FPGA Global Routing Based on a New Congestion Metric *

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

[1] *Department of Computer Sciences, University of Texas at Austin, Austin, Texas 78712*
[2] *Department of Computer Science, Chinese University of Hong Kong, Hong Kong*

Abstract

Unlike traditional ASIC routing, the feasibility of routing in FPGA's is constrained not only by the available space within a routing region, but also by the routing capacity of a switch block. Recent work [6] has established the switch-block capacity as a superior congestion-control metric for FPGA global routing. However, the work has two deficiencies: (1) its algorithm for computing the switch-block capacity is not efficient, and (2) it, as well as the other recent works [1, 4, 14], only modeled one type of routing segments–single-length lines. To remedy the deficiencies, we present in this paper efficient algorithms for obtaining the switch-block capacity and a graph modeling for routing on the new-generation FPGA's with a versatile set of segment lengths. Experiments show that our algorithms dramatically reduce the run times for obtaining the switch-block capacities. Experiments with a global router based on the switch-block and channel densities for congestion control show a significant improvement in the area performance, compared with one based on the traditional congestion metric.

1 Introduction

Symmetrical-array architectures are used in many popular commercial *field-programmable gate arrays* (*FPGA's*) [2, 24]. A typical symmetrical-array FPGA consists of a two-dimensional array of logic blocks interconnected by vertical and horizontal routing channels. Figure 1 shows the symmetrical-array FPGA model. The logic blocks, denoted by L in Figure 1, can be customized to realize a variety of logic functions. The routing channels comprise general routing resources used to connect logic-block pins. To improve circuit performance and maintain reasonable routability simultaneously, the routing channels are usually segmented, and thus routing tracks consist of wires with a versatile set of lengths [2, 24]. Longer segments are intended for high fan-out, time-critical signal nets. On the contrary, shorter segments are intended for short connections to avoid wasting routing resuorces. An intersection area of a horizontal and a vertical routing channels is referred to as a *switch block*, denoted by S in Figure 1. Each switch block has terminals on its four sides. Terminals on different sides of a switch block can be programmed to be electrically connected by internal switches. Thus nets can be routed through the switch block using some of these internal switches.

Routing is a key step in implementing circuits using FPGA's. Researchers have shown in [3, 19, 21] that the feasibility of FPGA design is most constrained by routing resources, and routing delays dominate the performance of FPGA's. The routing problem for the symmetrical-array

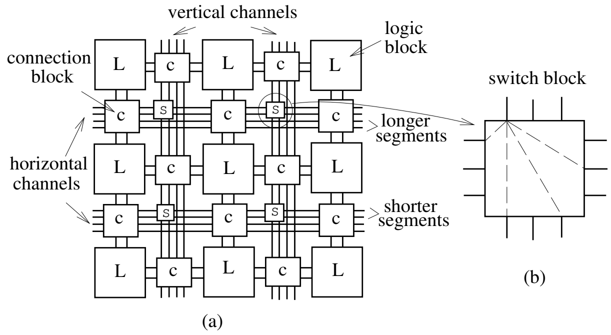

Figure 1: (a) The symmetrical-array FPGA model. (b) A switch block.

FPGA's was studied by [1, 4, 6, 14, 16]. As in traditional ASIC technologies, the FPGA routing problem is often solved in two stages, global routing followed by detailed routing, to make the problem manageable. The goals of these two stages are balancing the channel densities of all routing channels, and assigning nets to specific segments and internal switches, respectively. Most FPGA global routers were based on congestion metrics for traditional ASIC's; for example, the global router in [4] was based on a graph search technique guided by *channel density*. Chang *et al.* in [6] presented a new congestion metric based on *switch-block densities*[1] for FPGA global routing. Their method showed a significant reduction in the maximum channel density required for global routing, compared with those based on the traditional channel density for congestion control.

The new congestion metric is motivated by the intrinsic difference between the FPGA routing resources and those of the traditional ASIC routing. In traditional ASIC routing, wires can be routed in any available space within a routing region, and thus channel density itself can capture feasibility information well. In contrast, the feasibility of routing in FPGA's is constrained not only by a set of fixed routing channels, but also by the physical architecture of a switch block. For example, in Figure 2, although no channel density violates the capacity constraint of two, which is considered feasible in traditional ASIC global routing, there does not exist any FPGA routing solution by using the switch block shown in the figure. The reason is that the internal connections of the switch blocks S_1 and S_3 can not afford any top-to-bottom connection. Thus, to precisely capture the FPGA routing nature, the constraints induced by the physical structure of switch blocks should be considered.

Most existing FPGA routing algorithms such as those in

*This work was partially supported by the Texas Advanced Research Program under Grant No. 003658459.

†On leave from IBM Corporation, T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598.

[1] Number of nets routed through a set of specific directions of a switch block. Section 2 will give a formal definition.

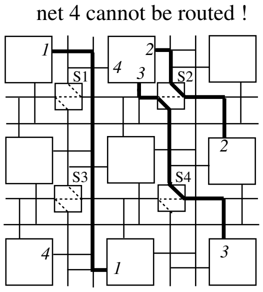

Figure 2: An infeasible routing instance. (Net 4 is not routable though the channel density does not violate the capacity constraint of two.)

[1, 4, 6, 14] only consider one type of routing segments, which is insufficient for the current commercial FPGA's with multi-length of routing segments. In this paper, we consider the global-routing problem on *new-generation* FPGA's with a versatile set of segment lengths. To obtain the capacity constraint of a switch block for congestion control, researchers in [6] presented an algorithm with $O(W^6)$ iterations, where W is the number of tracks in a routing channel, and in each iteration, an *integer linear programming* (*ILP*) was used to solve a switch-block routing problem. The ILP is NP-complete [9] and thus is computationally hard in the worst case.

In this paper, we model a new-generation FPGA as a "three-dimensional" weighted graph. The weights on the edges are proportional to the congestion on the corresponding resources. To obtain efficient solutions for congestion control, we propose faster algorithms. Instead of solving $O(W^6)$ ILP's, our new algorithm needs to check only $O(W^5)$ ILP's for *general* switch blocks. The asymptotical bound of our algorithm is tight. In particular, for an Xilinx XC4000-type switch block [24], we can compute its switch-block capacity in $O(W^5)$ time overall, without the necessity of resorting to any ILP. Experiments show that our algorithm dramatically reduces the run times for computing the capacity constraint of a switch block. We also have tested a global router with the new congestion metric (based on the switch-block and channel densities) on the new-generation FPGA's. Experimental results show that the global router with the new congestion metric has significantly better area performance than those with the traditional metric (based on the channel densities alone).

The rest of this paper is organized as follows. Section 2 gives the problem specifications. Section 3 proposes our new algorithms for acquiring the congestion-control mechanism. Section 4 presents a graph modeling and a modified global-routing algorithm for the new-generation FPGA's based on that in [6]. Experimental results are reported in Section 5.

2 Problem Formulation

The routing tracks in a new-generation FPGA [10, 24] consist of three types of interconnect, distinguished by the relative length of their segments: *single-length lines*, *double-length lines*, and *longlines*. See Figure 3 for an illustration. The single-length lines form a grid of horizontal and vertical lines that intersect at switch blocks. The double-length lines consist of a grid of segments twice as long as the single-length lines. The longlines are a grid of segments that run the entire vertical or horizontal channel.

A switch block is a $W \times W$ square block, where W is the number of terminals on each side of the switch block. Some pairs of terminals, on different sides of the block, may have programmable switches and thus can be connected by pro-

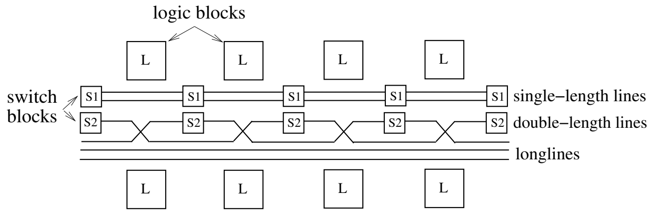

Figure 3: Three types of segments in a routing channel.

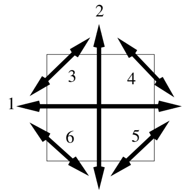

Figure 4: Six types of connections.

gramming the switches to be "ON." Moreover, these switches are electrically *non-interacting*, unless they share a terminal. The *switch-block flexibility*, denoted by F_S, is defined as the number of programming switches between a terminal and others. For example, the switch block in Figure 1(b) has $F_S = 4$. Switch blocks are used in symmetrical-array FPGA's for connections between single-length lines or between double-length lines [24].

A net can be routed through a switch block by programming some switch. To characterize such local routes, we say a *connection* is established in the switch block between two terminals, on different sides of the switch block, if the switch between those two terminals is programmed to be "ON." There are six types of connections. Each type is characterized by two sides of a switch block. Figure 4 shows the classification. The connection labeled i, $1 \leq i \leq 6$, in Figure 4, is said to be of *Type-i*. Type-1 and -2 connections are *straight connections* while the others are *bent connections.*

A *routing requirement vector* (*RRV*) $\vec{n}$ is a six-tuple $(n_1, n_2, n_3, n_4, n_5, n_6)$ where $0 \leq n_i \leq W$, $\forall 1 \leq i \leq 6$. A *routing* for an RRV on a given switch block is a set of connections such that there are n_i of Type-i connections, for $i \in \{1, \ldots, 6\}$, and those connections do not share any terminal. An RRV $\vec{n}$ is said to be *routable* on a switch block S if there exists a routing for $\vec{n}$ on S. For example, in Figure 2, the routing with the RRV $(0,0,0,1,0,1)$ on the switch block S_2 is shown. The RRV $(0,0,1,1,0,1)$ is not routable on any switch block shown in Figure 2.

We define the *switch-block density* of a switch block S, denoted by $\vec{d_S}$, as a vector $(m_1, m_2, \ldots, m_6)$, where m_i, $1 \leq i \leq 6$, is the number of Type-i connections currently routed through S. An RRV $\vec{n}$ is said to be *maximally routable* on S if $\vec{n}$ is routable on S and no additional nets can be routed through S. We define the *capacity* of a switch block S, denoted by C_S, as the set of maximally routable $\vec{n}$'s on S.

An RRV $\vec{m} = (m_1, \ldots, m_6)$ is *dominated* by another RRV $\vec{n} = (n_1, \ldots, n_6)$, denoted by $\vec{m} \leq \vec{n}$, if and only if $n_i \geq m_i, i = 1, \ldots, 6$. Any RRV $\vec{m}$ is routable if another RRV $\vec{n}$ is routable on a switch block S and $\vec{m} \leq \vec{n}$. We define a *dominating set* D for S as follows: for an RRV $\vec{m}$, $\vec{m}$ is routable on S if and only if there exists an RRV $\vec{n} \in D$ such that $\vec{m} \leq \vec{n}$. A dominating set D for S is called *minimal* if $\forall \vec{m}, \vec{n} \in D$ neither $\vec{m} \leq \vec{n}$ nor $\vec{n} \leq \vec{m}$. The *minimal dominating set* of a switch block S, denoted by D_S, well

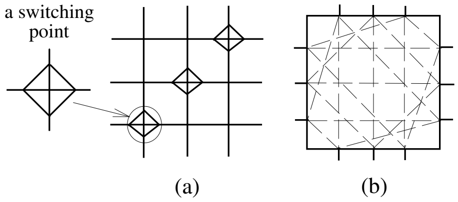

Figure 5: (a) A Xilinx XC4000-type switch block ($F_S = 3$) and (b) its corresponding switch-block model.

characterizes the capacity constraint of S. In fact, $C_S = D_S$ and thus we will use them interchangeably in this paper.

In this paper, we first study the following problems.

- *The Switch-Block Routing Problem* (*SBRP*): Given a switch block S and an RRV $\vec{n}$, is $\vec{n}$ routable on S?
- *The Minimal Dominating Set Problem* (*MDSP*): Given a switch block S, find the D_S (or C_S) for S.

The solution to the SBRP is used as a procedure to solve the MDSP, and the result of the MDSP paves the way for acquiring a new congestion-control mechanism. This mechanism is the key ingredient for solving the *FPGA global-routing problem* defined below:

- *The FPGA Global Routing Problem:* Given an FPGA architecture and a set of nets, find an assignment of a sequence of segments and switch blocks for each net so that all terminals of the net are connected.

3 Switch-Block Capacity

As mentioned earlier, the global router is guided by a new congestion metric of switch blocks and routing channels. It routes one net at a time. After each net is routed, the costs of using switch blocks and routing channels are updated. The cost of a routing on a switch block S is a function of the switch-block capacity C_S (D_S) and the current switch-block density $\vec{d_S}$. Thus we shall first study the techniques for obtaining the capacity constraint of a switch block.

The first subsection presents a constant-time algorithm for solving the SBRP's on a Xilinx XC4000-type switch block. The second subsection describes a faster algorithm which uses the solutions to the SBRP as a subroutine to compute C_S's (D_S's) for switch blocks.

3.1 Switch-Block Routing Problem

An exact algorithm based on integer linear programming (ILP) for solving the SBRP was proposed in [6, 18]. However, ILP is NP-complete [9] and thus is not computationally efficient. Whether the general SBRP can be solved in polynomial time is still open. In this subsection, we first explore the architecture of XC4000-type switch blocks and present a constant-time algorithm for solving the special SBRP's. Figure 5 shows the switch-block architecture with $W = 3$. We have the following theorem.

Theorem 1 *For an XC4000-type switch block S of size W, $\vec{n}$ is routable on S if and only if* $\max\{n_1, n_2\} + \max\{n_3, n_5\} + \max\{n_4, n_6\} \leq W$.

Corollary 1.1 *The SBRP can be solved in $O(1)$ time on an XC4000-type switch block.*

By Theorem 1, the SBRP's on an XC4000-type switch block can be solved in constant time by checking whether the components of a given RRV satisfy the inequality shown in Theorem 1. This avoids the necessity of using the ILP's which have exponential-time complexity in the worst case. For general switch blocks, we still resort to the ILP-based algorithm in [6]. (See [6] for its formulation.)

3.2 Minimal Dominating Set

An algorithm for computing the minimal dominating set for a given switch block was proposed in [6, 18]. It needs to consider $O(W^6)$ RRV's; for each RRV, an integer linear programming is used to determine if the RRV is routable on the given switch block, i.e., to solve the SBRP. We present in the following a new algorithm which only $O(W^5)$ RRV's need to be considered to obtain a minimal dominating set. Note that there exist instances which contain kW^5 RRV's in their corresponding minimal dominating sets, where k is a positive constant; thus the asymptotical bound of our algorithm is tight. In particular, by Corollary 1.1, we only need $O(W^5)$ time overall to compute a minimal dominating set for an XC4000-type switch block.

In this subsection, fix a switch block S. Let $R_S = \{\vec{n} | \vec{n}$ is routable on $S\}$. We can interpret each RRV $\vec{n}$ as a nonnegative lattice point in an orthogonal 6-D space with axes X_i, $1 \leq i \leq 6$. For convenience, we will use $\vec{n}$ to represent both a generic routing requirement *vector* and the corresponding 6-D lattice *point*. Since each component, n_i, $1 \leq i \leq 6$, of an RRV $\vec{n}$ satisfies the constraint $0 \leq n_i \leq W$, where W is the size of S. R_S can be bounded by a set of linear inequalities, and these inequalities form a 6-D bounding polyhedron. Let $n_i^* = \max\{n_i | \vec{n} \in R_S\}$, $1 \leq i \leq 6$, and define $\vec{n_i}^*$ as follows: $\pi_i(\vec{n_i}^*) = n_i^*$ and $\pi_j(\vec{n_i}^*) = 0$, where $\pi_i(\vec{n}) = n_i$, $1 \leq i, j \leq 6, i \neq j$. In other words, all components of $\vec{n_i}^*$ are zero, except that the i-th component is n_i^*. We refer to the bounding polyhedron, formed by the six axes, $\vec{n_i}^*$'s, $1 \leq i \leq 6$, and the RRV's in D_S, as P_S. We have the following lemma.

Lemma 1 *(1) P_S is convex; (2) $\vec{n_i}^* \in R_S, 1 \leq i \leq 6$.*

We describe the new algorithm for computing the minimal dominating set D_S for a given switch block S. Instead of checking all RRV's bounded by P_S to compute D_S, our algorithm applies a technique called *boundary scan* to avoid unnecessary computations: it only needs to consider the RRV's around the boundaries of P_S. Evidently each RRV inside P_S must be dominated by some RRV around the boundaries. Thus it suffices to check those boundary RRV's to obtain D_S.

Our algorithm consists of two stages: (1) the *locating stage* and (2) the *boundary-scan stage*. In the locating satge, we intend to find a starting position on a boundary of P_S for the boundary-scan stage. In the boundary-scan stage, we examine the RRV's around the boundaries of P_S to obtain D_S. We describe the algorithm as follows and summarize it in Figure 7.

Algorithm: Minimal_Dominating_Set(S)

1. The locating stage:
   We can apply a maximum-cardinality matching algorithm on a bipartite graph to compute the maximum number of non-interacting switches for an arbitrary type of connections of S, say Type-1 connections, as follows. To construct the bipartite graph, say $G = (U, V, E)$, we introduce a node in U (V) for each terminal on the left (right) side of S. We then construct an edge $e(u, v) \in E$, $u \in U$ and $v \in V$, if there is a switch between the corresponding terminals represented by u and v. Let the cardinality of the matching be n_1^*. By Lemma 1(2), $\vec{n_1}^* = (n_1^*, 0, 0, 0, 0, 0)$ is routable on S, but $\vec{n_1}^{*\prime} = (n_1 + 1, 0, 0, 0, 0, 0)$ is not. Note that $\vec{n_1}^*$ is a boundary lattice point of P_S on the axis X_1.
2. The boundary-scan stage:
   Then we shall start from $(n_1^*, 0, 0, 0, 0, 0)$ to examine the RRV's around the boundaries of P_S to acquire the minimal dominating set D_S. This requires to solve one SBRP on S for each RRV during the scan. To simplify

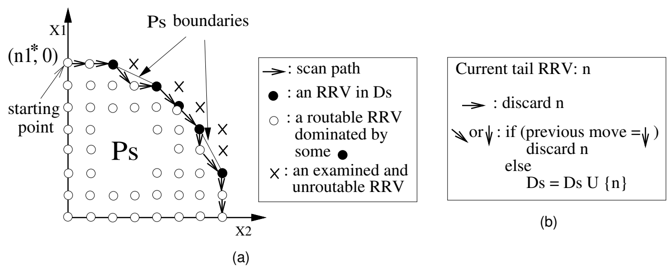

Figure 6: The boundary scan in a 2-D analogy with axes X_1 and X_2. (a) The boundary RRV's and the checking sequence. (b) Three types of moves on the scan path.

the explanation, we first describe the basic scenario of the boundary-scan technique for the cases in a 2-D space with axes X_1 and X_2 and then generalize it to the cases in the 6-D space. Figure 6 illustrates the technique in the 2-D space.

Suppose the maximum number of non-interacting switches is n_1^* for the Type-1 connections. The RRV $(n_1^*, 0)$ is routable on S and is served as the starting point of the boundary scan. There are three types of moves, Type-$\rightarrow$, Type-$\searrow$, and Type-$\downarrow$ moves, characterized by the relative positions of two consecutive routable RRV's on the scan path. The earlier-scanned RRV is called a *tail* RRV, and the other a *head* RRV. See Figure 6(a) for an illustration. The algorithm examines RRV's in the order (1) Type-$\rightarrow$, (2) Type-$\searrow$, and (3) Type-$\downarrow$. For each tail RRV, there is no need to examine the following type(s) of move(s) if any preceding type of move is found. The first case Type-$\rightarrow$ denotes the move from the tail RRV (n_1, n_2) to the head RRV (n_1, n_2+1), and both (n_1, n_2) and (n_1, n_2+1) are routable. Since $(n_1, n_2) \le (n_1, n_2+1)$, $(n_1, n_2) \notin D_S$. Thus we shall discard the tail RRV (n_1, n_2) and then continue the process by using the current head RRV (n_1, n_2+1) as the new tail RRV. In the second (third) case, Type-$\searrow$ (Type-$\downarrow$) represents the move from (n_1, n_2) to (n_1-1, n_2+1) ((n_1-1, n_2)), and both (n_1, n_2) and (n_1-1, n_2+1) ((n_1-1, n_2)) are routable. There are two subcases, distinguished by whether the previous move is Type-$\downarrow$:

(1) If the previous move is Type-$\downarrow$, (n_1, n_2) must be dominated by the previously scanned RRV, i.e., $(n_1, n_2) \notin D_S$. Hence we shall discard (n_1, n_2) and then continue the process by using the current head RRV as the new tail RRV.

(2) Otherwise, by the convexity of P_S (Lemma 1(1)), (n_1, n_2) is not dominated by any other routable RRV and thus must be contained in D_S; i.e., $D_S = D_S \cup \{(n_1, n_2)\}$. The process then continues by updating the current head RRV as the new tail RRV.

The boundary-scan technique applies to the cases in the 6-D space with axes X_i, $1 \le i \le 6$. However, for the 6-D cases, there are some submoves in each of Type-$\rightarrow$ and Type-$\searrow$ moves that need to be considered. A submove is called a $\rightarrow$*k-move* if it is a Type-$\rightarrow$ move and $\sum_{2 \le i \le 6} (n_i' - n_i) = k$, where $\vec{n} = (n_1, n_2, \ldots, n_6)$ and $\vec{n}' = (n_1', n_2', \ldots, n_6')$ are a tail and a head RRV's, respectively. Here $n_1 = n_1'$ and $n_i' - n_i \le 1$, $\forall 2 \le i \le 6$. For example, the move from the tail RRV $(6,0,0,0,0,0)$ to the head RRV $(6,1,0,1,0,1)$ is a $\rightarrow$3-move. Hence there are $C(5,k)$ $\rightarrow$$k$-moves, $1 \le k \le 5$, and $\sum_{1 \le k \le 5} C(5,k) = 31$ Type-$\rightarrow$ moves at most. $\searrow$*k-moves* are similarly defined except that, for Type-$\searrow$ moves, $n_1' = n_1 - 1$.

For each type of Type-$\rightarrow$ and Type-$\searrow$ moves, we need to consider 31 submoves at most. Since some of those 31 RRV's may not be routable on S or may be dominated by some other RRV's, we shall identify a dominating set for the 31 RRV's, denoted by $D_{\vec{n}}$, where $\vec{n}$ is the current tail RRV. Each RRV in $D_{\vec{n}}$ will be a head RRV for the current scan. To prevent unnecessary checking in finding $D_{\vec{n}}$, the algorithm proceeds in level $1 \cdots 5$. At level k, the set of the potential head RRV's of $\rightarrow$$k$-moves ($\searrow$$k$-moves) is considered. In particular, only the RRV $\vec{m}$, which all of those RRV's processed in the previous level dominated by $\vec{m}$ are routable, is considered. Note that all of the RRV's in $D_{\vec{n}}$ are current head RRV's and thus shall be stored for further processing after the current tail RRV is examined. Figure 7 shows the algorithm.

```
Algorithm: Minimal_Dominating_Set(S)
Input:  S -- switch-block specifications.
Output:  D_S -- the minimal dominating set of S.

The locating stage:
    Compute the maximum number of non-interacting switches
        for Type-1 connections.  Let the number be n_1^*.
The boundary-scan stage:
    D_S = ∅;
    Enqueue(Q, (n_1^*,0,0,0,0,0));
    while (Q ≠ ∅) do {
        n⃗ = Dequeue(Q);
        D_n⃗ = dominating set of the current potential
            Type-→ head RRV's wrt the tail RRV n⃗;
        if (D_n⃗ ≠ ∅) { /* case 1:  Type-→ */
            for each RRV m⃗ ∈ D_n⃗
                Enqueue(Q, m⃗);
        } else {
            D_n⃗ = dominating set of the current potential
                Type-↘ head RRV's wrt the tail RRV n⃗;
            if (D_n⃗ ≠ ∅) { /* case 2 */
                if (the previous move ≠ Type-↓)
                    D_S = D_S ∪ {n⃗};
                for each RRV m⃗ ∈ D_n⃗
                    Enqueue(Q, m⃗);
            } else { /* case 3:  Type-↓ */
                if (the previous move ≠ Type-↓)
                    D_S = D_S ∪ {n⃗};
                if (n_1 > 0)
                    Enqueue(Q, (n_1 - 1, n_2, n_3, n_4, n_5, n_6));
            } /* else */
        } /* else */
    } /* while */
    Output D_S;
```

Figure 7: The Minimal_Dominating_Set algorithm.

Theorem 2 *Algorithm Minimal_Dominating_Set correctly computes the minimal dominating set of a switch block by checking only $O(W^5)$ RRV's.*

Corollary 2.1
Algorithm Minimal_Dominating_Set correctly computes the minimal dominating set of an XC4000-type switch block in $O(W^5)$ time.

4 Graph Modeling and Global Routing

In this section, we use the precomputed minimal dominating set to define the new congestion metric in switch blocks and routing channels; that is, we compute the minimal dominating set off-line. We define $C_{\vec{d_S}} = \{ \vec{n} \in D_S | \vec{d_S} \leq \vec{n} \}$, where $\vec{d_S}$ is the density of a switch block S. Since the feasibility condition with respect to S can be characterized by its switch-block capacity, we can model congestion as a function of $\vec{d_S}$ and $C_{\vec{d_S}}$. We use a global router based on a graph search technique to demonstrate the new congestion metric. The router assigns higher costs to route nets through congested areas to balance the net distribution among routing channels.

4.1 Graph Modeling

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. Our solution is to model the FPGA as a "3-D 6-layer" graph. There is one layer for single-length lines, the *single-length layer*; four layers for double-length lines, *double-length layers*; one layer for longlines, the *longline layer*. Figure 8 illustrates the FPGA modeling. As shown in Figure 8, each logic block or connection block is represented by a node, called a *logic node* and a *connection node*, respectively. Each routing channel is modeled as an edge named a *channel edge*; each set of connections between a logic and a connection blocks is also 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 (four nodes) are referred to as *switch edges* (*nodes*).[2] There are *layer edges* between the corresponding switch nodes in different layers. Layer edges are used for switching a route between different types of routing segments.

Note that only the single-length layer contains logic nodes which house logic pins; thus a path must start and end at logic nodes in the single-length layer. Paths in the graph represent routes on the FPGA, and vice versa. Weights associated with edges represent congestion information. We shall denote the graph used to model an FPGA F as $G_F(V, E)$. The edge set E is partitioned into the set of channel edges $\mathcal{C}_e$, the set of switch edges $\mathcal{S}_e$, the set of connection edges $\mathcal{N}_e$, and the set of layer edges $\mathcal{L}_e$.

4.2 Global-Routing Algorithm

To *demonstrate* the new congestion metric, we modify the router used in [6] to deal with the routing on the new-generation FPGA's. It is based on the Dijkstra's shortest path algorithm [8]. In this subsection, we describe the modified routing algorithm. Unlike the classical ASIC global router which is guided by the channel density, the FPGA global router is guided by both the switch-block and channel densities simultaneously. The main goal is to evenly distribute the nets among routing channels so that the channel density after all nets are routed is minimized, and the capacity constraints of switch blocks are satisfied. The router does the routing net by net. For the net being routed currently, we prefer to route it along uncongested routing regions. We apply the *cost function* $\alpha : E \rightarrow \Re^{+} \cup \{0\}$ to guide the routing of a net:

$$\alpha(e) = \begin{cases} \max_{\vec{n} \in C_{\vec{d_S}}} (\sum_{i=1}^{6} 1/c^{(n_i - m_i)}), & e \in \mathcal{S}_e \\ 1/c^{W-d_i}, & e \in \mathcal{C}_e \\ 0, & e \in \mathcal{L}_e \cup \mathcal{N}_e \end{cases}$$

[2] A switching component in the longline layer is modeled as a degenerate node used for switching a route to and from other layers.

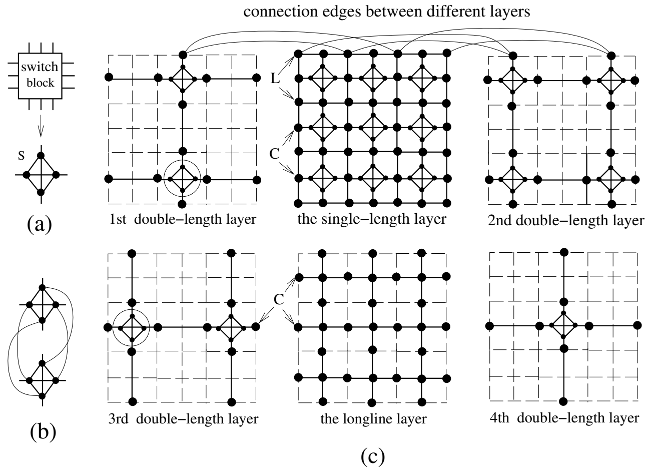

Figure 8: The graph modeling for an FPGA with the logic-block size 4×4. (a) The switch-block modeling. (b) Layer edges between two corresponding switching components in different layers. (c) Six layers for three types of routing segments. (Only the connection edges on the top rows between the single-length and the first [second] double-length layers are shown. Note that the dashed lines are used for reference, not part of the graph.)

where c is a positive constant, $e \in E$, $\vec{d_S} = (m_1, \ldots, m_6)$, and $\vec{d} = (d_1, \ldots, d_K)$ is the current *channel-density vector*. Here d_i is the density in channel i of the FPGA, and K is the number of channels in the FPGA. Hence, $W - d_i$ denotes the *slack* with respect to channel i, and $n_i - m_i$ is the slack with respect to the i-th components of $\vec{n}$ and $\vec{d_S}$, i.e., the number of currently available non-interacting switches for Type-i connections.

The whole routing procedure is illustrated in Figure 9. Given an FPGA F, we first construct a graph G_F to model F. Initially, $\vec{d_S} \leftarrow (0,0,0,0,0,0)$ for every switch block S, $d_i \leftarrow 0$, $1 \leq i \leq K$, for every channel i, and $C_{\vec{d_S}} \leftarrow C_S$ (D_S) (the minimal dominating set of S). The weight of every edge in F is computed from the cost function α above. See Figure 9(a) for the initial configuration. After a net is routed, all $\vec{d_s}$, $C_{\vec{d_S}}$, and $\vec{d}$ need to be updated to reflect the additional congestion resulting from the routing of the net. Therefore weights associated with the switch and channel edges on the route are recomputed using the updated $\vec{d_S}$, $C_{\vec{d_S}}$, $\vec{d}$, and the cost function α above. See Figure 9(b) for the update. In particular, for the double-length and longline layers, weights associated with all corresponding channel edges shall be updated simultaneously if a portion of a double-length line or a longline is used. Note that the weights of all connection and layer edges remain zero throughout the routing; that is, the router is guided by both the switch-block and channel densities. In Figure 9(b), those RRV's which no longer dominate $\vec{d_S}$ are removed from $C_{\vec{d_S}}$ during the update. The process continues as routing proceeds. See Figure 9(c). Notice that the cardinality of $C_{\vec{d_S}}$ monotonically decreases during the process. We let $C_{\vec{d_S}}$ resume the initial configuration, i.e., $C_{\vec{d_S}} \leftarrow C_S$, when its cardinality equals zero. See Figure 9(d).

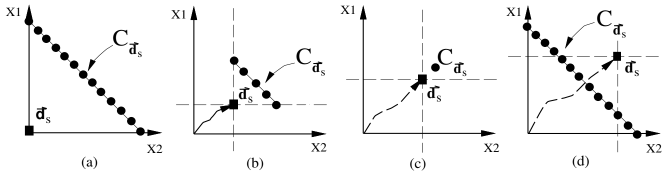

Figure 9: Dynamically update congestion information during routing, illustrated in a 2-D plane with axes X_1 and X_2. (a) The initial stage. (b) Update $\vec{d_S}$ and $C_{\vec{d_S}}$. (c) Only one vector remains in $C_{\vec{d_S}}$. (d) Resume the initial $C_{\vec{d_S}}$ when none in $C_{\vec{d_S}}$ dominates $\vec{d_S}$.

5 Experimental Results

We implemented Algorithm Minimal_Dominating_Set and the global router in C on a SUN SPARC 5 station. We first computed the minimal dominating sets for XC4000-type switch blocks off-line. In addition to the channel-capacity constraints, the FPGA global router used the minimal dominating sets as the capacity constraints of switch blocks to guide the routing; i.e., it used the new congestion metric of switch-block and channel capacities.

For the purpose of comparison, we also implemented a global router based on the same shortest-path algorithm [8]. Unlike the new router mentioned above, this counterpart used only the traditional measure of channel densities as a congestion-control parameter. Contrary to the method used by the new router, it assigned zero weights to all switch, connection, and layer edges, while the weights of channel edges were dynamically updated by the cost function $\beta : E \rightarrow \Re^+ \cup \{0\}$ during the routing:

$$\beta_i(e) = \begin{cases} 0, & e \in \mathcal{S}_e \\ 1/c^{W-d_i}, & e \in \mathcal{C}_e \\ 0, & e \in \mathcal{L}_e \cup \mathcal{N}_e \end{cases}$$

where c and $\vec{d}$ are the same as those in the cost function α. Note that the cost function β is used in traditional ASIC routing [13].

We tested both routers on the CGE benchmark circuits [4]. Table 1 gives the names of the circuits, the numbers of logic blocks in the FPGA's, and the numbers of equivalent two-pin nets in the circuits. For FPGA's, the capacity of a channel is the size of a switch block. In the experiment, we used the parameter $c = 2$ for both cost functions. The switch blocks used were those in Xilinx XC4000 series FPGA's [24]; the switch blocks have $F_S = 3$. (See Figure 5 for the switch-block architecture.) The numbers of tracks for single-length lines, double-length lines, and longlines were $\lceil W/2 \rceil$, $\lceil W/4 \rceil$, and $W - \lceil W/2 \rceil - \lceil W/4 \rceil$, respectively. To explore the effects of the metrics, all benchmark circuits were routed using the same net ordering, and no rip-up and re-route phase was incorporated to further improve the performance. The quality of an FPGA routing solution was evaluated by the channel density after all nets are routed so that no routing regions violate the capacity constraints. At the end of global routing, a switch block S is *feasible* if there exists an RRV $\vec{n} \in C_{\vec{d_S}}$ such that $\vec{d_S} \leq \vec{n}$. Similarly, a channel i is feasible if its density does not exceed the number of tracks in it. Table 1 lists the channel densities after all nets are routed such that every switch block and routing channel is feasible. It shows that the router with the new congestion metric (switch-block and channel densities) has much better area performance than that with the traditional one (channel densities alone). The router with the traditional congestion metric needs an average of 28% larger channel densities after global routing than the router with the new metric.

Notice that, at first glance, this may seem to be an unrealistic comparison that we did not compare our routing results with those in the recent work [1, 4, 6, 14]. The reasons are as follows.

- As mentioned earlier, the works in [1, 4, 6, 14] only modeled FPGA's with just one type of routing segments—single-length lines, while ours considers the new-generation FPGA's with a versatile set of segment lengths. Hence, it is meaningless to make comparison with those results.
- More crucially, what we have presented is a congestion metric which is intended for *general* FPGA global-routing formulations such as Steiner-tree-, shortest-path-based algorithms, etc. Thus the most important concern shall be the investigation of the effects of the two metrics. This is why 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 re-route was incorporated.

Circuit	logic blocks	2-pin nets	Channel densities	
			new metric	traditional
BUSC	12×13	392	10	11
DMA	16×18	771	10	16
BNRE	21×22	1257	13	16
DFSM	22×23	1422	12	16
Z03	26×27	2135	15	18
Total	-	-	60	77

Table 1: Global-routing results: channel densities for routing all nets and keeping all switch blocks and channels feasible (new metric: switch-block and channel densities; traditional metric: channel densities alone).

We also compare the run times for computing the minimal dominating sets for XC4000-type switch blocks. We applied the ILP-based algorithm in [6] using the ILP solver, *lp_solve*. Table 2 gives the run times required to obtain the minimal dominating sets for XC4000-type switch blocks by using (1) the algorithm in [6], (2) our Minimal_Dominating_Set algorithm based on ILP's (Theorem 2 + ILP's), and (3) our overall $O(W^5)$-time algorithm on an XC4000-type switch block (Corollary 2.1). It shows that our algorithms run much more efficiently than that used in [6].

W	Run times for computing a minimal dominating set		
	algorithm [6]	Thm 2 + ILP's	Corollary 2.1 ($O(W^5)$)
5	28 min	6 min	1 sec
10	862 min	84 min	24 sec
15	about 6 days	576 min	228 sec

Table 2: Run times for computing a minimal dominating set for an XC4000-type switch block.

6 Concluding Remarks

We have proposed an efficient algorithm which needs to check only $O(W^5)$ RRV's for computing the capacity constraint of a switch block. The asymptotical bound of our algorithm is tight. In particular, for an XC4000-type switch block, our algorithm can dramatically reduce the *overall* complexity to $O(W^5)$ time (from solving $O(W^6)$ ILP's). Experiments have shown the efficiency of our algorithms. We also have presented a graph modeling for FPGA global routing using a new congestion metric associated with both switch blocks and routing channels. Experiments have shown that the new combined congestion metric is superior to that associated with channel densities alone. The new congestion metric is intended for general FPGA routing *algorithms* as well as routing *architectures*; it also applies to other formulations such as Steiner trees and other architectures like that in [12]. Note that our experiments were intended for the comparison between the new and traditional congestion metrics. To achieve the best area performance, we suggest that a rip-up and re-route phase be incorporated, and/or some successful formulation such as Steiner trees be used.

The effect of an FPGA architecture on routing has drawn increasing attention recently [5, 7, 17, 20, 22, 23]. Whether the two-stage routing strategy, global routing followed by detailed routing, is suitable for FPGA's has been discussed in [1, 20, 22, 23]. Researchers in [22, 23] have shown that the final detailed routing, in the worst case, may require a much larger number of tracks than the channel density found by a global router; the worst cases occur when most nets are very long and are routed in some specially designed topologies. This worst-case analysis seems to suggest that the two-stage strategy is not fit for FPGA routing. Though theoretically sound and interesting, the worst-case scenario is often pathologically pessimistic and rarely happens in practical applications. (See Page 106 in [11], the Turing Award lecture by R. M. Karp.) In fact, researchers in [5, 7, 17] have shown by extensive experiments and stochastic analyses based on the two-stage approach that $F_S = 3$ combined with high pin-to-track connections is sufficient for detailed-routing completion; the following crucial factors contribute to this phenomenon in practical FPGA routing:

- For practical applications, most nets are short. For example, about 60% (90%) of nets in the CGE [4] and SEGA [15] benchmark circuits route through no more than two (five) switch blocks, independent of the sizes of FPGA's.
- Most logic-block pins are logically equivalent [19]; pin permutations can be used to optimize routing.

Futhermore, currently existing one-stage routers (with detailed routing alone) often have high time and space complexity. With the trend that the FPGA size will become substantially larger than it is now, it will make the routing problem hard to manage in one stage. Therefore, the two-stage approach, global routing followed by detailed routing, will become necessary for FPGA routing in the future.

7 Acknowledgments

We would like to thank Prof. Stephen Brown and the authors of [4] for providing us with the benchmark circuits.

References

[1] M. Alexander and G. Robins, "New performance-driven FPGA routing algorithms," *Proc. ACM/IEEE Design Automation Conference*, pp. 562–567, 1995.

[2] AT&T Microelectronics, "Optimized reconfigurable cell array (ORCA) series field-programmable gate arrays," Advance Data Sheet, February, 1993.

[3] N. Bhat and D. Hill, "Routable technology mapping for LUT FPGAs," *Proc. Intl. Conf. Computer Design, VLSI in Computers and Processors*, pp. 95–98, 1992.

[4] 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. 620–627, 1992.

[5] S. 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, 1993.

[6] Y.-W. Chang, S. Thakur, K. Zhu, and D. F. Wong, "A new global routing algorithm for FPGAs," *Proc. Intl. Conf. Computer-Aided Design*, pp. 380–385, 1994.

[7] Y.-W. Chang, D. F. Wong, and C. K. Wong, "Universal switch modules for FPGA design," *Technical Report TR95-27*, The University of Texas at Austin, 1995.

[8] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, 1:269–271, 1959.

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

[10] H.C. Hsieh, *et al.*, "Third-generation architecture boosts speed and density of field-programmable gate arrays," *Proc. IEEE Custom Integrated Circuits Conf.*, pp. 31.2.1–31.2.7, 1990.

[11] R. M. Karp, "Combinatorics, complexity, and Randomness," *Communications of the ACM*, vol. 29, no. 2, pp. 98–109, 1986.

[12] K. Kawana, H. Keida, M. Sakamoto, K. Shibata, and I. Moriyama, "An efficient logic block interconnect architecture for user-reprogrammable gate array," *Proc. IEEE Custom Integrated Circuits Conf.*, pp. 31.3.1–31.3.4, 1990.

[13] E. S. Kuh and M. Marek-Sadowska, "Global routing," in *Layout Design and Verification* (T. Ohtsuki, *ed.*), Elsevier Science Publishing, 1985.

[14] Y. -S. Lee and C. -H. Wu, "A performance and routability driven router for FPGA's considering path delay," *Proc. ACM/IEEE Design Automation Conference*, pp. 557–561, 1995.

[15] G. Lemieux and S. Brown, "A detailed routing algorithm for allocating wire segments field-programmable gate arrays," *Proc. ACM/SIGDA Physical Design Workshop*, Lake arrowhead, CA, 1993.

[16] M. Palczewski, "Plane parallel A* maze router and it applications," *Proc. Design Automation Conference*, pp. 691–697, 1992.

[17] J. Rose and S. D. Brown, "Flexibility of interconnection structures for field-programmable gate arrays," *IEEE J. Solid State Circuits*, vol.26, no.3, pp. 277–282, 1991.

[18] S. Thakur, D.F. Wong, and S. Muthukrishnan, "Algorithms for FPGA switch module routing," *Proc. European Design Automation Conference*, pp. 265–270, 1994.

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

[20] S. Trimberger, "Effects of FPGA architecture on FPGA routing," *Proc. ACM/IEEE Design Automation Conference*, pp. 574–578, 1995.

[21] S. Trimberger and M. Chene, "Placement-based partitioning for lookup-table based FPGAs," *Proc. Intl. Conf. Computer Design, VLSI in Computers and Processors*, pp. 91–94, 1992.

[22] Y.-L. Wu and D. Chang, "On the NP-completeness of regular 2-D FPGA routing architectures and a novel solution," *Proc. IEEE/ACM Intl. Conf. Computer-Aided Design*, pp. 362–366, 1994.

[23] Y.-L. Wu and M. Marek-Sadowska, "Graph Based Analysis of FPGA routing," *Proc. European Design Automation Conference*, pp. 104–109, 1993.

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