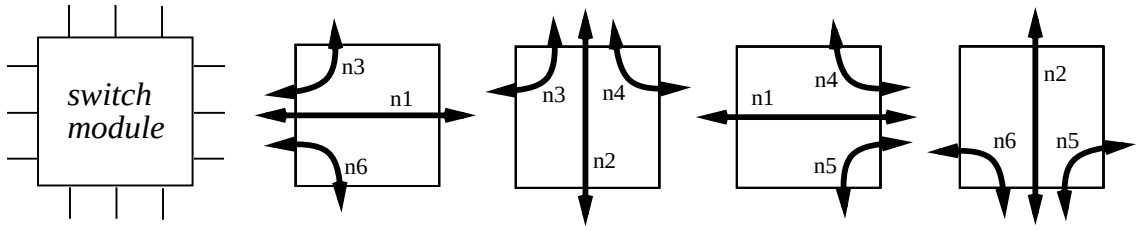

Figure 7: The four α-networks used to analyze the six types of connections for a switch module.

Also, Theorem 1 implies that the SMAP on some special switch-module architectures can be solved efficiently; those architectures include switch matrices with no separating switches (e.g., the switch matrices shown in Figure 8(a)), switch blocks with the connection topologies corresponding to the cases listed in Section 3.1 (e.g., the switch blocks shown in Figures 8(b) and (c)), etc.

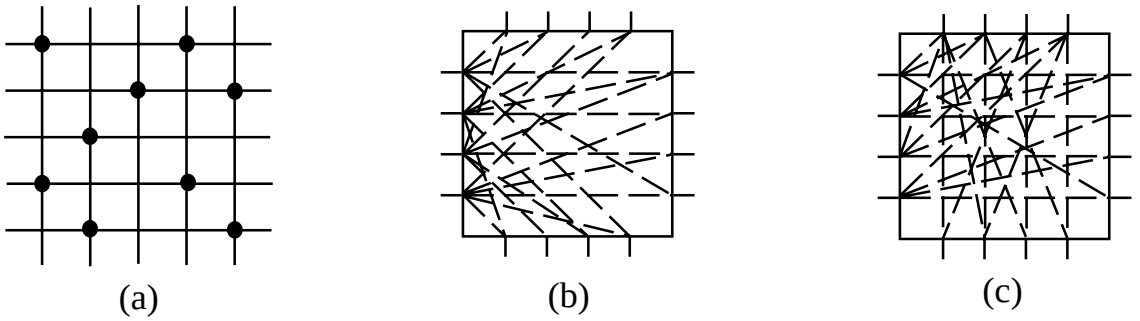

Figure 8: Example switch-module architectures on which the SMAP is solvable in polynomial time. (a) A switch matrix with no separating switches. (b) A switch block with only type-1, -3, and -6 switches. (c) A switch block with only type-1, -2, and -3 switches.

4 Concluding Remarks

We have considered four types of flow networks which can be used to solve four classes of special SMAP's in polynomial time. The four types of networks induce a graph-theoretic sufficient condition for the SMAP's.

Whether the general SMAP is NP-complete is still open. It is our conjecture that the SMAP becomes NP-complete if the corresponding $G_{\vec{n}}$ is cyclic.

References

[1] Actel Corp., *FPGA Data Book and Design Guide*, 1996.

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

[3] N. Bhat and D. Hill, "Routable technology mapping for LUT FPGAs," in *Proc. IEEE Int. Conf. Computer Design*, pp. 95–98, Cambridge, MA, Oct. 1992.

[4] S. D. Brown, R. J. Francis, J. Rose, and Z. G. Vranesic, *Field-Programmable Gate Arrays*, Kluwer Academic Publishers, Boston, MA, 1992.

[5] S. D. Brown, J. Rose, and Z. G. Vranesic, "A detailed router for field-programmable gate arrays," *IEEE Trans. Computer-Aided Design*, vol. 11, pp. 620–628, May 1992.

[6] S. D. Brown, J. Rose, and Z. G. Vranesic, "A stochastic model to predict the routability of field-programmable gate arrays," *IEEE Trans. Computer-Aided Design*, vol. 12, no. 12, pp. 1827–1838, Dec. 1993.

[7] Y.-W. Chang, S. Thakur, K. Zhu, and D.F. Wong, "A new global routing algorithm for FPGAs," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, pp. 380–385, San Jose, CA, Nov. 1994.

[8] Y.-W. Chang, D.F. Wong, and C. K. Wong, "FPGA global routing based on a new congestion metric," in *Proc. IEEE Int. Conf. Computer Design*, Austn, TX, Oct. 1995.

[9] Y.-W. Chang, D. F. Wong, and C. K. Wong, "Design and analysis of FPGA switch modules," in *Proc. IEEE Int. Conf. Computer Design*, pp. 394–401, Austin, TX, Oct. 1995.

[10] Y.-W. Chang, D. F. Wong, and C. K. Wong, "Universal switch modules for FPGA design," *ACM Trans. Design Automation of Electronic Systems*, vol. 1, no. 1, pp. 80–101, Jan. 1996.

[11] Y.-W. Chang, K. Zhu, D. F. Wong, G.-M. Wu, and C. K. Wong, "Analysis of FPGA/FPIC switch modules," *ACM Trans. on Design Automation of Electronic Systems*, vol. 8, no. 1, January 2003.

[12] H. Fan, J. Liu, and Y.-L. Wu, "General models for optimum arbitrary-dimension FPGA switch box designs," *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, pp. 93–98, San Jose, CA, Nov. 2000.

[13] H. Fan, J. Liu, and Y.-L. Wu, "On optimum switch box designs for 2-D FPGAs," *Proc. ACM/IEEE Design Automation Conference*, pp. 203–208, Las Vegas, NV, June 2001.

[14] M. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco, 1979.

[15] 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, pp. 215–226, 1993.

[16] QuickLogic Corp., *QuickLogic 1996/1997*, 1996.

[17] 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.

[18] D. D. Sleator and R. E. Tarjan, "A data structure for dynamic trees," *J. Comput. Syst. Sci.*, vol. 26, pp. 362–391, 1983.

[19] S. Thakur, Y.-W. Chang, D.F. Wong, and S. Muthukrishnan, "Algorithms for an FPGA switch module routing problem with application to global routing," *IEEE Trans. Computer-Aided Design*, pp. 32–47, January 1997.

[20] S. Trimberger and M. Chene, "Placement-based partitioning for lookup-table-based FPGA's," in *Proc. IEEE Intl. Conf. Computer Design*, pp. 91–94, 1992.

[21] G.-M. Wu and Y.-W. Chang, "Switch-matrix architecture and routing for FPDs," in *ACM Intl. Symp. Physical Design*, pp. 158–163, April 1998.

[22] G.-M. Wu and Y.-W. Chang, "Quasi-universal switch matrices for FPD design," *IEEE Trans. on Computers*, vol. 48, no. 10, pp. 1107–1122, Oct. 1999.

[23] Y.-L. Wu, S. Tsukiyama, and M. Marek-Sadowska, "Graph based analysis of 2-D FPGA routing," *IEEE Trans. Computer-Aided Design*, vol. 15, no. 1, pp. 33–44, Jan. 1996.

[24] Xilinx Inc., *The Programmable Logic Data Book,* 1996.

[25] K. Zhu, D. F. Wong and Y.-W. Chang, "Switch module design with application to two-dimensional segmentation design," in *Proc. IEEE/ACM Intl. Conf. Computer-Aided Design*, pp. 481–486, 1993.