Switch Bound Allocation in Timing-Driven Routing of FPGAs

Kai Zhu and D.F. Wong
Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712-1188

Abstract

In segmented channel routing of row-based FPGAs [6, 7], the routability and interconnection delays depend on the choice of the upper bounds on the number of programmable switches used in routing connections in the channel. In this paper, we present an algorithm for determining such upper bounds for all connections of a net simultaneously so that the predefined source-to-sink delay bound of the net is satisfied and the routability of the net is maximized. Distributed RC delay model is used to compute source-to-sink delay. The algorithm can also be applied to timing-driven routing of symmetrical-array based FPGAs and FPICs. Preliminary experimental results show that our algorithm can significantly improve routability and delay bound satisfiability as compared with the traditional uniform switch bound allocation approach [6, 7].

1 Introduction

Timing-driven physical design researches have been focusing on placement [12, 13]. The goal of timing-driven placement is to satisfy timing constraints on the circuit design and to alleviate subsequent routing as much as possible by, for instance, minimizing wire lengths. The timing constraints on a circuit design are usually specified as the arrival times on the primary inputs and the required arrival times on the primary outputs [10]. One of the effective timing-driven approaches is to convert the timing constraints on a circuit into delay bound constraints on individual nets [4, 9]. The delay bounds on nets are then honored in the subsequent routing phase. In traditional ASIC technologies, interconnection delays incurred in nets can be reasonably approximated as proportional to wire lengths, and it is relatively easier to honor the delay bounds on nets established in placement phase during routing. In Field-Programmable Gate Arrays (FPGAs), however, the routing resources are usually irregular and consist of routing segments of various lengths. The irregularity of routing resources in FPGAs makes it impractical to estimate the interconnection delays simply based on wire lengths. It is therefore much more difficult to control interconnection delays and to honor delay bounds on the nets in FPGAs.

In row-based FPGAs [5, 14], routing is usually performed in two phases: global routing and segmented channel routing. Global routing [15, 17] determines for each net the routing channels and vertical segments with objectives like minimizing wire lengths and channel congestion, but not interconnection delay. After global routing, channel routing is performed with the objective to
maintain a reasonable balance between interconnection delays and routability [6, 7]. The tradeoff between interconnection delays and routability is controlled by setting upper bounds on the number of switches, called switch bounds henceforth, used in routing every connection, i.e. a segment of a net in the channel. Traditionally, the switch bounds on all the connections in a channel are set uniformly to the same number \( K \) and the channel routing is performed as \( K \)-segment channel routing [6, 7]. Moreover, the choice of uniform switch bound \( K \) is made independently in each channel without consideration on the relationship between the connections belonging to the same net. As we know, however, the delay bounds established in placement are specified on nets, instead of on individual connections. Setting uniform switch bounds for connections of a net without consideration of the entire nets may lead to either poor routability or violations of the delay bounds. Therefore, to have high routability and at the same time to satisfy the delay bounds on the nets, it is necessary to consider the switch bounds for all connections of the same net together.

We have developed algorithms for switch bound allocation for row-based FPGAs. The algorithms consider all connections of a net simultaneously in the switch bound allocation. The switch bounds of connections in the same channel are in general non-uniform and are chosen to maximize the routability of the net and at the same time satisfy the delay bounds imposed by the timing-driven placement algorithms. The distributed RC delay model is used to compute routing delays on the nets. Both programmable switches and routing segments are included in the computation of routing delays. The switch bound allocation algorithms can be applied to both row-based and symmetrical-array FPGAs as an intermediate step between global routing and detailed routing in timing-driven routing.

Not much works have been reported on the switch bound allocation problem. For row-based FPGAs, an ad hoc approach is to set the switch bounds for all connections uniformly [6, 7]. For array-based architecture FPGAs, Fallah and Rose [3] studied the segment assignment problem in timing-driven routing. It is assumed routing resources consist of only a few types of segments, as in [1, 11]. Between global routing and detailed routing, the types of segments to be used by nets in detailed routing are chosen by a greedy algorithm to minimize the longest path delay and in general is not optimal for maximizing routability. The algorithm applies only to regular segmentations consisting of a few types of segments.

In the following section, we will discuss the problem formulation and review the model used for delay computation. Then in Section 3 we describe the switch bound allocation. The experimental results are presented in Section 4.

2 Problem Formulation

In this section, we describe the formulation of the switch bound allocation problem and briefly review the distributed RC delay model.
2.1 Switch Bounds Allocation

In row-based architecture, logic modules are grouped into rows separated by segmented channels [5, 6]. We first introduce several terms. (See Figure 1.) A net consists of a set of terminals. A routing tree of a net is a Steiner tree given by a global router. A net is a two-terminal net if it contains only two terminals. A net containing more than two terminals is a multi-terminal net. A segment of a net is an edge of the routing tree of the net. A vertical segment of a net is a segment of the net which represents the vertical routing segment assigned by the global router. A connection of a net is a segment of the net in a channel. For example, for net 1 in Figure 1, segments 1 and 3 are connections and segment 2 is a vertical segment.

Detailed routing in row-based FPGAs is performed in individual channel independently. Routability of a connection of a net depends on the upper bound of the number of switches allocated to the connection. The larger the switch bound is, the better chance the connection can be successfully routed. Thus it is reasonable to assume that the routability of a connection is given by an increasing function of the switch bound of the connection. Or, equivalently, the routing cost of a connection can be represented by a decreasing function of the switch bound of the connection.

Since the vertical segments of a net are fixed after global routing and will not be changed during channel routing phase, the routing costs of vertical segments are zero. Since detailed routings in different channels are performed independently, the routing cost of a net is defined as the sum of the routing costs of all the segments of the net.

The algorithm for two-terminal nets is simpler and provides insights to the design of algorithms for multi-terminal nets. We will formulate the problems for two-terminal nets and multi-terminal nets separately.

A two-terminal net contains a source terminal and a sink terminal, or source and sink for short. Consider a two-terminal net $N$ which has $m$ connections. Let $D$ be the delay bound from source to sink of $N$. In a switch bound allocation $\Phi$, every connection is allocated a switch bound, and the switch bound allocated to connection $i, 1 \leq i \leq m$, is denoted by $K_i$. Let $w_i, 1 \leq i \leq m$, be the routing cost function of connection $i$. The routing cost of $N$ with respect to allocation $\Phi$, denoted by $\omega(\Phi)$, is equal to $\sum_{1 \leq i \leq m} w_i(K_i)$. The delay from source to sink of $N$ with respect to $\Phi$ is
denoted by $d(\Phi)$. The allocation problem for two-terminal nets, referred to as the two-terminal net switch bound allocation problem, is formulated as follows:

**Problem 1 (Two-terminal net switch bound allocation)** Given $D, m$ and the routing cost functions $w_i, 1 \leq i \leq m$. Determine a switch bound allocation $\Phi$ such that $d(\Phi) \leq D$ and $\omega(\Phi)$ is minimized.

Consider a multi-terminal net $N = \{s, t_1, \ldots, t_n\}$, where $s$ is the source and $t_i, 1 \leq i \leq n$, are the sinks. The routing tree for $N$ is a Steiner tree $T = (V, E)$ rooted at $s$, where $V \supseteq N$. In a switch bound allocation $\Phi$ of $N$, every edge in $T$ is allocated a switch bound, and the switch bound allocated to edge $e, e \in E$, is denoted by $K_e$. Let $w_e, e \in E$, be the routing cost function of $e$. The routing cost of $N$ with respect to allocation $\Phi$, denoted by $\omega(\Phi)$, is equal to $\sum_{e \in E} w_e(K_e)$. In general, different sinks have different delay bounds, and we use $D_i$ to denote the delay bound from source $s$ to sink $t_i, 1 \leq i \leq n$. Let $d_i(\Phi)$ be the delay from source $s$ to sink $t_i$ with respect to allocation $\Phi$. The problem for multi-terminal nets, referred to as the multi-terminal net switch bound allocation problem, is stated as follows.

**Problem 2 (Multi-terminal net switch bound allocation)** Given $N = \{s, t_1, \ldots, t_n\}, T = (V, E), D_i, 1 \leq i \leq n$, and $w_e, \forall e \in E$. Determine a switch bound allocation $\Phi$, such that $d_i(\Phi) \leq D_i, \forall i \in N$, and the routing cost $\omega(\Phi)$ is minimized.

### 2.2 Distributed RC Delay Model

We use Elmore delay model [2, 16, 19] to compute source-sink delays in routing trees. To apply Elmore delay model, the routing tree of a net is modeled as an RC tree $T' = (V', E')$ in which each edge $e \in E'$ is associated with a resistance and each node $v \in V'$ is associated with a capacitance, denoted by $c_v$. Let $T_v$ denote the subtree rooted at node $v$. Let $C_v$ be the total capacitance of the subtree $T_v$, including $c_v$, and $r_v$ be the resistance of edge $e_v$.\(^1\) The Elmore delay from a node $u$

\(^1\)Note that there is only one unique edge between a node $v$ and its parent node in $T'$, and we thus can denote such an edge as $e_v$.
of \( T' \) to a descendant node \( v \) of \( u \), denoted by \( d(u, v) \), is

\[
d(u, v) = \sum_{u' \in \text{path}(u, v)} r_{u'} C_{u'},
\]

where \( \text{path}(u, v) \) is the set of nodes along the unique path from \( u \) to \( v \), excluding \( u \). Note that by replacing \( u \) with source \( s \) and \( v \) with a sink \( t_i \), we can compute the delay from \( s \) to \( t_i \), denoted by \( d(t_i) \). The Elmore delays from source to all sinks in the routing tree can be computed in linear time \( O(|V'|) \) [16, 18].

Before describing our switch bound allocation algorithms, we first show how to compute the RC delay of a chain of switches. The result will be used later in the allocation algorithms. Consider a programmable switch with on-resistance \( R_0 \) and capacitance \( C_0 \). The effect of a switch on RC delay is modeled by a two-terminal \( L \)-model equivalent circuit shown in Figure 2. Consider a chain of \( k \) switches with driver resistance \( R_d \) at the source and load capacitances \( C_L \) at the sink. The equivalent circuit is shown in Figure 3. By Equation (1), the RC delay from source to sink of the chain is:

\[
d(t) = \alpha + \beta k + \gamma k^2,
\]

where

\[
\alpha = R_d C_L, \\
\beta = R_d C_0 + R_0 C_L + \frac{1}{2} R_0 C_0, \\
\gamma = \frac{1}{2} R_0 C_0.
\]

3 The Switch Bound Allocation Algorithm

Due to space limit, only the algorithm for solving the two-terminal net switch bound allocation problem is presented in this paper. At the end of this section, we will briefly describe the basic ideas of the algorithm for solving multi-terminal net switch bound allocation problem and some extensions and applications of the algorithm.

Given a source-sink delay bound \( D \) for a two-terminal net \( N \), \( D \) is converted into an upper bound \( K \) on the total number of switches used by \( N \) by modeling \( N \) as a chain of switches \(^2\). From Equation (2), the switch bound \( K \) is determined by solving the inequality \( \alpha + \beta K + \gamma K^2 \leq D \), with the left-hand side of the inequality maximized. This gives

\[
K = \left[ -\beta + \sqrt{\beta^2 - 4\gamma(\alpha - D)} \right] / 2\gamma.
\]

\(^2\)Note that a switch is needed to connect a vertical segment and a horizontal segment. Such a switch is called crossing switch [5, 21]. Vertical segments may be segmented as well. After global routing, the number of crossing switches used in a net as well as the number of switches used in the vertical segments of a net are all known. Without loss of generality, we assume the conversion of delay bound into integer bound \( K \) for a two-terminal net has already taken into account of these switches and \( K \) represents the bound on the number of switches used in routing connections of the net.
In a **partial switch bound allocation** $\Phi_j$, $1 \leq j \leq m$, only connections $i, 1 \leq i \leq j$, are allocated switch bounds. If $j = m$, we have $\Phi_m = \Phi$. Similar to switch bound allocation $\Phi$, the routing cost of $N$ with respect to $\Phi_j$, denoted by $\omega(\Phi_j)$, is equal to $\sum_{1 \leq i \leq j} w_i(K_i)$. The upper bound on the number of switches used in $N$ with respect to $\Phi_j$, denoted by $\kappa(\Phi_j)$, is equal to $\sum_{1 \leq i \leq j} K_i$. We define a function $W : \{0, \ldots, m\} \times \{1, \ldots, K\} \mapsto \mathbb{R}$, where $\mathbb{R}$ is the set of real numbers, as follows:

$$W(j, k) = \min\{\omega(\Phi_j) : \forall \Phi_j, \kappa(\Phi_j) = k\}, \text{ for all } 1 \leq j \leq m, 1 \leq k \leq K.$$ 

That is, $W(j, k)$ is equal to the minimum routing cost of $N$ among all the partial switch bound allocations $\Phi_j$ in which $\kappa(\Phi_j) = k$. The following lemma states that $W$ is a decreasing function in the argument $k$.

**Lemma 1** For each fixed $j, 1 \leq j \leq m$, $W$ is a decreasing function in the argument $k$.

**Proof:** We prove by induction on $j$. The base case is $W(1, k) = w_1(k)$. By the definition of routing cost function of connection, $W(1, k)$ is a decreasing function. Consider $W(j, k), 2 \leq j \leq m$. For any two $W(j, k)$ and $W(j, k')$ where $k > k'$, we show that $W(j, k) < W(j, k')$. Assume $W(j, k) = W(j - 1, k_1) + w_j(k_2)$ and $W(j, k') = W(j - 1, k'_1) + w_j(k'_2)$. Let $\Delta_k = k - k'$. We have

$$W(j, k) = W(j - 1, k_1) + w_j(k_2) \leq W(j - 1, k'_1 + \Delta_k) + w_j(k'_2) \quad */ k_1 \text{ and } k_2 \text{ are optimal allocations for } k. */$$

$$< W(j - 1, k'_1) + w_j(k'_2) \quad */ \text{ induction hypothesis. */}$$

$$= W(j, k').$$

The basic idea of the switch bound allocation algorithm is to compute the function $W$ using dynamic programming and then determine an optimal allocation $\Phi$ using backtracking. First we describe the computation of function $W$. The function $W$ is initialized to $W(1, k) = w_1(k), 1 \leq k \leq K$, for $j = 1$. The function values $W(j, k)$'s for $j > 1$ are computed based on the previous partial switch bound allocation results. More specifically, the function value $W(j, k), 2 \leq j \leq m$, is computed from the optimal switch bound allocation $\Phi_{j-1}$ and the routing cost function $w_j$ of connection $j$ using the following formula:

$$W(j, k) = \min_{0 \leq k_1 \leq k} \{W(j - 1, k_1) + w_j(k - k_1)\}.$$

After computing $W$, the optimal switch allocation $\Phi$ is obtained using backtracking. The switch bound allocation algorithm is depicted in Figure 4. $k_1(i, j)$'s and $k_2(i, j)$'s are used to keep track of the optimal partial allocations. It is obvious that the algorithm can complete in time $O(mK^2)$ and the memory requirement is $O(mK)$. We have the following theorem regarding the efficiency and correctness of the algorithm. The proof of the algorithm is omitted.

**Theorem 1** The two-terminal net switch bound allocation problem can be solved in $O(mK^2)$ times using $O(mK)$ memory.
two_terminal_net()
{
    /* initialization. */
    for (j ← 1 to m)
        for (k ← 0 to K)
            W(j, k) ← +∞;
    for (k ← 0 to K)
        W(1, k) ← w_1(k);
    /* compute W(j, k) using dynamic programming. */
    for (j ← 2 to m)
        for (k ← 0 to K)
            for (k_1 ← 0 to k)
                k_2 ← k - k_1;
                if (W(j, k) > W(j - 1, k_1) + w_j(k_2)) {
                    W(j, k) ← W(j - 1, k_1) + w_j(k_2);
                    k_1(j, k) ← k_1;
                    k_2(j, k) ← k_2;
                }
    /* construct an optimal solution using backtracking. */
    k ← K;
    for (j ← m downto 2)
        K_j ← k_2(j, k);
    k ← k_1(j, k);
    Φ ← {K_j | 1 ≤ j ≤ m};
}

Figure 4: Algorithm for two-terminal net switch bound allocation.

Remark 1 The basic idea of multi-terminal net switch bound allocation algorithm is similar to the algorithm for two-terminal nets. For a given multi-terminal net \( N \), we model \( N \) by an RC tree \( T \) with the source terminal as root node and the sinks as leaf nodes. For a node \( u \) of \( T \), we define the node attribute of \( u \) as a set of ordered triples of the form \( (W_u, K_u, d_u) \), where \( W_u \) is the total routing cost of \( T_u \), \( K_u \) is the total number of switches in \( T_u \), and \( d_u \) is the maximum delay from \( u \) to any of the sinks in \( T_u \). The node attributes for all the nodes in \( T \) can be computed by dynamic programming from the leaf nodes bottom-up to the root node of \( T \). After computing node attributes, the switch bound allocation solution \( Φ \) can be obtained using backtracking from the root node.

Remark 2 Note that the switch bound allocation algorithms described previously do not include routing segment capacitances into delay computation. The capacitance of a routing segment is proportional to the length of the routing segment. In an irregular segmented channel, different
tracks usually have different segmentations. To include the capacitances of routing segments in Elmore delay computation, we can model each individual connection by a lumped equivalent circuit. The switch bound allocation algorithms for both two-terminal nets and multi-terminal nets can be easily adapted to the inclusion of routing segment capacitances.

Remark 3 The switch bound allocation algorithms can be applied to timing-driven routing of symmetrical-array based architecture. We propose that timing-driven routing to be consisted of three phases: global routing, switch bound allocation, and detailed routing. In the first phase, global routing defines the channels, but not specific tracks, used by every net. Then switch bounds are allocated on connections in the second phase. In the last phase, detailed routing determines the tracks used by each net and honors the switch bounds established in the second phase.

4 Experimental Results

The switch bound allocation algorithm is implemented in C programming language on a SUN SPARC 1 workstation. We test the algorithms on row-based architecture FPGAs.

A row-based FPGA chip is assumed to be consisting of 10 channels with 11 rows of logic modules. Each row has 100 logic modules. For each channel, we construct a 2-segmentation by using the channel segmentation design algorithms of [20]. The channel segmentations are based on the net distribution D4 used in the experimentation of [20], which is in turn derived from a set of 300 channels of connections. We pick 10 out of these 300 channels of connections for the 10 channels in the FPGA chip. It is assumed there are sufficient vertical routing segments in the FPGA chip for routing nets acrossing multiple rows.

Nets are constructed from connections in the channels as follows. We pick a number randomly and uniformly within the range of 1 to 7, inclusively, as the number of connections in a net. We assume there is at most one connection in any channel. The net can be either a two-terminal or a multi-terminal net. After a connection is chosen for a net, the connection cannot be used again for other nets. Nets are constructed in this way until all the connections in the channels are chosen. The source of a net is selected randomly from the terminals of the net. The delay bounds for sinks of nets are chosen randomly from a suitable range of delays. (Note that in practice, the delay bounds for the nets can be computed by a slack allocation algorithm, such as the zero-slack algorithm [9].)

After constructing the nets, we apply the switch bound allocation algorithms on the nets to obtain the switch bounds for all the connections in the channels. A valid track of a connection with respect to a switch bound $K$ is the track in the channel on which the connection will not use more than $K$ switches. Given a connection, let $\nu(K)$ be the number of valid tracks of the connection with respect to $K$. Note that the function $\nu$ is a non-decreasing function of $K$. We tested the switch bound allocation algorithms with three different routing cost functions: $w(K) \propto \nu(K)^{-1}$, $w(K) \propto \nu(K)^{-2}$, $w(K) \propto e^{-a\nu(K)}$, where $a > 0$ is a constant. Note that all these three routing cost functions are non-increasing functions. We then route each channel independently using a segmented channel router which is obtained from a modification of the router used in [20] so that it can handle non-uniform switch bounds on the connections. The channel router reports the number of connections
<table>
<thead>
<tr>
<th>Measurements</th>
<th>Our algorithm</th>
<th>Uniform Allocations</th>
</tr>
</thead>
</table>
|                   | \( \nu(K)^{-1} \) | \( \nu(K)^{-2} \) | \( e^{-a
\nu(K)} \) | \( K = 0 \) | \( K = 1 \) | \( K = 2 \) | \( K = 3 \) |
| Routability %     | 91.7          | 92.2               | 90.6              | 82.2       | 90.6       | 93.3       | 92.8       |
| Delay Satisfiability % | 91.7   | 92.2               | 90.6              | 82.2       | 61.7       | 66.1       | 66.1       |

Table 1: Experimental results.

successfully routed and the number of switches used by each routed connection. The routing results are measured in two parameters: routability and delay bound satisfiability. Routability is defined as percentage of nets in the FPGA chip which are successfully routed. Delay bound satisfiability is defined as the percentage of nets which are successfully routed and all source-sink delay bounds of the net are satisfied. Table 1 summarizes the experimental results. Columns \( \nu(K)^{-1} \), \( \nu(K)^{-2} \), and \( e^{-a
\nu(K)} \) are the results obtained by using our switch bound allocation algorithms with routing costs as indicated. Routability and delay bound satisfiability results are shown in the last two rows.

To see how well our switch bound allocation algorithm performs, we compare with the uniform switch bound allocation approach [5, 7]. That is, all connection in a channel are allocated the same upper bound \( K \). We choose \( K = 0, 1, 2, 3 \), and perform \( K \)-switch channel routing for each case 3. The results are also shown in Table 1 in the columns \( \text{"} K = 0 \text{"} \) to \( \text{"} K = 3 \text{"} \). For \( K = 0 \), we see that both routability and delay bound satisfiability produced by our allocation algorithms are better than 0-switch channel routing by up to 10\%. For \( K \)-switch channel routing with \( K > 0 \), the routability results produced by our algorithms and the uniform allocation algorithm are similar. However, the delay bound satisfiability results produced by our algorithms are significantly better than that produced by uniform allocation approach by 26\% to 30\%.

From the results, we also observe that switch bound allocation algorithm is not very sensitive to the choices of routing cost function.

References


3Note that \( K \)-switch channel routing is equivalent to \( (K + 1) \)-segment channel routing.


