A New FPGA Detailed Routing Approach Via Search-Based Boolean Satisfiability

Gi-Joon Nam, *Member, IEEE*, Karem A. Sakallah, *Fellow, IEEE*, and Rob A. Rutenbar, *Fellow, IEEE*

***Abstract*—Boolean-based routing methods transform the geometric FPGA routing task into a large but atomic Boolean function with the property that any assignment of input variables that satisfies the function specifies a valid routing solution. Previous attempts at FPGA routing using Boolean methods were based on binary decision diagrams which limited their scopes, because of size limitations, to only individual FPGA channels. Thus, FPGA layouts were decomposed into channel-wise slices that are handled separately, and those individual channel slices were stitched together later. In this paper, we present a new search-based satisfiability (SAT) FPGA detailed routing formulation that handles all channels in an FPGA simultaneously. The formulation has the virtue that it considers all nets concurrently allowing higher degrees of freedom for each net, in contrast to the classical one-net-at-a-time approaches and is able to prove the unroutability of a given circuit by demonstrating the absence of a satisfying assignment to the routing Boolean function. To demonstrate the effectiveness of this method, we first present comparative experimental results between integer linear programming (ILP)-based routing, which is an alternative concurrent method, and SAT-based routing. We also present the first comparisons of search-based Boolean SAT routing results to other conventional routers and offer the first evidence that SAT methods can actually demonstrate the unroutability of a layout. Preliminary experimental results suggest that our approach compares very favorably with both the ILP-based approach and conventional FPGA routers.**

***Index Terms*—Boolean satisfiability, field programmable gate arrays, routing.**

I. Introduction

FIELD PROGRAMMABLE gate arrays (FPGAs) already have adopted and successfully adapted a variety of ASIC layout techniques including iterative improvement placers [3] and maze-style [14] and channel-style [13] routers. However, the discrete nature of FPGA logic blocks and routing fabrics differentiate the FPGA layout problem from those of other design styles. Unlike conventional layout tools, FPGA placers and routers have less flexibility and must contend with a rigid, discrete, prefabricated set of interconnection patterns. This discrete nature of FPGAs thus admits unique layout strategies that can strive to leverage the limited palette of geometric alternatives to prune the space of viable layouts. Search-based global and detailed routers [2], [4], channel routers [12], and simultaneous placer/routers [17]–[22] are a few examples of such attacks that actually exploit the rigid limitations on FPGA layout geometries. Unfortunately, these geometric limitations still render the problem of predicting whether a given netlist can fit on a specific FPGA architecture—in particular, whether it can route successfully after placement—very difficult. Improvements in routability estimators [5], statistical estimators [8], simultaneous placer/routers [17], and routing tactics that can heuristically abandon the layout when unroutability appears inevitable [21] are all viable responses to this critical problem. Nevertheless, it remains a practical impossibility to answer exactly this simple question for most FPGA placements: is this layout routable?

Boolean-based routing is a new approach that addresses this question. We render the routing constraints as a large but atomic Boolean function which is satisfiable (has an assignment of input variables such that the generated function evaluates to constant "1") if and only if the layout is routable. In other words, any satisfying assignment to the variables of the routing Boolean function represents a legal routing solution. Moreover, by demonstrating that there is no satisfying assignment for a generated routing Boolean function, we can prove that no layout solution exists, which is very rare in other layout approaches. A particular virtue of this formulation is that much of the geometric complexity of the interaction among nets is hidden and rendered implicitly in the Boolean constraint functions so that all paths for all nets are considered simultaneously. FPGAs are a natural first target for Boolean-based routing since their discrete geometric structure maps easily onto various Boolean formulations. In our previous approach [23], we solved the resulting Boolean problem by creating a binary decision diagram (BDD [6], [7]) to represent the routing constraint formulas. The BDD-based approach has a variety of useful properties, e.g., all possible routing solutions are captured as traversals of the BDD from its root to the distinguished "1" leaf node, and the layout is unroutable if the BDD degenerates to the function that is identically "0." Unfortunately, BDDs are difficult to construct for large routing problems. Variable orderings are difficult to derive, and the BDD graph itself can become unmanageably large during intermediate computations. To overcome this, we decomposed our FPGA layouts into channelwise slices, used BDD-based routing only on each vertical channel and introduced additional routing constraints on entrance/exit points for each net in the considered channel.

Manuscript received January 24, 2001; revised October 23, 2001. This research was supported in part by the National Science Foundation under Grant 9404632. This paper was recommended by Associate Editor M. Pedram.

G.-J. Nam is with the IBM Austin Research Laboratory, Austin, TX 78758 USA (e-mail: gnam@us.ibm.com).

K. A. Sakallah is with the University of Michigan, Electrical Engineering and Computer Science Department, Ann Arbor, MI 48109 USA (e-mail: karem@eecs.umich.edu).

R. A. Rutenbar is with Carnegie Mellon University, Electrical and Computer Engineering Department, Pittsburgh, PA 15213 USA (e-mail: rutenbar@ece.cmu.edu).

Publisher Item Identifier S 0278-0070(02)04701-2.

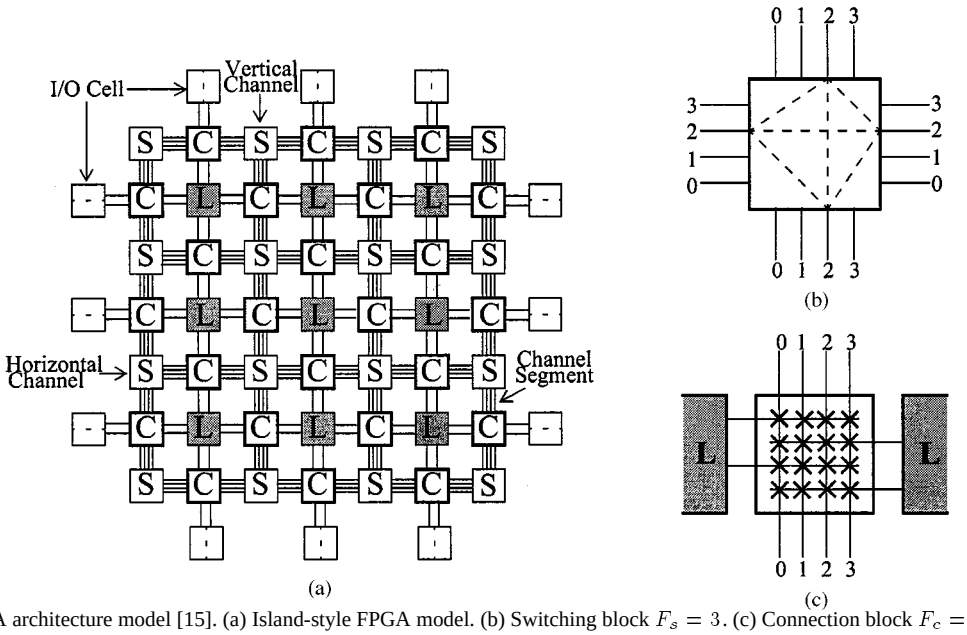


Fig. 1. Assumed FPGA architecture model [15]. (a) Island-style FPGA model. (b) Switching block $F_s = 3$. (c) Connection block $F_c = 4$.

Obviously, a more desirable approach would be to model the entire FPGA routing problem by a single Boolean function, but this appears to be impractical for large designs using BDDs as the solution engine. A BDD-based approach essentially "solves" the function yielding all satisfying assignments. If we are merely interested in finding a single satisfying assignment we can instead employ an implicit systematic search in the n-dimensional Boolean space of the input variables to locate such an assignment, or to prove that it does not exist. Such search-based approaches are commonly referred to as satisfiability (SAT) algorithms.

In this paper, we revisit the Boolean-based routing ideas from [23] and propose a new FPGA detailed routing formulation which enables us to not only allow more flexibility for each net, but also to complete more complex FPGA routing tasks. In particular, we show the first results from routing entire FPGAs—all nets embedded simultaneously—using a new SAT-based routing formulation. An alternative to solving the routing constraints using Boolean satisfiability is to use integer linear programming (ILP) techniques, which is another concurrent layout method. Thus, we present extensive comparative experimental results between ILP-based and SAT-based FPGA routing to demonstrate the effectiveness of out approach. Then, we show the first comparisons of SAT-based routing quality to other published FPGA routers, and, to the best of our knowledge, the first conclusions of unroutability for a placement given a specified global routing and track counts per channel.

The rest of paper is organized as follows. Section II reviews the typical island-style FPGA model that we employ and key terminologies are explained. The constraint types in FPGA detailed routing and how to represent them in Boolean SAT functions as well as ILP forms are illustrated in Section III. The overall flow of the approach is also provided there. Section IV presents extensive experimental results showing the comparative performance of SAT-based FPGA detailed routing method on standard benchmarks. Finally some concluding remarks are given in Section V.

II. Target FPGA Architecture Model and Terminology

We based our modeling on a standard island-style FPGA architecture (e.g., Xilinx 4000 type [26]). This is one of the most commonly used layout models in FPGA applications and is depicted in Fig. 1. An island-style FPGA is comprised of a two-dimensional array of configurable logic blocks (CLBs), connection blocks (C-blocks) and switching blocks (S-blocks). Each CLB (marked "L" in Fig. 1) contains the combinational and sequential logic that implements the functionality of a circuit. C- and S-blocks contain programmable switches and form the routing resources. C-blocks connect CLB pins to tracks (wire segments) in the adjoining channels. S-blocks are surrounded by C-blocks and allow signals to either pass through or make 90° turns. Personalization of the routing resources is achieved through proper programming of the routing switches. Programmable IO cells reside on the boundary of the array.

We assume that every wire is fully segmented, meaning that each wire segment spans only one block size. This is not the limitation of our formulation method. It is always possible to extend the formulation to consider different lengths of wire segments within a channel. However, we employed only fully segmented wires within a channel to facilitate the comparison with other routing results in Section IV. A set of wire segments on the same row/column in horizontal/vertical channels forms a signal track. The routing capacity of a given FPGA architecture is conveniently expressed by three parameters [5]; the channel width W is the number of tracks in a vertical or horizontal channel; the C-block flexibility F_c is the number of tracks in adjacent channels that each CLB logic pin may connect to; and the S-block flexibility F_s is the number of other tracks that each wire segment entering an S-block can connect to. In the sequel, we assume that all channels (vertical and horizontal) have the same number of tracks. In Fig. 1(b), each wire segment entering this S-block can connect to one track on each of the other three sides; hence, $F_s = 3$. In Fig. 1(c), each logic pin can be connected up to any of the four tracks in the C-block; thus, $F_c = W = 4$.

A net is a set of CLB pins that must be electrically connected; nets are decomposed into sets of two-pin connections which represent source/sink pairs. A two-pin connection is further decomposed into one or more horizontal and/or vertical net-segments, each of which is an alternating sequence of C- and S-blocks within the channel that forms an uninterrupted straight path. The decomposition of a multipin net into a set of two-pin connections is necessary to find higher quality of routing solutions as will be demonstrated in Section III. However, for a better runtime, it is possible to formulate the routing problem without decomposition process. In other words, the routing formulation is flexible enough to consider both two-pin connection-based nets as well as multipin nets. A detailed route of a net is a set of wire segments and routing switches, within the restricted routing area set by the global router. For each net-segment, a detailed router assigns wire segments and routing switches following the topology specified by the global router such that no overlap among detailed routes of different nets occurs. Our focus of SAT routing is a detailed routing task in this scenario, i.e., given a placement and a global routing, find a legal detailed routing solution or demonstrate that no such detailed routing exists.

III. FPGA Detailed Routing Formulation

In this section, we first describe a set of constraint types which explicitly captures the requirements for complete FPGA detailed routing solutions and then show how to represent those constraints in conjunctive normal form (CNF) satisfiability problems and ILP form.

In the simplest form, FPGA detailed routing problems can be transformed into a net-to-track assignment problem. Each net segment in the layout is represented by a track variable that indicates the index of the horizontal or vertical track over which the net segment might be routed. A routing constraint function is then defined over these variables. There is only a finite number of routing resources in an FPGA and each routing resource is discrete and rigid. Because of these unique features of FPGAs, it turns out that the following two types of routing constraints are necessary and sufficient to guarantee a legal FPGA detailed routing solution.

- **Connectivity constraints** to ensure the existence of a conductive path for each two-pin connection through the sequence of C- and S-blocks specified by the global router. These constraints basically model the routing flexibility available in the C- and S-blocks. It also makes sure that each two-pin connection is assigned only to valid routing tracks within the routing channels. One connectivity constraint function is constructed per two-pin connection of a net.
- **Exclusivity constraints** to guarantee that electrically distinct nets with overlapping vertical or horizontal spans in the same channel are assigned to different tracks. These constraints are essentially instances of channel routing problems and one exclusivity constraint function is formed per horizontal/vertical channel.

An example that illustrates the construction of FPGA routing constraints is shown in Fig. 2 for an FPGA with $W = F_c = F_s = 3$. Each net is assumed to have been assigned a global route (a sequence of C- and S-blocks) by a global router [Fig. 2(a)]. Track variables are then created for each net segment to model its possible assignment to specific tracks in each of the channels specified by its global route. For instance, in this example two track variables are associated with net A: AH to indicate its track assignment in horizontal channel 1 and AV to indicate its track assignment in vertical channel 1. Each track variable is multivalued with a domain $\{0, \ldots, W-1\}$.

The construction of the connectivity constraint of net A is depicted in Fig. 2(b). The connectivity constraint for a given net restricts the net's track variables to those values that ensure a continuous conductive path between the net's pins. For example, net A can be assigned to any track in horizontal channel 1 as well as vertical channel 1 as long as the same track number is used in both channels. This reflects the connectivity requirement at C-block(4,1), S-block(1,1), and C-block(1,2) whose flexibilities are $F_c = W$ and $F_s = 3$, respectively [see Fig. 2(b)]. In a similar way, the connectivity constraints of net B and C can be formulated.

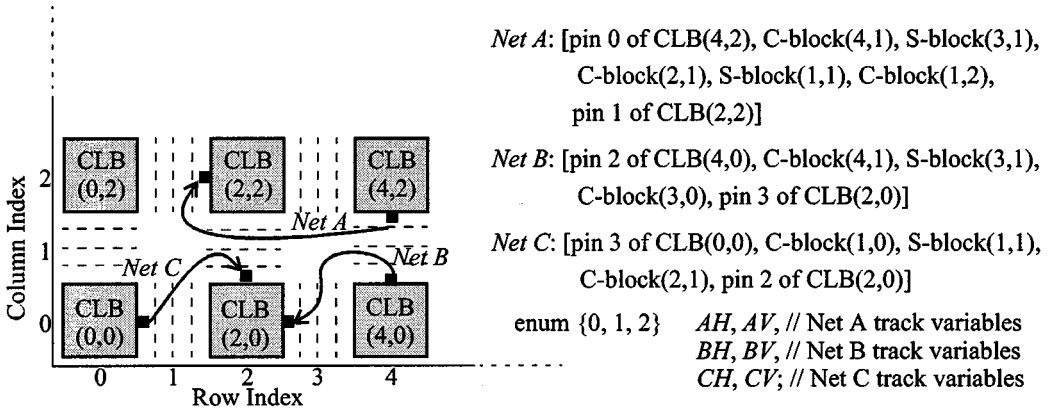
The exclusivity constraints in this example ensure that nets A and B as well as A and C are assigned to different track numbers in horizontal channel 1. In general, whenever two different net segments have overlapping spans in any routing channel, a single inequality constraint function (for example, $AH \neq BH$) is enforced. The exclusivity constraint function of a horizontal/vertical routing channel is the conjunction (logical "and") of a set of such inequality constraint functions occurring in the channel. Fig. 2(c) shows the actual exclusivity constraint function in horizontal channel 1 for the example FPGA.

The routing constraint Boolean function that models the routability of all the nets is now simply the conjunction of all the connectivity and exclusivity requirements. For the current example, the routability constraint Boolean function of these three nets is

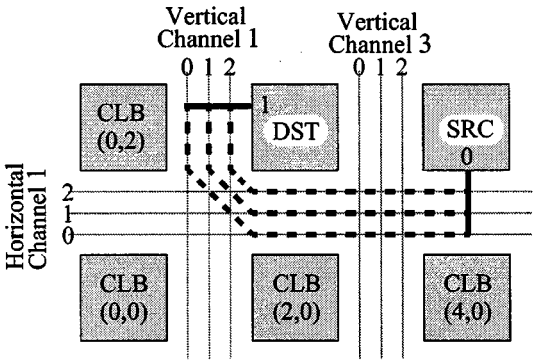
$$\text{Routable}(X) = \text{Conn}(A) \wedge \text{Conn}(B) \wedge \text{Conn}(C) \wedge \text{Excl}(H1)$$

where X is a vector of track variables: AH, AV, BH, BV, CH, and CV. Assuming that a circuit has total n two-pin connections and each global route of a two-pin connection passes through m channels on average, (i.e., m net segments per two-pin connection), the number of required Boolean variables of the routability function is approximately $n \cdot \lceil \log_2 W \rceil \cdot m$ in this formulation.

Another issue worth mentioning in Boolean-based FPGA routing is pin doglegs. Pin doglegs allow us to route a net using more than one egressing track per CLB logic pin. Betz [2] and [28] point out that in commercial FPGAs such as the Xilinx XC4000-series and Lucent ORCA FPGAs [16], the input logic pin of a CLB is connected to routing wire segments via a multiplexer rather than a set of independent pass transistors so that input pin doglegs are not possible while output pins in a CLB are allowed to be doglegged (see Fig. 3). Doglegging can reduce the required number of tracks per channel in an FPGA, but it was not well researched in the literature thus far. Fig. 4(a) and (b) illustrates how pin doglegs contribute to the reduction



(a)



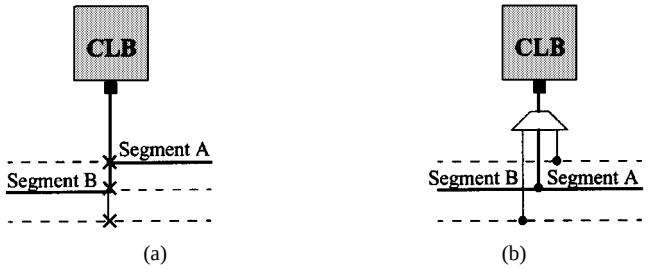
$$Conn(A) = [(AH \equiv 0) \vee (AH \equiv 1) \vee (AH \equiv 2)] \wedge \quad \text{// Horizontal channel 1}$$
$$[(AH = AV)] \wedge \quad \text{// S-block(1,1)}$$
$$[(AV \equiv 0) \vee (AV \equiv 1) \vee (AV \equiv 2)] \quad \text{// Vertical channel 1}$$

(b)

$$Excl(H1) = (AH \neq BH) \wedge (AH \neq CH)$$

(c)

Fig. 2. Generation of connectivity and exclusivity constraints. (a) Global routing configuration for nets A, B, and C and corresponding variable declarations. (b) Possible detailed routes of net A and the corresponding connectivity constraint. (c) Exclusivity constraint.

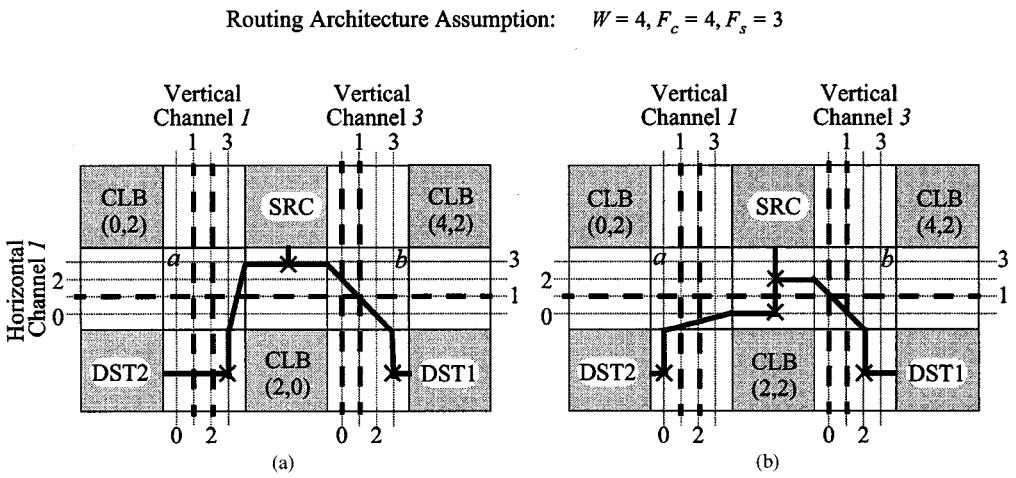


(a) (b)

Fig. 3. Pin connection routing architecture at CLB. (a) Output pin connection. (b) Input pin connection.

in the required number of tracks. Net N consists of a source pin in SRC CLB and two sink pins in DST1 and DST2 CLBs. Dotted lines indicate those tracks that are already used by other nets in corresponding channels so that net N is not allowed to use them. Fig. 4(a) shows the example where source pin is not allowed to be doglegged, whereas Fig. 4(b) is the case where doglegs are allowed. In the example Fig. 4(a), track 2 cannot be used in vertical channel 3 because track 2 in vertical channel 1 is already used by another net. Note that every net segment connected must use the same track number due to the symmetrical S-block architecture assumption. By utilizing the doglegging at the source pin, the net N was able to be routed with one less track per channel in the example Fig. 4(b).

In general, for the purpose of taking full advantage of doglegging at logic pins of CLBs, multipin nets should be decomposed into multiple two-pin connections, and each two-pin connection must be assigned a separate set of constraint track variables. As shown in Fig. 4(b), we first decompose the net N into two distinct two-pin connections, $NC1$ and $NC2$, then construct one connectivity constraint function for each of $NC1$ and $NC2$ with separate constraint track variables as shown in Fig. 4(c). This method increases the number of constraint track variables needed to model a given net—in this example from 3 to 4—but enables two different two-pin connections from the same net to utilize the option of using different tracks even in the same



Two-pin connection 1: NC1 = { *NC1H, NC1V* }

Two-pin connection 2: NC2 = { *NC2H, NC2V* }

$$Conn(NC1) = [(NC1H \equiv 0) \vee (NC1H \equiv 1) \vee (NC1H \equiv 2)] \wedge [NC1H = NC1V] \wedge [(NC1V \equiv 0) \vee (NC1V \equiv 1) \vee (NC1V \equiv 2)]$$

// Horizontal channel 1
// S-block(3,1)
// Vertical channel 3

$$Conn(NC2) = [(NC2H \equiv 0) \vee (NC2H \equiv 1) \vee (NC2H \equiv 2)] \wedge [NC2H = NC2V] \wedge [(NC2V \equiv 0) \vee (NC2V \equiv 1) \vee (NC2V \equiv 2)]$$

// Horizontal channel 1
// S-block(1,1)
// Vertical channel 1

(c)

Fig. 4. Example illustrating how a dogleg reduces the track counts in channels and corresponding routing constraint formulation. (a) When a dogleg is NOT allowed at source pin. (b) When a dogleg is allowed at source pin. (c) Connectivity constraint functions which handle pin doglegging.

channels. From our experiments, this technique turns out to be critical for reducing the number of tracks required per channel.

A. Routing Constraint Representation in CNF

In this section, we describe the transformation of these Boolean routing constraint functions derived above into CNF. Every routing constraint can be expressed in CNF clauses through simple Boolean function manipulation. Fig. 5 demonstrates how to generate CNF Boolean SAT clauses from FPGA routing constraints. We assume the same routing architectural parameters used in Fig. 2; $W = 3$, $F_c = W$, and $F_s = 3$.

With a channel width of three tracks, two Boolean variables are assigned to each track variable X and Y to encode track numbers 0, 1, and 2 [Fig. 5(a)]. The connectivity constraint has two different forms depending on the block type: a C-block connectivity constraint enumerates the tracks available for routing in a given channel, whereas an S-block connectivity constraint forces the track numbers of incoming and outgoing segments to be equal for $F_s = 3$. The CNF representation of each connectivity constraint is illustrated in Fig. 5(b) and (c). The number of CNF clauses required to express a C-block connectivity constraint is dependent on the actual value of W and hard to formulate simply in one equation. For most cases, a C-block connectivity constraint requires fewer than three CNF clauses, each having at most $2 \cdot \lceil \log_2 W \rceil$ literals. The S-block connectivity constraint whose form is an equality between two track variables, however, can be expressed with exactly $2 \cdot \lceil \log_2 W \rceil$ 2-literal CNF clauses. Exclusivity constraints, which are basically inequalities between track variables, can be represented with W CNF clauses, each having $2 \cdot \lceil \log_2 W \rceil$ literals [Fig. 5(d)].

If different routing architecture is assumed (i.e., different F_c, F_s, and W values), slightly different Boolean functions for connectivity constraints can be built and plugged in. In other words, all those practical implementations of FPGA routing architecture can be hidden and handled through simple Boolean functional manipulations. Once we know the routing architectural parameters of the target FPGA, we can build a template table for a variety of routing constraint Boolean functions *a priori* by optimizing and expressing each routing constraint function in CNF clauses space. Thus, whenever a new routing constraint function is required, we can just look up the table and obtain the corresponding set of CNF clauses on the fly avoiding tedious Boolean function manipulation during the formulation process. In this way, we can more efficiently transform routing constraints into a Boolean SAT problem instance.

B. ILP Modeling of Routing Constraints

An alternative to solving the routing constraints using Boolean satisfiability is to use ILP techniques. Expressing the routing constraints for use by an ILP solver is straightforward as illustrated by Fig. 6 assuming $W = 3$, $F_c = W$, and $F_s = 3$. Connectivity and exclusivity constraints are naturally expressed in the integer domain. Thus, unlike the Boolean SAT representation, track variables X and Y do not require

$$X = [X_0, X_1] \equiv 2X_1 + X_0$$
$$Y = [Y_0, Y_1] \equiv 2Y_1 + Y_0$$
(a)

$$\begin{aligned} f(X) &= (X \equiv 0) \vee (X \equiv 1) \vee (X \equiv 2) \\ &= \overline{(X \equiv 3)} \\ &= \overline{(X_0 \wedge X_1)} \\ &= (\overline{X_0} \vee \overline{X_1}) \end{aligned}$$
(b)

$$\begin{aligned} f(X, Y) &= (X \equiv Y) \\ &= [X_0 \equiv Y_0] \wedge [X_1 \equiv Y_1] \\ &= [(X_0 \rightarrow Y_0) \wedge (Y_0 \rightarrow X_0)] \wedge [(X_1 \rightarrow Y_1) \wedge (Y_1 \rightarrow X_1)] \\ &= [(\overline{X_0} \vee Y_0) \wedge (X_0 \wedge \overline{Y_0})] \wedge [(\overline{X_1} \vee Y_1) \wedge (X_1 \wedge \overline{Y_1})] \end{aligned}$$
(c)

$$\begin{aligned} f(X, Y) &= (X \neq Y) \\ &= \overline{(X \equiv 0 \wedge Y \equiv 0) \vee (X \equiv 1 \wedge Y \equiv 1) \vee (X \equiv 2 \wedge Y \equiv 2)} \\ &= \overline{(X \equiv 0 \wedge Y \equiv 0)} \wedge \overline{(X \equiv 1 \wedge Y \equiv 1)} \wedge \overline{(X \equiv 2 \wedge Y \equiv 2)} \\ &= (X_0 \vee X_1 \vee Y_0 \vee Y_1) \wedge (\overline{X_0} \vee X_1 \vee \overline{Y_0} \vee Y_1) \wedge (X_0 \vee \overline{X_1} \vee Y_0 \vee \overline{Y_1}) \end{aligned}$$
(d)

Fig. 5. Routing constraint representation in CNF. (a) Track variable X, Y and corresponding Boolean variables. (b) Connectivity constraint at C-block in CNF. (c) Connectivity constraint at S-block in CNF. (d) Exclusivity constraint in CNF.

$$\begin{aligned} f(X) &= (X \equiv 0) \vee (X \equiv 1) \vee (X \equiv 2) \\ &= [0 \leq X \leq 2] \end{aligned}$$
(a)

$$\begin{aligned} f(X, Y) &= (X \equiv Y) \\ &= [(X - Y) \geq 0] \wedge [(Y - X) \geq 0] \end{aligned}$$
(b)

$$\begin{aligned} f(X, Y) &= (X \neq Y) \\ &= [|X - Y| \geq 1] \\ &= [(X - Y) \geq 1] \vee [(Y - X) \geq 1] \\ &= [c \cdot Z + (X - Y) \geq 1] \wedge [c \cdot (1 - Z) + (Y - X) \geq 1] \wedge [0 \leq Z \leq 1] \end{aligned}$$
(c)

Fig. 6. Routing constraint representation for use by an ILP solver. (a) Connectivity constraint at C-block. (b) Connectivity constraint at S-block. (c) Exclusivity constraint. Note that c is an arbitrary constant larger than $W - 1$.

any encoding scheme. Furthermore, connectivity constraints can be simply expressed as linear inequalities over these integer variables [see Fig. 6(a) and (b)]. Exclusivity constraints, however, are harder to render as inequalities. Specifically, a binary variable must be introduced for every pair of track variables and a transformation like that shown in Fig. 6(c) must be applied. The large number of required binary variables causes a significant increase in the size of the ILP problem, which significantly degrades the performance of the search for a solution (see Section IV). We should note that for our FPGA detailed routing problem, the goal is simply to find out whether there exists any routing solution or not. Thus, the objective function of ILPs can be safely omitted.

C. Boolean-Based FPGA Detailed Routing: Overall Flow

Our attempt at addressing the FPGA detailed routing problem led to the development of **s**atisfiability-based **d**etailed **r**outer (SDR). SDR casts an FPGA detailed routing problem as a CNF satisfiability problem that can be input to a Boolean SAT solver. The overall flow diagram of SDR is shown in Fig. 7. In the diagram, a rectangle indicates a procedure and an oval denotes an object generated by the preceding procedure. To allow for pin doglegging, the input netlist is assumed to resented in terms of two-pin connections.

1) **Global Routing**: Given a placement, we invoke any global router to assign each two-pin connection to a sequence of routing regions consisting of C- and S-blocks. The global router does not choose or fix any specific detailed routing resources.
2) **Net Distribution**: This procedure divides each net into several horizontal and vertical net segments, then sorts them in each channel. After this procedure, we have a set of net segments per channel, regardless of the channel type (horizontal or vertical), and a single track variable (i.e., a set of encoding Boolean variables) is assigned to each net segment.
3) **Track Count Estimation**: If the target FPGA architecture is not provided by the user, we determine a channel width W by applying a *left-edge channel routing algorithm* [13]. Assuming each channel is fully segmented, the left-edge algorithm produces the lower bound $W_{\min}$ on the number of tracks needed to route a given circuit, and we set $W = W_{\min}$.
4) **Constraint Generation**: Connectivity and exclusivity constraint Boolean functions are generated, as described in Section III, to yield the routing constraint Boolean function $Routable(X)$.
5) **Constraint Evaluation**: The Boolean SAT solver is invoked to find a satisfying assignment for $\mathrm{Routable}(X)$ or to prove that $\mathrm{Routable}(X)$ is unsatisfiable.
6) **Solution Interpretation**: In the case that $\mathrm{Routable}(X)$ is satisfiable, the solution from the Boolean SAT solver is an assignment of binary values ("1" or "0") to the Boolean variables which encode track variables. This information is transformed into an assignment of actual routing resources (wiring tracks and corresponding routing switches) to nets which forms the actual FPGA routing solution. If $\mathrm{Routable}(X)$ is not satisfiable, then no legal detailed routing solution exists with the given placement and global routing topology.

IV. Comparative Experiments and Analyses

The above SAT-based FPGA detailed routing approach can be employed in two different experimental scenarios.

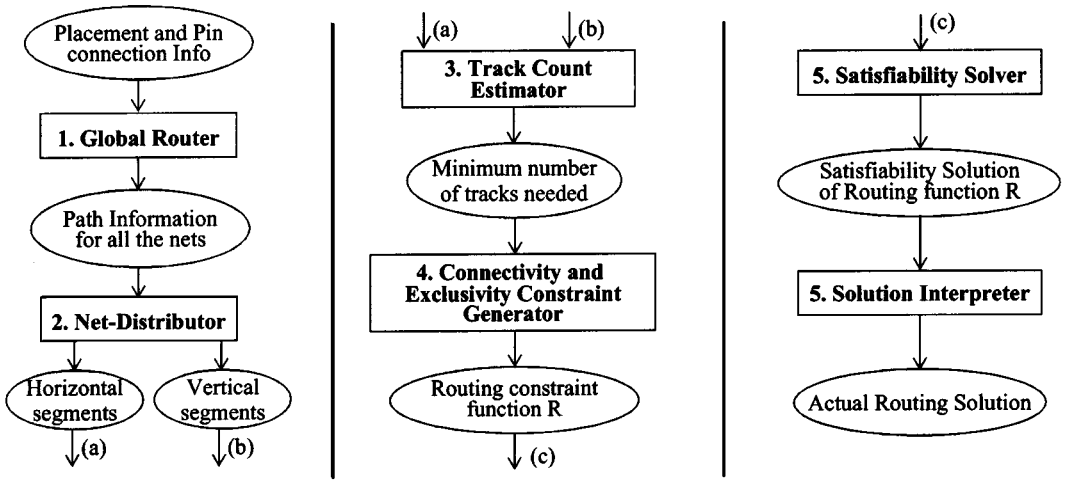


Fig. 7. Overall flow diagram.

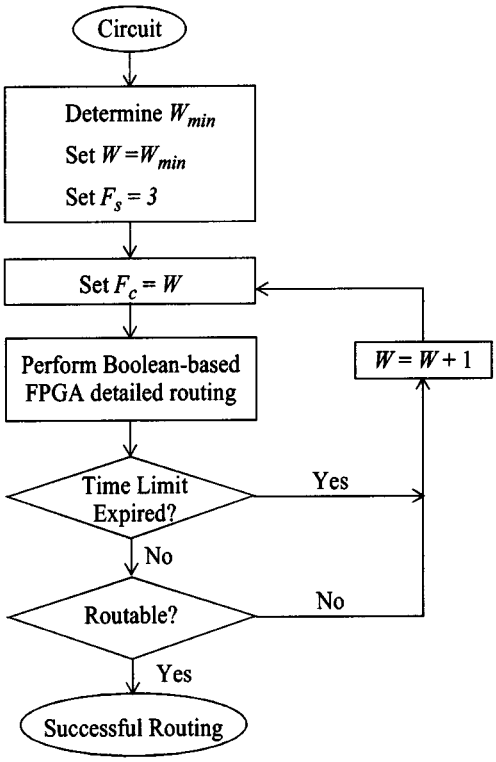


Fig. 8. Experimental testbench.

- When a placed and globally routed circuit as well as a target FPGA architecture are given, this method is able to determine whether the circuit is routable or not.
- When only a placed and globally routed circuit is given, it can find the "smallest" FPGA routing architecture, i.e., the architecture that has the minimum values of the W, F_c, and F_s parameters, to render the given circuit routable.

Fig. 8 shows the experimental plan we employed. We assume that no target FPGA is provided, but set the S-block flexibility at the minimum value $F_s = 3$ and make the C-block flexibility equal to the number of available tracks in each channel, i.e., set $F_c = W$. These settings reflect the Xilinx XC4000-series architecture model [26]. Following the second scenario, the primary goal of the experiments was to find the minimum value of W which renders a given circuit routable. This was done by attempting the routing for increasing values W of starting from the lower bound $W_{\min}$ found by the left-edge channel routing algorithm. The track width is incremented when a Boolean SAT solver determines that the routability function $\mathrm{Routable}(X)$ is unsatisfiable or when it is aborted after a preset runtime limit (10 h). The procedure terminates when a Boolean SAT solver concludes that $\mathrm{Routable}(X)$ is satisfiable and returns the corresponding track count W and detailed net-to-track assignments of the satisfying solution.

We experimentally tested the effectiveness of the SAT-based routing formulation on the standard MCNC benchmark circuits downloadable from [27]. The relevant properties of these circuits, listed in Table I, include the number of multipin nets (column "#Nets"), the corresponding number of two-pin connections that will be routed individually (column "#2pin Conns"), the average number of sink pins per net (column "Ave. Sinks/Net"), the average number of channel segments per 2pin connection (column "Ave. ChanSegs/2pin Conn"), the size of the target FPGA CLB array (column "CLB Dim."), the actual number of CLBs used by a circuit (column "#CLB used"), and the CLB utilization (column "%CLB util"). The average number of channel segments per two-pin connection is important because the number of Boolean variables required to formulate routing problems is dependent on this factor. The utilization of CLBs is provided here because it indicates how densely the circuit is packed on the target FPGA. The benchmark circuits range in size from 70 CLBs with 79 multipin nets for 9symml to 358 CLBs with 404 multipin nets for k2. On average they have about 500 two-pin connections with an 80% utilization of available CLBs.

A. Comparative Experiment: Boolean SAT Routing Versus ILP Routing

As described in Sections III-A and III-B, FPGA detailed routing problems can be transformed into either Boolean SAT or ILP instances. In this section, we present comparative

TABLE I
BENCHMARK CIRCUIT CHARACTERISTICS

Circuits	#Nets	#2pin Conns	Ave. Sinks/Net	Ave. ChanSegs/ 2pin Conn.	CLB Dim.	#CLB Used	%CLB Util.
9symml	79	259	3.28	10.99	9 x 9	70	86
alu2	153	510	3.33	8.46	12 x 12	143	99
apex7	126	300	2.38	5.25	11 x 11	77	64
C499	115	312	2.71	6.00	10 x 10	74	74
C880	234	656	2.80	6.59	14 x 14	174	89
C1355	115	312	2.71	6.31	10 x 10	74	74
example2	205	444	2.17	5.86	19 x 19	120	33
k2	404	1257	3.11	8.15	19 x 19	358	99
term1	88	202	2.30	4.24	8 x 8	54	84
too_large	186	519	2.79	7.12	13 x 13	148	88
vda	225	722	3.21	8.26	15 x 15	208	92
Average	176	499	*	*	*	*	80

TABLE II
COMPARISON BETWEEN BOOLEAN SAT- AND ILP-BASED ROUTING

	SDR				ILP-based Routing			
Circuits	#Vars	#Clauses	CPU (secs)	W	#Vars	#Constraints	CPU (secs)	W
9symml	2604	32450	1.94	6	6277	10824	2845.43	8
alu2	3882	94088	26.52	8	12467	22346	6846.13	16
apex7	1893	15358	0.39	5	3035	4748	270.70	5
C499	2070	22470	460.76	6	3942	6504	699.09	11
C880	4623	72021	5718.00	7	10849	18620	3992.61	12
C1355	2178	23706	31326.00	6	4142	6832	469.28	11
example2	3603	41023	1.91	6	7080	11760	2134.29	8
k2	13176	445254	130.04	12	38765	70944	N.C	N.C
term1	746	3964	0.38	4	1189	1640	382.21	4
too_large	3972	60432	1001.60	7	9077	15508	3214.57	11
vda	7436	168604	98.14	10	17438	31158	30004.65	18
Total ΣW	*	*	*	**77**	*	*	*	**104**

experimental results for these two approaches. We believe this to be the first such comparison between two concurrent routing approaches. Using the placement and global routing solutions generated by VPR [2], routing constraint functions based on Boolean SAT and ILP were produced to determine the minimum number of tracks required to route the circuit. These routability functions were subsequently evaluated by the GRASP SAT solver [20] and CPLEX version 6.5.2 [29] which is the most advanced and popular ILP solver. The experiment was conducted on a SUN Ultra 5 Sparc running SunOS with 512 M of physical memory.

Table II summarizes the results of this experiment. For each of the Boolean SAT and ILP routing methods, the table shows the number of corresponding variables (either Boolean or integer), the number of constraints (CNF clauses or integer linear constraints), the solution search CPU time in seconds and the minimum routable channel widths found within the runtime limit (10 h). The SAT search CPU time records only the best runtime among multiple runs with different variable ordering heuristics. These data clearly suggest the superiority of the Boolean SAT approach over the ILP approach. For just two cases (apex7 and term1), the ILP method was able to route the benchmark circuits with the same number of tracks as the Boolean SAT method. For the rest of the circuits, the ILP approach required at least two more routing tracks per channel than SDR. Even worse, for alu2 and vda, the solutions found by the ILP approach are al-

TABLE III
TRACK NUMBER COMPARISON AMONG DIFFERENT FPGA ROUTING ALGORITHMS. THE NUMBER OF TRACKS FOR OTHER ROUTERS THAN SAT-BASED APPROACH WERE CITED FROM [2] AND [15] FOR REFERENCE

	Placer				SPLACE	Altor [19]		
	G. Router	VPR [2]			SROUTE [24]	L.Route [18]	OGC [25]	FPR [1]
	D.Router	SDR	SEGA[15]	VPR		CGE [4]		
Circuits	9symml	6	6	5	7	9	9	9
	alu2	8	9	6	8	12	9	10
	apex7	5	6	4	6	13	10	9
	example2	6	6	6	7	18	12	13
	k2	12	11	9	11	19	16	17
	term1	4	6	4	5	10	9	8
	too_large	7	9	6	8	13	11	11
	vda	10	10	8	10	14	11	13
	Total ΣW	58	63	48	62	108	87	90

most twice the number of tracks required by SDR, and for the benchmark k2 circuit, the ILP method failed to find any routing solution. From the table, we can also observe that the quality gap between these two methods grows with circuit size. With respect to runtime, SDR is also generally faster than CPLEX. For every case, SDR was able to find a better routing solution in less CPU time. For only two cases, C880 and C1355, it took more than 1 h for SDR to find a routing solution. However, when we targeted a routing architecture with one more track per channel (i.e., $W = 8$ for C880 and $W = 7$ for C1355), SDR took only 1170 and 10 CPU seconds to find a routing solution, respectively. Overall, we can conclude that the Boolean SAT-based routing approach is more efficient than the ILP approach for FPGA detailed routing.

B. Performance Comparison With Conventional Routers

Table III presents the performance comparison between the Boolean SAT router and other conventional one-net-at-a-time routers on the same FPGA detailed routing problems. The table shows the number of tracks required to successfully route each benchmark circuit using the indicated combination of programs for placement, global routing, and detailed routing. The most meaningful comparison in this table is between SDR, SEGA [15], and VPR [2] because they differ only in how detailed routing was done. The results of using other placers and routers are provided here just for reference.

These results clearly show that SDR produces better results than SEGA, achieving routability with fewer tracks. When compared to VPR, however, SDR is slightly worse, on average requiring 1.25 more tracks per channel. More interestingly, the SAT approach indicated the unroutability of the four benchmark circuits highlighted in the table (9symml, alu2, apex7, too_large) for channel width that VPR was able to route successfully (i.e., $W = 5, 6, 4, 6$, respectively). This discrepancy was traced that VPR is able to change the current global routing configuration when it cannot find a detailed routing solution easily.

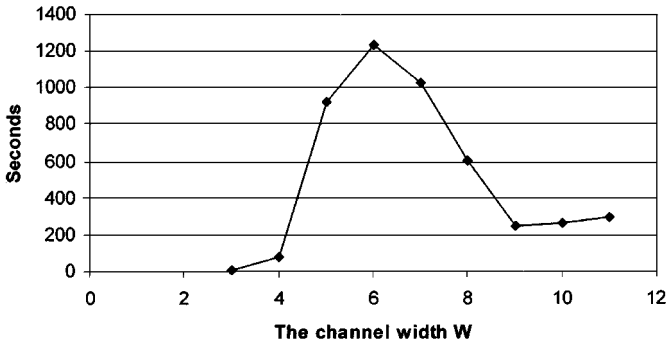


Fig. 9. *too_large* SAT solving time versus channel width.

In spite of this limitation, the performance of SDR seems to be very satisfactory as an FPGA detailed router.

Finally, Fig. 9 displays search time of the SAT solver as a function of channel width W for benchmark too-large. The circuit was proven to be unroutable with six or fewer tracks per channel and routable with seven or more tracks per channel. The figure clearly shows that the search time is longest at the marginal channel width $W = 6$ which sits at the boundary between routability and unroutability. The search time decreases when the problem becomes either underconstrained (larger W) or overconstrained (smaller W). This behavior is consistent with the intuition that as the number of tracks increases, the number of possible solutions and the likelihood of finding one increases. Conversely, with fewer and fewer tracks available for routing, the likelihood of quickly discovering that the constraints cannot be satisfied increases and unroutability can be proved easily. The range of channel widths that makes the generated routing problems difficult to prove routable or unroutable seems to be narrow, usually within two tracks per channel based on the experimental results. In other words, even though the SAT approach may fail to find a routing solution with T tracks per channel in a certain routing problem, increasing channel width to $T + 1$ or $T + 2$ tracks seems to generate easy SAT instances whose solutions can be found quickly. This observation leads

to an interesting possibility for using the SAT approach as an estimator of the difficulty of detailed routing problems. For instance, if a certain problem cannot be solved within a certain execution time, the problem is considered "difficult," suggesting the need of more routing resources (tracks) or an alternative placement/global routing.

Overall, we regard these as very satisfactory initial results for our preliminary implementation. Specifically, the search-based Boolean SAT approach seems more capable of handling complete FPGA routing tasks than the ILP-based routing attack. Also, the ability to determine unroutability—or, at least, near-unroutability, in the sense that we can find the track counts where SAT-based routing is able to demonstrate concretely the absence of any satisfying assignment—could open a new avenue of research. For example, it has long been known that "well behaved" placements have the property that a large fraction of their nets can be embedded with simple pattern routes; indeed, this is the basis for many board and IC routing strategies which begin with fast, simple pattern routes, and then move on to maze-routing only for clean-up, e.g., see [11] for an early discussion of the idea. With SAT-based routing, we can determine exactly if a layout is routable with only a limited set of patterns. The open question is whether we can infer practical routability from this. It would be extremely valuable if this sort of analysis is able to predict the onset of "difficulty" in routing, the point at which many routes must deviate from simple patterns to embed.

V. Conclusion

In this paper, we described a new strategy for SAT-based FPGA detailed routing. We showed that the routing constraints can be expressed in two different forms: CNF which is suitable for use with generic SAT solvers and a integer linear constraints which can be input to ILP solvers. Experimental evaluation of these two approaches showed that the Boolean SAT has an edge over the ILP method in speed, solution quality, and modeling flexibility. In addition, by moving from BDD-based techniques to search-based techniques, we have been able to solve much larger SAT instances, which has allowed us for the first time to formulate the entire routing problem, embedding all nets simultaneously, as a single SAT instance. Preliminary results are encouraging; we have been able to completely route a variety of FPGA benchmarks and also demonstrate unroutability for some cases. As we noted earlier, the technique at its core is a variant of pattern routing, so that our routing and proofs of unroutability are always with respect to some finite (albeit very large) set of pattern constraints for the nets. We think that search-based SAT is a viable vehicle for further work on SAT-based FPGA layout.

Acknowledgment

The authors are grateful to V. Betz and G. Lemieux of the University of Toronto for providing benchmark circuits for the experiments. They are grateful to Texas Instruments for grants of the Sun workstations used for their experiments.

References

[1] M. J. Alexander, J. P. Cohoon, J. L. Ganley, and G. Robins, "Performance-oriented placement and routing for field-programmable gate arrays," in *Proc. Eur. Design Automation Conf.*, 1995, pp. 259–264.

[2] V. Betz and J. Rose, "VPR: A new packing, placement and routing tool for FPGA research," in *Seventh Annual Workshop Field Programmable Logic and Applications*, 1997, pp. 213–222.

[3] M. A. Breuer, "A class of min-cut placement algorithms," in *Proc. Design Automation Conf.*, 1997, pp. 284–290.

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

[5] S. Brown, R. Francis, J. Rose, and Z. Vranesic, *Field Programmable Gate Arrays*. Boston, MA: Kluwer, 1992.

[6] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation," *IEEE Trans. Comput.*, pp. 677–691, 1986.

[7] ——, "Symbolic Boolean manipulation with ordered binary decision diagrams," *ACM Computing Surveys*, vol. 24, no. 3, pp. 293–318, Sept. 1992.

[8] P. K. Chan *et al.*, "On routability prediction for field programmable gate arrays," in *Proc. Design Automation Conf.*, June 1993.

[9] M. Davis and H. Putnam, "A computing procedure for quantification theory," *J. Assn. Computing Machinery*, vol. 7, pp. 201–215, 1960.

[10] Center for discrete mathematics and theoretical computer science (DIMACS) website. [Online]. Available: Available: http://DIMACS.Rutgers.EDU.

[11] J. Dion, "Fast printed circuit board routing," in *Proc. Design Automation Conf.*, 1988.

[12] J. Greene, V. Roychowdhury, S. Kaptanoglu, and A. El Gamal, "Segmented channel routing," in *Proc. Design Automation Conf.*, 1990, pp. 567–572.

[13] A. Hashimoto and J. Stevens, "Wire routing by optimizing channel assignment within large apertures," in *Proc. Design Automation Conf.*, 1971, pp. 155–169.

[14] C. Y. Lee, "An algorithm for path connections and its applications," *IRE Trans. Electron. Computers*, 1961.

[15] G. Lemieux, S. Brown, and D. Vranesic, "On two-step routing for FPGAs," in *Int. Symp. Physical Design*, 1997, pp. 60–66.

[16] Lucent Technologies, *Field-Programmable Gate Arrays Data Book*, Oct. 1996.

[17] S. K. Nag and R. A. Rutenbar, "Performance-driven simultaneous placement and routing for FPGAs," *IEEE Trans. Computer-Aided Design*, pp. 499–518, June 1998.

[18] J. Rose, "Parallel global routing for standard cells," *IEEE Trans. Computer-Aided Design*, vol. 9, pp. 1085–1095, Oct. 1990.

[19] J. Rose, W. Snelgrove, and Z. Vranesic, "ALTOR: An automatic standard cell layout program," in *Canadian Conf. Very Large Scale Integration*, Nov. 1985, pp. 169–173.

[20] J. P. M. Silva and K. A. Sakallah, "GRASP: A search algorithm for propositional satisfiability," *IEEE Trans. Computers*, vol. 48, May 1999.

[21] J. S. Swartz, V. Betz, and J. Rose, "A fast routability driven router for FPGAs," in *Int. Symp. FPGAs*, Feb. 1998.

[22] N. Togawa, M. Yanagisawa, and T. Ohtsuki, "Maple-opt: A performance-oriented simultaneous technology mapping, placement, and global routing algorithm for FPGA's," *IEEE Trans. Computer-Aided Design*, vol. 17, pp. 803–815, Sept. 1998.

[23] R. G. Wood and R. A. Rutenbar, "FPGA routing and routability estimation via Boolean satisfiability," *IEEE Trans. VLSI Syst.*, vol. 6, pp. 222–231, June 1998.

[24] S. Wilton, "Architectures and algorithms for field-programmable gate arrays with embedded memories," Ph.D. dissertation, Univ. Toronto, 1997.

[25] Y.-L Wu and M. Marek-Sadowska, "Orthogonal greedy coupling—A new optimization approach to 2-D FPGA routing," in *Proc. Design Automation Conf.*, June 1995.

[26] XILINX, *The Programmable Gate Array Data Book*. San Jose, CA: Xilinx, Inc., 1993.

[27] G. Lemieux.. SEGA FPGA Routing Tool website. [Online]. Available: Available: http://www.eecg.toronto.edu/~lemieux/sega/sega.html.

[28] V. Betz.. FPGA Place-and-Route Challenge website. [Online]. Available: Available: http://www.eecg.toronto.edu/~vaughn/challenge/challenge.html.

[29] CPLEX Optimization Tool website.. [Online]. Available: Available: http://www.ilog.com/products/cplex/.

Gi-Joon Nam (S'99–M'01) received the B.S degree in computer engineering from Seoul National University, Seoul, Korea, in 1995, and the M.S. and Ph.D degrees in computer science and engineering from the University of Michigan, Ann Arbor, MI, in 1999 and 2001, respectively.

After graduation, he joined IBM Austin Research Lab and is currently working as a research member since August 2001. His research interests include computer aided-design algorithms of high performance ASICs FPGAs, design, and analysis of algorithms and computer architecture.

Karem A. Sakallah (S'76–M'81–SM'92–F'98) received the B.E. degree in electrical engineering from the American University of Beirut, Beirut, Lebanon, in 1975, and the M.S.E.E. and Ph.D. degrees in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, in 1977 and 1981, respectively.

In 1981, he was with the Department of Electrical Engineering at Carnegie Mellon University as a Visiting Assistant Professor. From 1982 to 1988, he was with the Semiconductor Engineering Computer-Aided Design Group at Digital Equipment Corporation in Hudson, MA, where he headed the Analysis and Simulation Advanced Development Team. Since September 1988, he has been with the University of Michigan, Ann Arbor, as a Professor of Electrical Engineering and Computer Science. From September 1994 to March 1995, he was with the Cadence Berkeley Laboratory in Berkeley, CA, on a six-month sabbatical leave. He has authored or coauthored more than 120 papers and has presented seminars and tutorials at many professional meetings and various industrial sites. His research interests include the area of computer-aided design with emphasis on simulation, timing verification and optimal clocking, logic and layout synthesis, Boolean satisfiability, and design verification.

Dr. Sakallah was an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS from 1995 to 1997 and has served on the program committees of the International Conference on Computer-Aided Design, Design Automation Conference, and the International Conference of Computer Design as well as and numerous other workshops. He is currently an Associate Editor of the IEEE TRANSACTIONS ON COMPUTERS. He is a member of the ACM and Sigma Xi.

Rob A. Rutenbar (S'77–M'84–SM'90–F'98) received the Ph.D. degree from the University of Michigan, Ann Arbor, in 1984.

He joined the faculty of Carnegie Mellon University, Pittsburgh, PA, in 1984. He is currently the Stephen J. Jatras Professor of Electrical and Computer Engineering, and (by courtesy) of Computer Science. From 1993 to 1998, he was Director of the CMU Center for Electronic Design Automation. He is a cofounder of Neolinear, Inc., and served as its Chief Technologist on a 1998 leave from CMU. He is the founding Director of the MARCO/DARPA Center for Circuits, Systems, Software (C2S2), a consortium of U.S. universities chartered in 2001 to explore long-term solutions for next-generation circuit challenges. His research interests include circuit and layout synthesis algorithms for mixed-signal ASICs, for high-speed digital systems, and for FPGAs.

In 1987, Dr. Rutenbar received a Presidential Young Investigator Award from the National Science Foundation. He has won Best/Distinguished paper awards from the Design Automation Conference (1987), the International Conference on CAD (1991), and the Semiconductor Research Corporation TECHCON (1993 and 2000). He has been on the program committees for the IEEE International Conference on CAD, the ACM/IEEE Design Automation Conference, the ACM International Symposium on FPGAs, and the ACM International Symposium on Physical Design. He also served on the Editorial Board of IEEE SPECTRUM. He was General Chair of the 1996 ICCAD. From 1992 through 1996, he chaired the Analog Technical Advisory Board for Cadence Design Systems. In 2001, he was co-winner of the Semiconductor Research Corporation's Aristotle Award for contributions to graduate education. He is a member of the ACM and Eta Kappa Nu.