An Effective Hop-Based Detailed Router for FPGAs for Optimizing Track Usage and Circuit Performance*

Hasan Arslan and Shantanu Dutt
Dept. of ECE, University of Illinois-Chicago
Chicago, IL 60607
{harslan, dutt}@ece.uic.edu

ABSTRACT

We have developed a hop-based complete detailed router *ROAD-HOP* that uses the Bump & Refit ($B\&R$) approach to route a FPGA circuit in a near-optimal manner. This approach is based on generating a minimum-spanning tree (*MST*) from the complete pin-to-pin graph of each net with each edge cost based on a combination of its contribution to the net length, channel congestion and potential average "bumping" cost in the channels in which the edge lies. Using the *MST*, a hop-based routing of each net is performed that attempts to minimize the combination of net length, number of hops and total number of tracks needed in the FPGA. Given each net's global route, a FPGA detailed router can minimize net delays by minimizing the number of hops or equivalently the number of track switchings in complex switchboxes of current FPGAs—hop-based routing can model routing using complex switchboxes. By minimizing the number of hops and total net length, *ROAD-HOP* minimizes net delay. Note that *ROAD-HOP* can only be compared to another detailed router and we compare it to the best previous detailed router *SEGA* for the *VPR* architecture. We use the output of the *VPR* global router as input to both *ROAD-HOP* and *SEGA*. Our new algorithm achieves significantly better results than *SEGA* with respect to the number of tracks needed and the circuit speed. Across a number of benchmark circuits, our algorithm needs about 8% fewer tracks than *SEGA*. Furthermore, the average net delay of the routing generated by our algorithm is 34% less than that of *SEGA*, and is 52% less for the longest net.

Categories and Subject Descriptors:
B.7 Integrated Circuits: B.7.2 Design Aids, J.6 Computer-Aided Engineering: J.6.0 Computer-aided design (CAD)

General Terms:
Algorithms, Design

Keywords and Phrases:
FPGAs, Bump and refit paradigm, MST, bumping cost, detailed routing, switchbox, hop-based routing

*This work was supported in part by NSF grant CCR-0204097.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
GLSVLSI'04, April 2628, 2004, Boston,Massachusetts, USA.
Copyright 2004 ACM 1-58113-853-9/04/0004 ...$5.00

1. INTRODUCTION

Recent FPGA routing results have suggested that a separate global and detailed routing strategy is inferior to a combined routing process. Similarly, the practice of dividing multipoint nets into multiple two-point nets for routing was thought to negatively impact routability. The results of VPR [7] which has been developed lately as a flat router (global and detailed routing performed in an integrated manner) had used significantly fewer tracks to route circuits than the best-known two-step routers, *CGD* [6] and *SEGA* [5]. However, a recently developed order-impervious detailed router *ROAD* [4] gives promising results when supported by a decent global routing phase. This augurs well for routing large FPGA circuits using 2-phase routing, as such a divide-and-conquer paradigm is more scalable with circuit size than the flat routing paradigm exemplified by VPR.

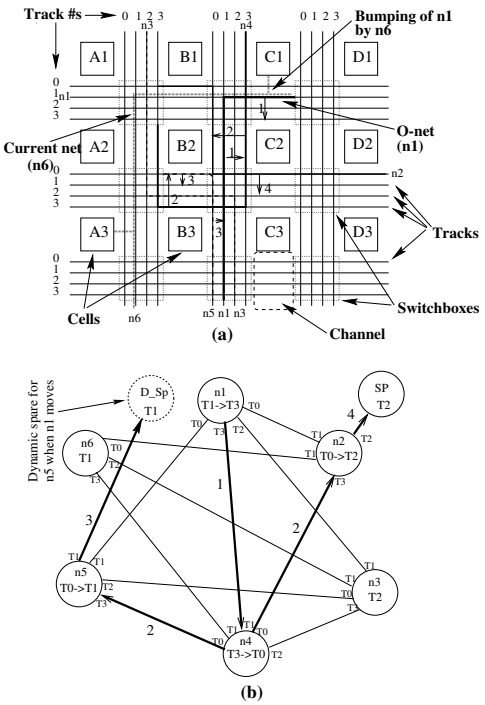

Figure 1: *(a) Routing in an FPGA, and a $B\&R$ process for current net n_6 connecting cells $A3$ and $C1$; for simplicity pin connections of existing nets are not shown. Nets are shown by dark or dotted lines on the tracks. Numbered arrows from existing nets show the sequence of bumpings to accommodate the routing for n_6. (b) Searching the OG for a converging transition DAG for the O-net n_1. Track labels at each end of an edge in the OG refers to the track on which the neighboring net corresponding to that edge lies.*

In this paper, we present an efficient hop-based detailed routing algorithm *ROAD-HOP* which uses a $bump\text{-}and\text{-}refit$ ($B\&R$) strategy and track hops from either source or input/output pins to route a circuit using first a minimum number of tracks and then using a minimum number of hops. Track hops are an approximation of a general switchbox structure and thus can be used to simulate and model complex switchboxes found in current FPGAs. *ROAD-HOP* yields much improved routing quality compared to the previously best-known detailed router *SEGA*, in terms of both the number of tracks and net delays.

The rest of the paper is organized as follows. Section 2. discusses the basics of the $B\&R$ methodology proposed in [1, 2] and explains how such a methodology is used for an order-impervious optimal detailed router *ROAD* for FPGAs with i-to-i switchboxes. Section 3. explains how we can model i-to-j switchbox structures used by modern FPGAs by using hops from source or from I/O pins. We explain our hop-based detailed routing algorithm *ROAD-HOP* in Section 4.. Experimental results for a set of small to large VPR benchmark circuits are given in Sec. 5. and we conclude in Sec. 6..

2. B&R-BASED HOP-LESS ROUTING FOR FPGAs

In this section, we set the stage for using $B\&R$ in hop-based routing by recapping the basic $B\&R$-based hop-less router *ROAD* for FPGAs with i-to-i switchboxes from [4].

We first define some terminology pertaining to FPGAs; Figure 1a illustrates many of these terms. We define a *channel* in an FPGA as the set of all track segments between two adjacent *switchboxes (SBoxes)* of the FPGA; see Figure 1a. Each channel has the same number t of tracks, which we denote by $T_0, T_1, \ldots, T_{t-1}$. The length of a *track segment* is the number of channels it spans before it needs to connect to another segment via a SBox. For simplicity of exposition, we describe our $B\&R$ algorithm for FPGAs with track segments of length one.

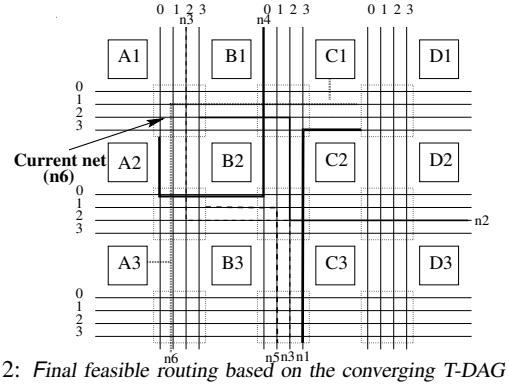

Figure 2: *Final feasible routing based on the converging T-DAG of Figure 1b.*

2.1. Single-Net Routing Using B&R

After the global routing of all nets is determined (in [4] as well as here we use *VPR* to perform global routing), the nets are detailed routed in some sequential order. Suppose after routing $i-1$ nets, $i-1 \geq 1$, the number of tracks occupied by these nets is t. For routing the next net n_i, *ROAD* attempts to pack it among the occupied t tracks even if some empty tracks are available. Net n_i is assigned a track among the t tracks in which its "bumping cost" or "transition cost", defined later, is minimized. The rest of this subsection describes how n_i is routed on its assigned track using the B&R paradigm[1].

[1] A corollary of Theorem 1 stated later is that if there is a solution to routing n_i among the occupied t tracks it will be found by the B&R algorithm `Conv-T-DAG` (Fig. 3) irrespective of which track is assigned to net n_i.

For the next net n_i to be routed, if the required track segments on its assigned track are vacant, then no existing nets are bumped and we have a solution for n_i. Note that the transition cost function (defined later) will always assign a track to n_i that has all vacant segments along the channels in n_i's global route if such a track exists. However, if no such track exits, then n_i will use one or more segments that are occupied by other net(s), and the routing of n_i will cause a "bumping" of these set of nets called the *occupying set*; each net in this set is called an *occupying net (O-net)*. In Figure 1a, n_6 is the current net to be routed that connects cells A3 and C1. The track assignment of n_6 bumps only one net n_1; it is thus the only O-net. In general, refitting solutions for each O-net can be explored independently and in any order to arrive at a feasible refitting of all bumped nets (see Theorem 1 given later). In the sequel, without loss of generality, we thus describe the $B\&R$ process for a single O-net. The O-net needs to be moved out of its current track to make space for the current net. We use $n_i^{T_j}$ to denote a net n_i on track T_j. Let a *transition* be defined as the movement of net n_i on a track T_j to another track T_k, and is denoted by $n_i^{T_j \rightarrow T_k}$. This transition may result in net n_i bumping into one or more nets on track T_k. These nets in turn will have to move out of their current track T_k, giving rise to a transition for each of them. This transition sequence is shown in Figure 1b by dark arcs, where net n_1 initiates a set of transitions which finally terminate in "spare" track segments, which are vacant segments of appropriate total lengths into which bumped nets can move without bumping other nets. As seen in the Figure 1b, the set of transitions take on a directed-acyclic graph (DAG) structure, termed a *transition DAG (T-DAG)*, with the spares forming the leaf nodes. The current net's routing is successful if a T-DAG rooted at the corresponding O-net can be found whose leaves are spare segments; such a T-DAG is termed a *converging T-DAG*.

We introduce the concept of an *overlap graph (OG)*, which is a graph representation of the circuit routing in the FPGA. The OG is an undirected graph with the circuit nets represented by the nodes of the graph. In the overlap graph OG(V,E), the set of nodes $V = S \cup \{n_1, n_2, \ldots, n_m\}$, where each n_i is a routed net of the circuit and S is the set of "spare" track segments as described above (please note that a set of vacant segments of a track T_j is a spare only with respect to specific net(s) that can be moved to T_j without bumping any nets on that track). There exists an edge between n_i and n_j in the OG if nets n_i and n_j share a channel in the FPGA. Figure 1b shows the OG for the routing of Fig. 1a. In the OG, for e.g., nets n_1 and n_5 have an edge between them since they are routed through a common vertical channel to the right of cell B3. The OG can be used to determine if the required converging T-DAG exists. Since the OG represents the circuit routing in the FPGA, a T-DAG is a DAG embedded in the OG (the undirected edges of the OG become directed arcs in the direction of the transitions; see Fig. 1b). Thus a converging T-DAG rooted at an O-net can be determined by performing a search on the OG until all leaf nodes of the search DAG are spare nodes. This process is illustrated in Figure 1 for a small circuit and for the current net n_6. The corresponding O-net n_1 transits from T_1 to T_3 and bumps into n_4. The movement of n_1 from T_1 creates a "dynamic" spare node (labelled by D_Sp in Figure 1b) for net n_5. The bumped net n_4 then transits from T_3 to T_0 where it bumps n_5 and n_2. n_5 then transits to the above dynamically created spare on T_1, while n_2 transits to its spare track segments on T_2. Thus a converging T-DAG is determined in the OG. The transition arcs are shown dark in Figure 1b and numbered chronologically in the order in which they are traversed in the search process. Figure 2 shows the final routing of the FPGA after the bumping sequence converges.

2.2. An Optimal Depth-First B&R Algorithm

A $B\&R$ based single-net routing algorithm `Conv-T-DAG` (Figure 3) that performs a depth-first based search in the OG for a con-

```
Algorithm Conv-T-DAG(OG,n_i^{T_j}) /*Find a converging
                T-DAG rooted at n_i^{T_j} */
for T_k=(T_{i_1},.....,T_{i_t}) in order of increasing TC do begin
    if (adj^{T_k}(n_i)==0) then  /*there exists a spare for n_i on T_k*/
        return (success)
    else begin
        for each n_r ∈ (adj^{T_k}(n_i) do begin
            if (n_r is an ancestor) then break;
            /*this transition to T_k results in a cycle, or
            there is a failure try the next best transition */
        for each n_r ∈ (adj^{T_k}(n_i) do
            result=Conv-T-DAG(OG,n_r^{T_j} );
            if (result==fail) then break;
            else numb_succ=numb_succ+1;
        endfor
            if (numb_succ==|adj^{T_k}(n_i)|) then
                    return(success);
/*converging T-DAGs were found for all nets in adj^{T_k}(n_i) */
    endelse
endfor
return(fail); /* no transition of n_i was successful */
End. /* Conv-T-DAG() */
```

Figure 3: *The optimal depth-first search algorithm for finding a converging T-DAG*

verging T-DAG rooted at the O-net was developed in [2]. For a transition of the O-net to some track T_k, it recursively searches for converging T-DAGs rooted at each net on T_k bumped by the O-net. A depth-first path terminates in success if a spare node is reached, and in failure either when all OG nodes have been visited in that path or a cycle is detected (an ancestor along the current path is revisited). When an $n_i^{T_j \rightarrow T_k}$ transition fails in this manner, the search backtracks and tries an unexplored transition $n_i^{T_j \rightarrow T_l}$. The following result establishes the optimality of `Conv-T-DAG` [2].

THEOREM 1 *[2] If a converging T-DAG exists (among the currently used tracks of the FPGA) for the O-nets bumped by the current net's routing,* `Conv-T-DAG` *will find it .*

While an existing converging T-DAG will ultimately be found by `Conv-T-DAG`, it will be time-efficient if some suitable "cost" measure can be used to determine which transitions are more likely to be successful so that fewer T-DAGs are searched and backtracked. A good cost measure will consider both the "magnitude" of bumpings (total length of bumped nets) and the likelihood of convergence of these bumpings. Two transition cost (TC) measures evaluated are as follows:

(1) $TC_1^{sum}(n_i^{T_j \rightarrow T_k}) = \sum_{n_j \in adj^{T_k}(n_i)} l(n_j)$,

where $adj^{T_k}(n_i)$ are the neighbors of n_i in the OG that are on track T_k, and $l(n_j)$ is the total length of n_j in terms of the track segments (each of length 1) that it occupies. This heuristic is reasonable, but only considers the bumping magnitude. For example, according to it, it is equally costly to bump a net of length 9 as it is to bump 3 nets each of length 3. However, the latter case has a higher likelihood of convergence since there is greater flexibility in moving 3 bumped nets than a single net of the same total length. This leads to the next cost function.

(2) $TC_1^{sqrt}(n_i^{T_j \rightarrow T_k}) = [\sum_{n_j \in adj^{T_k}(n_i)} l(n_j)]/\sqrt{|adj^{T_k}(n_i)|}$.

Using such TC functions to guide the search results in time-to-solution reduction by an order of magnitude compared to a "blind" depth-first search [2]. We term the above TC functions as *1st-level TC functions*.

2.3. Applying B&R to Complete Detailed Routing

In Fig. 4, the pseudo code `Route-With-B&R` for hop-less detail routing of a given set of nets using the $B\&R$ strategy is given. For

```
Algorithm Route-With-B&R(N) /*assign tracks to nets in N*/
Begin
    The overlap graph OG = ∅;
    for each n_i ∈ N
        Insert n_i in the OG; /* update neighbor nets TC */
        Get the track T_k for which n_i has the least TC;
        Assign n_i to track T_k;
        Let S be the set of the nets bumped on T_k by n_i,
        for each n_c ∈ S do
          result = Conv-T-DAG(OG,n_c^{T_k});
          if (result == fail) then
            increase track number, put n_i on new track;
            restore track assignments of all bumped nets
            to that prior to n_i's assignment to T_k;
          endif
        endfor
    endfor
End.
```

Figure 4: *Algorithm for hop-less detailed routing using B&R*

the next net n_i to be routed in the channels designated by the global router, a track T_k is chosen for which n_i's TC is the least. If n_i does not bump any net in T_k, we are done. Otherwise, a $B\&R$ solution is obtained for every O-net n_c bumped by n_i on T_k by calling `Conv-T-DAG`. If such a solution does not exist for some O-net n_c, the number of tracks in the FPGA is increased by one, n_i placed on the new track and the track positions of all other nets are restored.

The next theorem establishes the optimality of `Route-With-B&R`.

THEOREM 2 *[4]* `Route-With-B&R` *obtains a final detailed routing solution with the minimum number of tracks irrespective of the order in which the nets are routed.*

For detailed routing of large circuits, the time-efficiency of the DFS algorithm for `Conv-T-DAG` (Fig. 3) becomes critical as many prior routed nets are bumped leading to extensive DAG searches. The crux of practically applying this optimal algorithm to the detailed routing problem for large FPGA circuits is to determine significant search-space prunings that will not sacrifice optimality, as well as to develop much better DFS ordering heuristics than those given by the 1st-level TC functions of Sec. 2.2. so that paths that are most likely to lead to solutions are explored first. This has been accomplished in the *ROAD* algorithm of [4] where a speed up factor of around 5000 times over the basic `Route-With-B&R` algorithm. These speedup methods are also used in the *ROAD-HOP* algorithms, discussed shortly, that perform hop-based detailed routing.

3. MODELING SWITCHBOXES

FPGA switchboxes have either i-to-i or an i-to-j structures. In an i-to-i switchbox, a segment on track T_i on one side of the switchbox is only connected to another segment of track T_i on the other three sides of the switchbox. In an i-to-j switchbox structure, a segment on track T_i is allowed to be connected to segments on other track(s) T_j besides segments on track T_i. The advantage of i-to-j switchboxes is that if among the set of available tracks, a net cannot be routed on the same track throughout its global topology, then a successful routing of the net can still be performed by routing subparts of the net on different tracks that are inconnected by such switchboxes. Thus i-to-j switchboxes are more flexible and afford a larger solution space to detailed routing than i-to-i switchboxes. Currently, most modern FPGAs use i-to-j switchboxes.

The FPGA architecture that we are using for routing has i-to-i switchboxes. During routing in such an architecture it is desirable to have the flexibility afforded by i-to-j switchboxes of being able to route the subnets of a multi-pin net on different tracks. This can

be accomplished with i-to-i switchboxes by creating hops (connection of a net pin to two or more track segments in those channels). We use two types of hops which allow us to route subnets of a multi-pin net on different tracks in order to model i-to-j switchboxes. In the first type of hop, we create hops from the source pin only which means all sink pins are directly connected to the source pin. The second type of hop is one in which hops are created at all possible net pins (input/output [I/O] pins) and afford more flexibility for routing. Figure 5a shows a global topology of a multi-pin net. It is split into two-pin nets by using either hop from source as seen in Fig. 5d or hops from I/O pins as seen in Fig. 5e. Figure 5f shows a similar routing to that of Fig. 5e but using i-to-j switchboxes instead of hops at I/O pins. In both routings, the multi-pin net is divided into subnets that are routed on different tracks. In Fig. 5e, the tracks occupied by the net are changed at I/O pins, whereas in Fig. 5f they are changed at switchboxes.

4. HOP-BASED DETAILED ROUTER

For detailed routing the main objective is to route all nets in the circuit by using a minimum number of tracks. As described in Sec. 3. to be able to do that we have to create hops to allow more flexibility in determining different routing patterns for each net. However, at the same time we must avoid creating unnecessary hops to reduce the number of used segments and to keep a net's delay close to optimal for a given global topology. We have extended *ROAD* [4] to perform such hop-based detailed routing for FPGAs. There are two phases in our hop-based detailed router.

In phase 1, the router splits all multi-pin nets into two-pin nets as explained below and sorts them according to their lengths. If nets are to be routed using hops from I/O pins, a minimum spanning tree (*MST*) is generated from the complete pin-to-pin graph of each net; see Fig. 5c. For hops from source only, a two-pin-net list is constructed between the source and all sink pins as shown in Fig. 5b.

In phase 2, a hop-based routing is performed for each net that attempts to minimize a combination of net length in terms of total segment usage, circuit delay and total number of tracks needed in the FPGA.

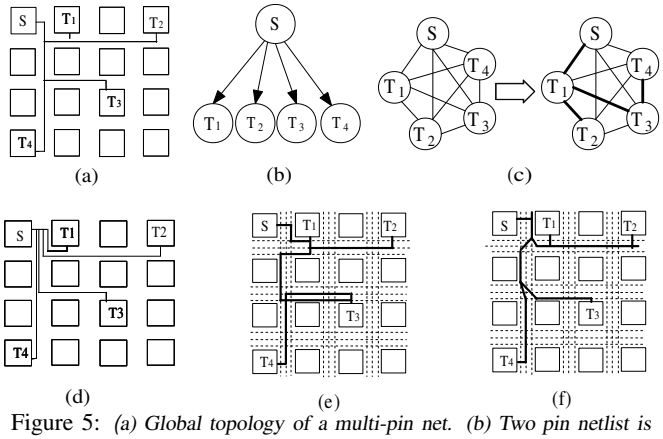

Figure 5: *(a) Global topology of a multi-pin net. (b) Two pin netlist is constructed between the source and all sink pins (c) Generating a MST from the complete pin-to-pin graph. (d) Multi-pin net in [a] is split into two-pin nets using hop from source. (e) Each edge in the MST in [c] is converted back to a two-pin net, and routed on different tracks according to the global topology of the net. It models routing using i-to-j switchboxes which is shown in (f).*

4.1. Phase 1: Splitting Multi-Pin Nets into Two-Pin Nets

For hop-based routing, we define a global routing tree (*GRT*) in which the nodes comprise of pins and branch points, and edges representing the paths between them. Branch points in *GRT* represent switchboxes where net n_i is split into branches; see Fig. 6. Based on the paths between the *GRT*'s pin nodes, we form a global routing graph (*GRG*) which represents the complete pin-to-pin routing between the pin nodes in terms of two-pin-nets. We define the *interior pin node (IP)* as non-leaf pin node. We have used the edge cost given below for edges of the *GRG* to find a *MST* to obtain the final interconnection routing.

The cost of an edge between two nodes in *GRG* consists of a measure of congestion on the channels where that net goes through and potential net bumping cost on those channels. Also each channel has a base cost which takes care of the net length; more channels on the path implies a longer net and hence higher cost. The edge cost in the *GRG* is given by:

$$\textit{Edge-cost} = \sum_i (\alpha.\textit{base-cost}_i) + \beta.\textit{density}_i.\textit{Avr}(\textit{net-length}_i)$$

where i is a channel that net goes through. α and β are the weighting factors for the net length and channel congestion metrics, respectively. The second term in the above cost equation is the average of the total length of the nets in the channel times the probability of bumping a net in that channel (i.e., the channel density taken as the fraction of occupied tracks); this captures the potential average bumping cost that may be incurred in the detailed routing phase. For e.g., a channel with longer nets is costlier with respect to bumping than a channel with the same number of shorter length nets.

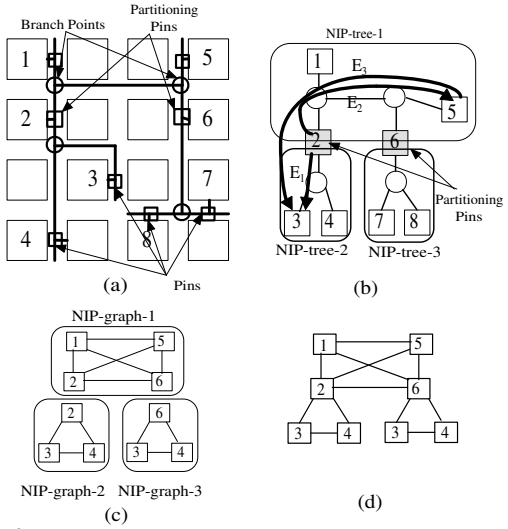

Figure 6: *(a) Global topology of a multi-pin net (b) GRT and three NIP subtrees derived from [a]. Branch points are shown as circles and pins as rectangles. (c) Independent NIP graphs. (d) Union of NIP graphs in [c].*

For large circuits, it will be time consuming to determine an *MST* in the full *GRG*. Fortunately, it is possible to reduce the *MST* computation time without loss of optimality in finding the *MST* by first partitioning the *GRT* at *IP* nodes into a number of non-interior-pin (*NIP*) trees; each IP node is replicated in both *NIP* trees obtained by partitioning the *GRT* at that node. IP pins are also called *partitioning pins*. The partitioning process is shown in Fig. 6. Based on the global-routing paths between the *NIP* tree's node pins, we form independent *NIP* graphs which are complete graphs of the nodes of their respective *NIP* trees; these graphs are subgraphs of the *GRG*. Then, independent *MST*s of each of these *NIP* graphs are constructed. If there are m nodes in the *GRG*, then it has $\Theta(m^2)$ edges and the *MST* determination time in it is $\Theta(m^2)$. If the *GRT* can be partitioned into k *NIP* trees, and assuming they are of roughly equal sizes, then forming the independent *MST*s in them takes a total of $\Theta((m/k)^2)$ time, a speedup factor of $\Theta(k^2)$ over forming an *MST* in the *GRG*. The following result established the equivalency of the two *MST* determination techniques.

THEOREM 3 *The union of the independent MSTs of the NIP graphs is an MST of the GRG.*

Proof Sketch: If we obtain a *MST* directly from the full *GRG* (without partitioning), each edge of such an *MST* will belong to only one *NIP* graph. The reason for this is that an edge which passes through a "partitioning channel" (a channel in which an IP is present) to connect two pins which belong to two different *NIP* graphs will not be selected in the *GRG*'s *MST* because of its higher cost compared to other edges. As shown in the Fig. 6b, if we want to connect pin 5 and pin 3 to create a two-pin-net, this two-pin-net will pass through the channel where the partitioning pin 2 is connected. Since the cost of the edge between pin 5 and pin 3 (E_3) will be more than the individual costs of the edges between pins 2 and 3 (E_1) and pins 2 and 5 (E_2) and since the latter two edges connect nodes 2, 3, 5 while E_3 connects only a subset of these nodes 3, 5, E_3 will never be chosen in an *MST* obtained directly from the *GRG*. A similar argument holds for any non-NIP-graph edge of the *GRG* and thus such edges can never be in any *MST* of the *GRG*. If these edges are eliminated from the *GRG*, then the remaining graph is exactly the union of independent *NIP* graphs as shown in Fig. 6c-d. Hence the union of the independent *MST*s of the *NIP* graphs will be an *MST* of the *GRG*. □

After the *MST* is obtained, each edge in the *MST* is converted back to a two-pin net that follows the global route and connects the pins as shown in Fig. 5e, for the net of Fig. 5a.

4.2. Phase 2: Path Selection

Let N_i be a multi-pin net, P_i be the set of two-pin-nets of N_i, R_i be the subset of P_i whose two-pin nets have been routed, and K_i be the set of tracks on which nets in R_i are routed. After phase 1, each two-pin net $m \in P_i$ might be routed on different tracks. Our objective is to route all nets in P_i on the least possible number of tracks without increasing the total number of tracks in the FPGA. That way, first, the number of FPGA tracks is minimized and secondly the number of hops (equivalently, the number of track changes via i-to-j switchboxes in architectures with such switchboxes) for each net is minimized; the latter minimization translates to a minimization of the delay on each net incurred due to track hops—hops increase the resistive and capacitive load of the net—(or equivalently due to track switchings at i-to-j switchboxes). The pseudo code for this phase is given in Fig. 7 where the key idea is that it tries to maximize the sharing of wire segments among two-pin nets $r_i \in R_i$. Every time a net $m \in P_i$ is to be routed, if $R_i = \emptyset$ we directly route net m by calling $Route\text{-}With\text{-}B\&R$. If $R_i \neq \emptyset$, we try to route net m on a track $T_j \in K_i$, with which net m has maximum sharing with nets in R_i. Selection is done in decreasing order of shared length.

If there does not exist a solution to route net m on any track $T_j \in K_i$, a hop is then created and that net m is routed by calling the $Route\text{-}With\text{-}B\&R$ again, which now tries to route net m on track $T_l \notin K_i$ without increasing the total number of tracks. If $B\&R$ still can not find a solution, this net is skipped and a new unrouted two-pin net is chosen, until all two-pin-nets are tested once to find solutions on the set of currently occupied tracks. The number of tracks is then increased, and the remaining unrouted nets are tested again. These steps are repeated until all nets are routed.

Note that there is a trade off between a reduction in the number of FPGA tracks and net delays. As we attempt to reduce the number of tracks by exploring various possible net routing patterns via hop creation, these hops increase net delays. However, all possible hops or even a great many of them may not be necessary to minimize the number of FPGA tracks, and we should create hops judiciously. Our algorithm described above tends to minimize the number of hops by routing the two-pin subnets of a net in a continuous (as opposed to a fragmented) manner and maximizing sharing of track segments of the current 2-pin subnet with those of previously routed 2-pin subnets of the same net. Therefore it results in significant

```
Algorithm Path-Selection(S) /*assign tracks to nets in S*/
Input: Two-pin netlist S sorted in decreasing order of lengths;
Output: Detailed routing result
Begin
   F = ∅; /* F keeps the subnet cannot be routed on used tracks*/
   While (S ≠ ∅) do     /* until all two-pin nets are tested */
      L_N =Longest-Two-Pin-Net(S);
            /*maximize the sharing of wire segments if fails*/
            /*call B&R to route on separate track by creating hop */
      result=Route-With-Max-Shared-Net(L_N );
      if (result==fail) then
         result_2=Route-With-B&R(L_N );
         if (result_2==fail) then /* cannot be routed on used tracks.
            F ⇐ F ∪ L_N ; /* put it in failure net set to route later;*/
         endif
      endif
   endwhile;
   if(F ≠ ∅)then
      Active_Track_Number++;
      Path-Selection(F);
   endif
End. /* Path-Selection */
```

Figure 7: *Pseudo code for hop-based detailed router*

Circuit Name	FPGA Size	# Net	# Pin	Circuit Name	FPGA Size	# Net	# Pin
alu2	15x15	236	1001	misex3c	24x24	663	2773
sand	16x16	285	1236	ex5p	33x33	1072	5391
planet	17x17	307	1357	tseng	33x33	1248	5409
x4	21x21	310	1136	misex3	38x38	1658	7013
vda	18x18	399	1511	alu4	40x40	1748	7632
frg2	36x36	558	2177	diffeq	39x39	1786	7588
apex6	30x30	575	2199	apex2	44x44	2284	9431
mm30a	23x23	651	2634	elliptic	61x61	4175	15565

Table 1: *Characteristics of VPR benchmark circuits.*

improvement in net delay compared to other hop-based routers like SEGA. The delay is determined by calculating the Elmore delay [8] of the RC-tree network(s) of each net. The delay of a net is defined as the largest delay from the source pin to any of sink pins. Under the Elmore delay model, the signal delay from source node S_0 to a sink node u is given by the following equation:

$$D_{elmore}(S_0, u) = \sum_{e,v \in Path(S_0,u)} r_e(\frac{c_e}{2} + C_v),$$

where e is the "current" track segment and v the switch at the switchbox connecting e to the next track segment of the net, c_e is the capacitance of track segment e and the pass transistor switch v, r_e is the resistance of track segment e and the pass transistor switch v, and C_v is the total downstream capacitance rooted at node v.

5. EXPERIMENTAL RESULTS

We call the set of techniques described in the previous section including the two phases for net splitting into 2-pin subnets and of path selection as the *ROAD-HOP* router. We ran *ROAD-HOP* and *VPR* on 550MHz Pentium-III Linux and *SEGA* on 440MHz Sun Ultra 5-10 Solaris machines. To compare *SEGA* runtime with *ROAD-HOP* runtime, we obtained the speed ratio and scaled *SEGA* runtime to the Pentium-III equivalent time. For each circuit shown in Table 1 we have used VPR [7] to get the global topology of each net. Then we used *ROAD-HOP* to perform hop-based detailed routing. For *ROAD-HOP*, the number of tracks needed for routing, as well as the number of used segments, circuit delays and runtimes were collected and compared with *SEGA*'s results. To show how *SEGA* and *ROAD-HOP* reduce the number of tracks when they use hops, we compare their results with *ROAD* [4] which routes nets without creating any hops and by using the optimal number of tracks under this conditions. The versions of *ROAD-HOP* and *SEGA* that use hop from source for routing are called *ROAD-HOP*^{SRC} and *SEGA*^{SRC}, respectively, while those versions that use hop from *I/O* pins are called *ROAD-HOP*$^{I/O}$ and *SEGA*$^{I/O}$, respectively.

Circuit Name	SEGA^{SRC} tracks	SEGA^{SRC} used sgmnts.	SEGA^{SRC} time	ROAD tracks	ROAD used sgmnts.	ROAD time	ROAD-HOP^{SRC} tracks	ROAD-HOP^{SRC} used sgmnts.	ROAD-HOP^{SRC} time
alu2	9	1829	2.12	10	1654	2.26	8	1692	10.31
sand	11	2452	8.34	11	2106	2.86	10	2121	33.43
planet	10	2401	7.88	10	2101	3.23	9	2107	17.27
x4	8	2023	1.42	8	1875	4.49	7	1877	5.16
vda	14	3816	7.14	15	3131	4.8	12	3254	20.48
frg2	8	4494	4.80	9	4105	14.44	8	4108	28.82
apex6	9	4511	5.55	8	4067	10.85	8	4067	22.02
mm30a	9	4442	15.61	9	3866	11.14	8	3869	22.43
misex3c	13	6259	34.94	16	5512	13.75	12	5602	57.53
ex5p	17	17263	483.77	18	15304	51.97	17	15309	144.26
tseng	11	8906	142.42	10	7846	44.46	9	7865	100.94
misex3	18	20708	859.47	23	17419	96.81	17	17560	718.17
alu4	16	20623	2789.66	21	17413	127.83	16	17984	1198.8
diffeq	13	14145	326.79	12	12428	91.24	11	12462	235.12
apex2	22	30175	1745.86	29	24990	192.14	21	26025	1033.92
elliptic	19	44570	6722.36	25	36791	775.6	18	37935	1953.25
Total	**207**	**188617**	**13158.11**	**234**	**160608**	**1447.87**	**191**	**163837**	**5601.91**
Imp. over SEGA^{SRC}				**-12%**	**17%**		**8%**	**15%**	
Speedup over SEGA^{SRC}						**9**			**2.4**

Table 2: *Comparison of ROAD, ROAD-HOP^{SRC} and SEGA^{SRC} in terms of number of used segments, number of tracks and runtimes (secs).*

Circuit Name	Circuit Speeds: Longest Net SEGA^{SRC}	Longest Net ROAD HOP^{SRC}	Longest Net Imp over SEGA^{SRC}	Average SEGA^{SRC}	Average ROAD HOP^{SRC}	Average Imp over SEGA^{SRC}
alu2	37.941	11.26	70.32%	4.061	2.90	28.69%
sand	39.948	28.12	29.61%	3.784	2.90	23.35%
planet	129.69	19.8	84.73%	3.518	1.98	43.62%
x4	120.557	28.14	76.66%	5.315	2.90	45.37%
vda	288.283	170	41.03%	11.229	7.88	29.84%
frg2	134.336	29.38	78.13%	6.91	3.96	42.62%
apex6	72.32	48.46	32.99%	7.011	5.53	21.14%
mm30a	86.056	69.91	18.76%	5.366	4.69	12.67%
misex3c	160.741	114.19	28.96%	6.682	5.33	20.28%
ex5p	412.45	181.85	55.91%	5.72	3.47	39.31%
tseng	103.45	55.98	45.89%	1.78	1.31	26.44%
misex3	440.686	224.75	49.00%	13.101	6.93	47.11%
alu4	587.682	199.22	67.81%	8.145	3.35	62.01%
diffeq	245.19	92.44	62.30%	1.87	1.02	45.47%
apex2	313.469	184.1	41.27%	12.277	8.08	34.22%
elliptic	643.25	328.57	48.92%	9.34	5.88	37.07%
Total	**3764.05**	**1786.17**	**52%**	**108.11**	**68.10**	**34%**

Table 3: *Comparison of ROAD-HOP^{SRC} and SEGA^{SRC} circuit speeds (ns).*

Circuit Name	SEGA$^{I/O}$ tracks	SEGA$^{I/O}$ used sgmnts.	SEGA$^{I/O}$ time	ROAD-HOP$^{I/O}$ tracks	ROAD-HOP$^{I/O}$ used sgmnts.	ROAD-HOP$^{I/O}$ time
alu2	7	1888	1.93	7	1710	7.32
sand	7	2397	2.46	7	2185	10.27
planet	7	2371	2.84	7	2140	11.97
x4	7	2060	2.05	6	1991	8.13
vda	9	3601	4.3	8	3312	15.43
frg2	7	4519	6.02	6	4128	27.8
apex6	6	4449	5.29	6	4132	14.99
mm30a	7	4411	12.16	6	4017	37.39
misex3c	9	6320	10.43	8	5767	19.61
ex5p	14	17242	50.91	13	15477	438.95
misex3	12	19742	80.16	11	17612	77.84
alu4	11	19867	69.71	10	18213	173.63
apex2	12	28277	146.4	11	26422	154.7
Total	**115**	**117144**	**394.66**	**106**	**107106**	**998.03**
Imp. over SEGA$^{I/O}$				**7.83%**	**8.57%**	
Speedup over SEGA$^{I/O}$						**1/2.53**

Table 4: *Comparison of ROAD-HOP$^{I/O}$ and SEGA$^{I/O}$ in terms of number of used segments, number of tracks and runtimes (secs).*

Table 2 and Table 3 show the results when hop from source is allowed for the circuits of Table 1. In Table 2 we compare the number of tracks used by three algorithms, *ROAD*, *ROAD-HOP^{SRC}* and *SEGA^{SRC}*. *ROAD* routes all circuits without using any hop and uses 12% more tracks than *SEGA^{SRC}* . However *ROAD* is able to route some of the circuits using either the same number of tracks as *SEGA^{SRC}* or less. This means that even if there exists a solution to route the nets in those circuits without using any hop, *SEGA^{SRC}* cannot find that solution, whereas *ROAD* and *ROAD-HOP^{SRC}* are able to find it. Also, *ROAD* uses 17% less segments and is 9 times faster than *SEGA^{SRC}*. *ROAD-HOP^{SRC}* uses 8% fewer tracks than *SEGA^{SRC}* and is 2.4 times faster than *SEGA^{SRC}*. In addition to that, the number of segments used by *ROAD-HOP^{SRC}* does not increase significantly over that of *ROAD*; *ROAD-HOP^{SRC}* uses only 2% more segments to reduce the number of used tracks by 20% compared to *ROAD*. By contrast, *SEGA^{SRC}* uses 17% more segments than *ROAD* to reduce number of tracks by 12%. It is thus clear that *ROAD-HOP^{SRC}* creates hops very efficiently. As a result of this, most of the nets are routed on a single track, which implies that circuit speeds are close to optimal for the given global net topologies. Across the benchmark circuits, average net delay for *ROAD-HOP^{SRC}* is 34% less than the average net delay for *SEGA^{SRC}*, while the average longest-net delay for *ROAD-HOP^{SRC}* is 52% less than that of *SEGA^{SRC}*.

Table 4 shows the results when hop from I/O pins is used. Compared to *SEGA$^{I/O}$*, *ROAD-HOP$^{I/O}$* routes the circuits by using 7.8% fewer tracks and 8.5% fewer segments. Since *SEGA$^{I/O}$* crashed while producing its routing results to calculate net delay, we couldn't compare the circuit delays directly. However, because of fewer segments, we can expect that *ROAD-HOP$^{I/O}$*'s circuit delay also will be significantly less than *SEGA$^{I/O}$*'s circuit delay. On the other hand, because of crashing we are not sure that *SEGA$^{I/O}$* results are 100% valid. We probably need to keep this in mind when making runtime comparisons between *ROAD-HOP$^{I/O}$* and *SEGA$^{I/O}$*; *ROAD-HOP$^{I/O}$* is 2.5 times slower than *SEGA$^{I/O}$*.

6. CONCLUSIONS

We presented a new approach called *ROAD-HOP* for detailed routing in FPGAs that uses the bump-and-refit ($B\&R$) paradigm and judicious creation of hops to minimize the number of tracks as well as to reduce circuit delay. By using B&R, it overcomes the well-known net ordering problem of detailed routing and obtains routings with minimum number of tracks, and by using the net splitting and path selection techniques we describe in Sec. 4., it eliminates unnecessary hops and increases circuit performance. *ROAD-HOP* can also be extended to perform similar routing in FPGAs with complex i-to-j switchboxes. We believe *ROAD-HOP* represents an important advance in detailed routing technology for FPGAs.

References

[1] S. Dutt, V. Verma and H. Arslan, "A Search-Based Bump-and-Refit Approach to Incremental Routing for ECO Applications in FPGAs", *TODAES, ACM (TODAES)*, 7(4), pp. 664-693, 2002.

[2] S. Dutt, V. Shanmugavel and S. Trimberger, "Efficient Incremental Rerouting for Fault Reconf. in FPGAs",*ICCAD*, pp. 173-176, 1999.

[3] E. Rosenberg, "New Iterative Supply/Demand Router with Rip-up and Reroute Strategy",*DAC 1987*.

[4] H. Arslan and S. Dutt, "ROAD:An Order-Impervious Complete Detailed Router for FPGAs",*ICCD 2003 pp.350-356*.

[5] G. Lemieux and S. Brown,"A Detailed Router for Allocating Wire Segments in FPGAs", *ACM/SIGDA Physical Design Workshop, pp.215-226, 1993*

[6] S. Brown, J. Rose, Z.G. Vranesic, "A Detailed Router for FPGAs", *IEEE Transactions on Computer Aided Design, 11(5), pp.620-628,1992*

[7] V. Betz and J. Rose,"VPR:A New Packing, placement and routing Tool for FPGA Research",*Int. Wrkshp. of FPGAs*,London,1997.

[8] W.C.Elmore: "The Transient Response of Damped Linear Network with Particular Regard to Wideband Amplifiers",*J. Applied Physics*, 1948, 19, pp.55–63.