Satisfiability-Based Layout Revisited: Detailed Routing of Complex FPGAs Via Search-Based Boolean SAT

Gi-Joon Nam
Department of EECS
University of Michigan
Ann Arbor, MI 48109-2122
Phone: (734) 936-2828
criteria@eecs.umich.edu

Karem A. Sakallah
Department of EECS
University of Michigan
Ann Arbor, MI 48109-2122
Phone: (734) 936-1350
karem@eecs.umich.edu

Rob A. Rutenbar
Department of ECE
Carnegie Mellon University
Pittsburgh, PA 15213
Phone: (412) 268-3334
rutenbar@ece.cmu.edu

1. ABSTRACT

Boolean-based routing transforms the geometric FPGA routing task into a single, large Boolean equation with the property that any assignment of input variables that "satisfies" the equation (that renders equation identically "1") specifies a valid routing. The formulation has the virtue that it considers all nets simultaneously, and the absence of a satisfying assignment implies that the layout is unroutable. Initial Boolean-based approaches to routing used Binary Decision Diagrams (BDDs) to represent and solve the layout problem. BDDs, however, limit the size and complexity of the FPGAs that can be routed, leading these approaches to concentrate only on individual FPGA channels. In this paper, we present a new search-based Satisfiability (SAT) formulation that can handle entire FPGAs, routing all nets concurrently. The approach relies on a recently developed SAT engine (GRASP) that uses systematic search with conflict-directed non-chronological backtracking, capable of handling very large SAT instances. We present the first comparisons of search-based SAT routing results to other routers, and offer the first evidence that SAT methods can actually demonstrate the unroutability of a layout. Preliminary experimental results suggest that this approach to FPGA routing is more viable than earlier BDD-based methods.

1.1 Keywords

Boolean satisfiability, FPGA routing, conflict-directed search.

2. INTRODUCTION

Field-Programmable Gate Arrays (FPGAs) have adopted and successfully adapted a variety of ASIC layout techniques. Iterative improvement placers [4], maze-style [19] and channel-style routers [16] are extremely common here. But the discrete nature of FPGA logic blocks and routing fabrics also admits unique layout strategies that each strives to leverage the limited palette of geometric alternatives to prune the space of viable layouts. Search-based global and detailed routers [3][5], channel routers [15], and simultaneous placer/routers [22][28] 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 [6], statistical estimators [10], simultaneous placer/routers [22], and routing tactics that can heuristically abandon the layout when unroutability appears inevitable [27] 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*?

In [29][30] we suggested a radical transformation of the routing problem to address this question. Rather than deal with the layout as a geometric problem, we deal with it as a Boolean problem. We render the routing constraints as a single, large Boolean equation which is *satisfiable* (has an assignment of input variables such that the equation evaluates to "1") if and only if the layout is routable. The idea is essentially a variant of pattern routing, which has long been used in both board and chip-level routing. Each 2-pin connection may be assigned to a numerable set of possible paths through the FPGA routing fabric. Each alternative is encoded with a unique binary vector; these variable values determine the exact final routing. Since path patterns can quickly become complex and their number unmanageable, we enumerate patterns only through small regions of the fabric, and "assemble" patterns together to complete nets. Other Boolean equations serve to constrain this "assembly" process: *connectivity* constraints ensure that path patterns connect legally to make full net routes; *exclusivity* constraints ensure that two electrically different nets do not occupy the same routing resource at the same time. A particular virtue of this formulation, which we refer to as *Boolean-based routing*, is that much of the geometric complexity of the interaction among patterns is hidden, and rendered *implicitly* in the Boolean constraint formulas. Also, all paths for all nets are considered simultaneously. In [29][30], we solved the resulting Boolean problem by creating a Binary Decision Diagram (BDD, [7][8]) 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 indentically "0". Unfortunately, BDDs are also 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, in [29][30] we decomposed our FPGA layouts into channel-wise slices, and used BDD-based routing only on each channel.

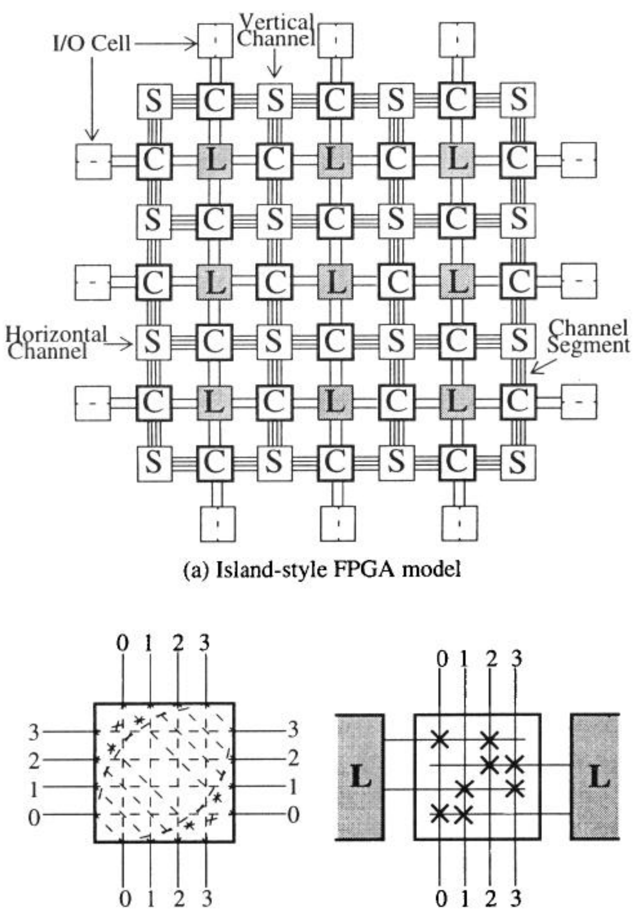
Obviously, a more desirable approach would be to model the entire FPGA routing problem by a single Boolean function, but this appears to be impractical for large designs using BDDs as the solution engine. A BDD-based approach essentially "solves" the function yielding *all* satisfying assignments. If 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 apporaches are commonly referred to as satisfiability (SAT, for short) algorithms.

The SAT community today is surprisingly large and vigorous [12], due to the broad range of problems which can be modeled as instances of SAT. Today, search-based SAT solvers appear to have the edge in robustness and scalability. Many search-style solutions have been proposed for SAT, the most well known being variations of the Davis-Putnam procedure [11]. The best known version is based on a backtracking search algorithm that, at each node in the search tree, elects an assignment and prunes subsequent search by iteratively applying the *unit clause* and the *pure literal* rules [33]. Iterated application of the unit clause rule is commonly referred to as *Boolean Constraint Propagation* (BCP) or as *derivation of implications* in the VLSI CAD literature [1]. Interest in the direct application of SAT algorithms to EDA problems has been on the rise recently [9] [18][21][26]. In addition, improvements to the traditional structural (path sensitization) algorithms for some EDA problems, such as ATPG, include search-pruning techniques that are also applicable to SAT algorithms in general [14][17][24].

In this paper we revisit the Boolean-based routing ideas from [29][30], and generalize them to 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 formulation. We also 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. Our results are made possible through the use a recently developed search-based SAT engine called **GRASP** [25], developed at the University of Michigan. It is the ability of GRASP to handle much larger Boolean functions than our former BDD methods that has allowed us to attack more complex FPGAs. The rest of paper is organized as follows. Section 3 reviews the typical island-style FPGA model that we employ. Section 4 develops our new SAT-based detailed FPGA routing algorithm, called **SDR** ("Satisfiability-based Detailed Router"). Section 5 describes the GRASP search-based SAT approach. Section 6 presents extensive experimental results showing the comparative performance of SDR on standard benchmarks. Finally, Section 7 offers some concluding remarks.

3. Target FPGA Architecture Model and Terminology

We based our modeling on a standard *island-style* FPGA architecture (e.g., Xilinx 4000 type [32]). This is one of the most commonly used layout models in FPGA applications, and is depicted in Fig. 1(a). 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



(a) Island-style FPGA model

(b) Switching Block, $F_s = 3$ (c) Connection Block, $F_c = 2$

Figure 1. Assumed FPGA Architecture Model. [20]

programmable switches and form the routing resources. C-blocks connect CLB pins to channels via programmable switches. S-blocks are surrounded by C-blocks and allow signals to either pass through or make 90-degree turns. Personalization of routing resources is achieved through proper programming of the routing switches. IO cells reside on the boundary of the array.

A *net* is a set of CLB pins that must be electrically connected and can be decomposed into one or more horizontal and/or vertical *net-segments*, each of which is an alternating sequence of C- and S-blocks that forms an uninterrupted path. 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 overlapping among detailed routes of different nets occurs. Our focus for SAT routing is detailed routing in this scenario, i.e., given a placement and a global routing, find a legal detailed routing, or demonstrate that no such detailed routing exists.

The routing capacity of a given FPGA architecture is conveniently expressed by 3 parameters, W, F_c, F_s [6]. The channel width W is the number of tracks in a vertical or horizontal channel. The C-block flexibility F_c is defined to be the number of tracks that each logic pin can connect to. The S-block flexibility F_s denotes the number of other tracks that each wire segment entering an S-block can connect to. In the sequel, we assume that all the vertical and horizontal channels 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 (c), each logic pin can be connected up to any 2 tracks in the C-block; thus $F_c = 2$.

4. SDR: A Satisfiability-Based Detailed FPGA Router

SDR (Satisfiability-based Detailed Router) is a newly developed detailed FPGA routing program that relies on the GRASP [25] SAT solver. SDR casts a detailed FPGA routing problem as a CNF (Conjunctive Normal Form) satisfiability problem that can be input to a Boolean SAT solver. The basic idea is that we construct a set of CNF clauses representing routing constraints over the entire FPGA assuming particular values for the routing flexibility parameters W, F_c, F_s. Then, satisfying assignments can be found by efficient backtracking search of the Boolean space using GRASP. A satisfiable CNF clause problem instance implies that the circuit is routable on the given FPGA architecture, and any satisfying assignment of Boolean values to variables corresponds to a specific detailed routing solution. SDR considers horizontal and vertical channels simultaneously, thereby removing the artificial boundary conditions inherent in the vertical-channel-only approach of [29]. As our experiments demonstrate, the CNF-based SAT formulation adds more modeling flexibility and enables the solution of problems that are too large to be handled by BDD-based methods.

To route simultaneously through both horizontal and vertical channels we generate a routing function $R(X)$ where X is a suitable vector of binary variables that encode the horizontal and vertical track numbers. This function can be expressed as the conjunction $R(X) = C(X) \wedge E(X)$ where:

- $C(X)$ model net **connectivity constraints** that insure the existence of a conductive path for each 2-pin net 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. Any reasonable routing architecture can be accommodated via proper Boolean function manipulations.

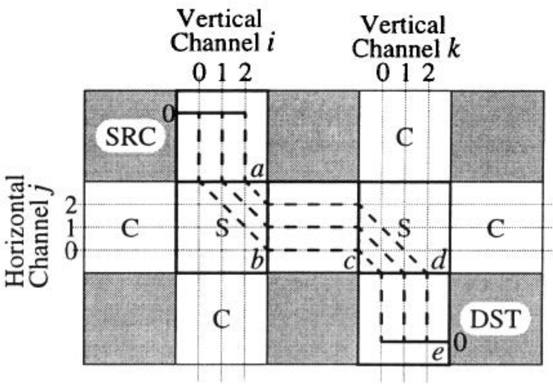
- $E(X)$ model net **exclusivity constraints** to insure that electrically distinct nets with overlapping vertical or horizontal spans in the same channel are assigned to different tracks.

The construction of the connectivity constraints is illustrated in Fig. 2 (a) and (b) for a net N whose global route specification is *(pin 0 of CLB SRC, C-block a, S-block b, C-block c, S-block d, C-block e, pin 0 of CLB DST)*. Assuming a simple routing fabric with routing flexibility $F_c = W = 3$ and $F_s = 1 \times 3$, net N can be modeled by three 2-bit vectors: V_i, and V_k which encode the net's track assignment in vertical channels i and k, and H_j which represents the net's track assignment in horizontal channel j. The connectivity constraints can now be conveniently expressed as shown in Fig. 2 (b). For example, a C-block constraint, such as C_a expresses the fact that, given $F_c = 3$, net N can be assigned to any of the three tracks in vertical channel i. Similarly, an S-block constraint, such as S_b, expresses the fact that a net entering on track i from the top must exit on track i on the right.

Fig. 2 (c) illustrates the construction of exclusivity constraints. Since the horizontal span of net A in channel m intersects those of nets B and C, net A should be assigned to a different track from the others. However, net B and C can share the same track in a legal routing solution because their horizontal spans do not overlap.

The overall flow diagram of the SDR algorithm is shown in Fig. 3. In the diagram, a rectangle indicates a procedure and an oval denotes an object generated by the preceding procedure. The remainder of this section describes each procedure of the algorithm in detail.

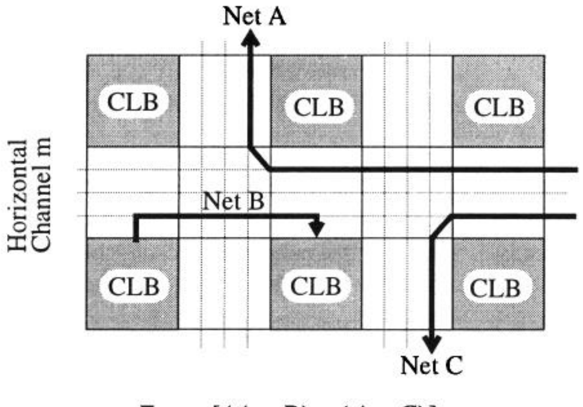
1. **Global Routing:** Given a placement, SDR invokes a global router to assign each net to a sequence of atomic routing regions consisting of C- and S-blocks. The global router does not choose or fix any specific detailed routing resources. SDR accepts global route information either as a set of 2-pin connections or as a set of more general multi-pin connections.
2. **Net Distribution:** SDR's *net-distribute* 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).



(a) Global route specification for net N (shaded boxes are CLBs)

$$C_a = [(V_i \equiv 0) \vee (V_i \equiv 1) \vee (V_i \equiv 2)]$$
$$S_b = [(V_i \equiv 0) \rightarrow (H_j \equiv 0)] \wedge [(V_i \equiv 1) \rightarrow (H_j \equiv 1)] \wedge [(V_i \equiv 2) \rightarrow (H_j \equiv 2)]$$
$$C_c = [(H_j \equiv 0) \vee (H_j \equiv 1) \vee (H_j \equiv 2)]$$
$$S_d = [(H_j \equiv 0) \rightarrow (V_k \equiv 0)] \wedge [(H_j \equiv 1) \rightarrow (V_k \equiv 1)] \wedge [(H_j \equiv 2) \rightarrow (V_j \equiv 2)]$$
$$C_e = [(V_k \equiv 0) \vee (V_k \equiv 1) \vee (V_k \equiv 2)]$$

(b) Connectivity constraints for net X. C formulas model C-blocks; S formulas model S-blocks.



$$E_m = [(A \neq B) \wedge (A \neq C)]$$

(c) Exclusivity constraint in horizontal channel m

Figure 2. Generation of Connectivity and Exclusivity Constraints in SDR.

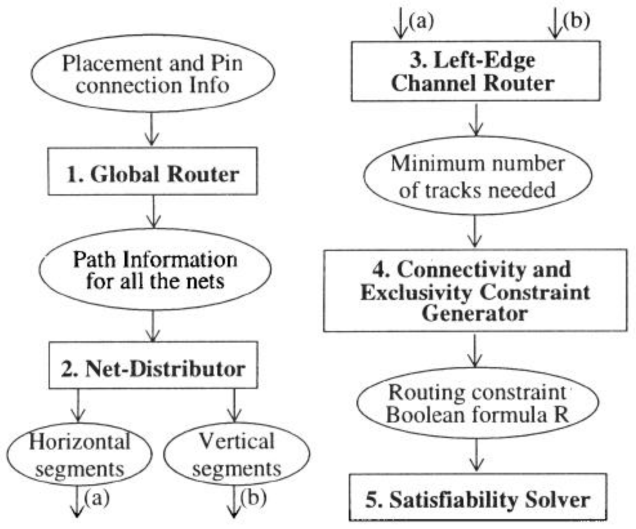
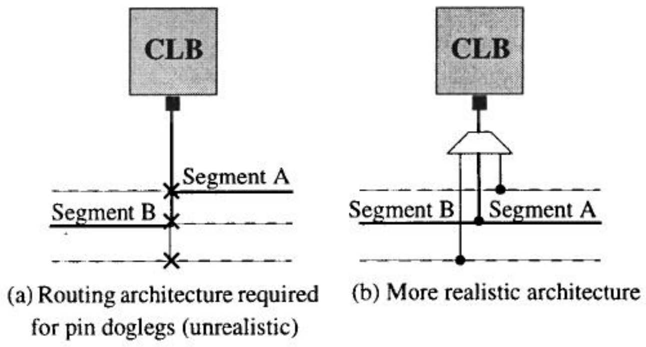

(a)
(b)
Placement and Pin connection Info
1. Global Router
Path Information for all the nets
2. Net-Distributor
Horizontal segments
Vertical segments
(a)
(b)
3. Left-Edge Channel Router
Minimum number of tracks needed
4. Connectivity and Exclusivity Constraint Generator
Routing constraint Boolean formula R
5. Satisfiability Solver

Figure 3. Flow Diagram of SDR

3. **Track Count Estimation:** If the target FPGA architecture is not provided by the user, SDR determines a channel width W by applying a *left-edge channel routing algorithm* [16]. 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 sets $W = W_{min}$. The size of the Boolean vectors necessary to encode track number assignments for each net segment is also set to $\lceil \log_2(W) \rceil$.
4. **Constraint Generation:** Connectivity and exclusivity constraints are generated, as described above, to yield the routing constraint function $R(X)$ in conjunctive normal form. $R(X)$ will be referred to as the "CNF clause database."
5. **Constraint Evaluation:** The GRASP solver is invoked to find a satisfying assignment for $R(X)$ or to show that $R(X)$ is identically "0".

The above steps comprise the basic overall flow for SAT routing. However, there is one additional subtlety that must be addressed to achieve practical routing solutions. We address this next.

The formulation procedure described above might allow "pin doglegs" whenever necessary. Pin doglegs allow us to route a net using more than one egress track per logic pin, as illustrated in Fig. 4. However, [3] and [34] point out that in commercial FPGAs, the logic pin is connected to routing wire segments via a multiplexer rather than a set of independent pass transistors, so that pin doglegs are not possible. Pin doglegs are easily prevented by adding an extra Boolean constraint, as illustrated in Fig. 4 (c). Suppose a net has two segments A, B starting from the same logic pin, but exiting


CLB
Segment A
Segment B
(a) Routing architecture required for pin doglegs (unrealistic)
CLB
Segment B
Segment A
(b) More realistic architecture

$$(A = B)$$
$$\equiv (A[0] = B[0]) \wedge (A[1] = B[1])$$

(c) Boolean function preventing pin doglegs

Figure 4. Pin Doglegs example [3], [34] and how to prevent doglegs using Boolean functions.

in different directions. If pin doglegs are allowed, A and B can be placed on different tracks in this channel. If pin doglegs are not allowed, we simply formulate a new Boolean function to force net segments A and B to be assigned the same track in a channel if they are from the same net. In the figure, we show the formula assuming A, B are each 2-bit vectors. In short, pin doglegs are easily controlled using Boolean function manipulation and show how SAT-based methods allow considerable flexibility in modeling low-level geometric constraints.

5. GRASP: A Conflict-Based Search Engine for Large SAT Problems

We solve the SAT instances created from our FPGA routing formulation using **GRASP** (Generic seaRch Algorithm for the Satisfiability Problem), an integrated algorithmic framework for SAT. Due to space constraints, we give only a brief overview of the technique; details appear in [25]. GRASP is premised on the inevitability of conflicts during search and its most distinguishing feature is the augmentation of basic backtracking search with a powerful conflict analysis procedure. Analyzing conflicts to determine their causes enables GRASP to backtrack non-chronologically to earlier levels in the search tree, potentially pruning large portions of the search space. In addition, by "recording" the causes of conflicts, GRASP can recognize and preempt the occurrence of similar conflicts later on in the search. Finally, straightforward bookkeeping of the causality chains leading up to conflicts allows GRASP to identify assignments that are necessary for a solution to be found. Experimental results obtained from a large number of benchmarks, both CAD-related and otherwise, indicate that application of the pro-

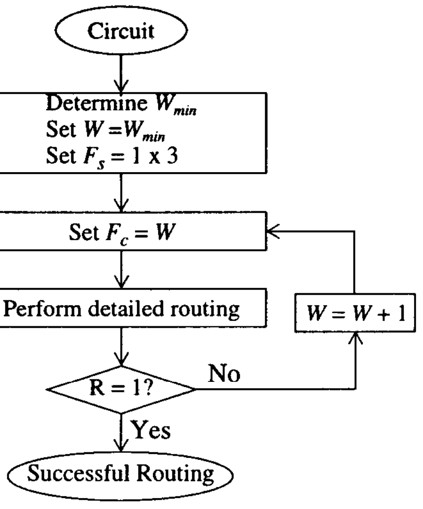


Figure 5. Experimental Testbench

posed conflict analysis techniques to SAT algorithms can be extremely effective.

It is also worth noting the operational differences between BDD and search-based SAT attacks. BDDs explicitly represent all possible satisfying assignments as paths through the BDD DAG. A BDD is unsatisfiable if and only if it is the trivial "0" BDD. In contrast, search-based solvers search to find just one satisfying assignment, and must search more exhaustively to conclude that no satisfying assignments exist. For us, the trade-off of more search-time for manageable memory size makes search-based SAT viable for FPGA routing.

6. Experimental Results

SDR can be employed in two different scenarios.

- When a placed and globally routed circuit as well as a target FPGA architecture are given, SDR is able to determine whether the circuit is routable in the given FPGA architecture. If routable, it provides the detailed net-to-track assignments for every net. Otherwise, it proves the unroutability which means there is no routing solution.
- When only a placed and globally routed circuit is given, SDR finds the "smallest" FPGA architecture, i.e. the architecture that has the minimum values of the W, F_c, F_s parameters, to render the given circuit routable.

All experiments were conducted on a SUN Ultra Sparc-2 running SunOS with 1 Gb of physical memory. Fig. 5 shows the experimental plan we employed. We assume no target FPGA is provided, but set the S-block flexibility at the minimum value $F_s = 1 \times 3$ 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 4000 series architecture model [32]. 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 of W starting from the lower-bound W_{min} found by the left-edge channel routing algorithm. The track width is incremented when GRASP determines that the routability function R is unsatisfiable or when it is aborted after running for 24 hours. The procedure terminates when GRASP concludes that R is satisfiable, and returns the corresponding track count W and detailed net-to-track assignments of the satisfying solution.

Table 1 shows SDR routing results for the geometrically simpler case when input pin doglegs are allowed. The circuit benchmarks are from [35], and range in size from 22 x 22 CLBs with 79 multi-pin Steiner nets for *9symml* to 48 x 48 CLBs with 404 multi-pin nets for *k2*. The benchmarks were placed by Altor [23] and globally routed using VPR [3]. The global routing data is provided in terms of 2-pin connections that were obtained by decomposing the multi-pin global routes generated by VPR.

The six columns under the SDR heading give the track count, size of the clause database, and CPU time obtained by SDR for the last unsatisfiable and first satisfiable instances of the routing function R (see Fig. 5). In other words, column 2 gives the largest value of W for which a benchmark was proved unroutable, whereas column 5 gives the smallest value of W for which routing succeeded. We note that in two cases (*k2* and *vda*), GRASP was aborted inconclusively leaving open the possibility that these two circuits could be routed with 9 and 7 tracks, respectively.

The most meaningful comparison in this table is between SDR and SEGA because they differ only in how detailed routing was done. Clearly, SDR is able to obtain significantly better results, achieving routability with on average 3.6 fewer required tracks per channel. The FPR and CGE results, which on these benchmarks are worse than SEGA's, are provided for reference.

SDR also compares surprisingly well with VPR achieving routability with the same number of tracks in all but the two cases highlighted in the table: *apex7* and *k2*. We conjecture that this is due to the use of different placements by the two programs.

We can observe immediately that typical solution times are low--less than 3 minutes in these cases. The times for showing unroutability are typically, but not always, longer. This is hardly surprising; in search-based SAT, we stop whenever we find a satisfying assignment, but we must search more exhaustively to deduce that no such assignment can exist. In two cases here (circuits k2 and vda) we were unable to search to completion on our proposed unroutable case. We also note that the number of clauses in the final CNF database for the routing turns out to be only a weak predictor of runtime; very large clause sets are likely to take longer, but it is the really the structural complexity of the route embedding itself that determines the runtime. For example, circuit too_large has 32K clauses, versus circuit vda's 67K, yet runs twice as long to find a satisfying assignment.

In Table 2 we compare performance of the various algorithms in the case where input pin doglegs are disallowed. 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. In this set of experiments, the most meaningful comparison is between columns 2 and 3 which differ only in the detailed router used. Unlike the results shown in Table 1, SDR's performance in this case are significantly worse than VPR's, on average requiring two more tracks per channel. This discrepancy, it turn outs, is due to an inadvertent routing restriction in SDR that *disallows output pin doglegs* in addition to disallowing input pin doglegs. The effect of placement on solution quality can also be clearly seen by comparing the second and third columns: using Altor for global placement requires, on average, 3 more tracks than using VPR.

Overall, we regard these as very satisfactory initial results for our preliminary implementation. Specifically, the search-based SAT approach seems more capable of handling complete FPGA routing tasks than our earlier BDD-based attack. Also, the preliminary evidence we have that we can determine unroutability--or, at least, *near*-unroutability, in the sense that we can find the track counts where SDR is able to demonstrate concretely the absence of any satisfying assignment--leads to some interesting possibilities. 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 [13] for an early discussion of the idea. With SDR we have the interesting possibility of determining *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. For example, in our experiments, if SDR fails at T tracks/channel, VPR succeeds at T or T-1 tracks. We are intrigued at the possibility that this sort of analysis may be able to

	SDR									
	Proving Unroutability			**Routable case**			**W from SEGA [20]**	**W from VPR [3]**	**W from FPR [2]**	**W from CGE [5]**
Circuit	**W**	**#Clauses**	**CPU (sec)**	**W**	**#Clauses**	**CPU (sec)**				
9symml	4	4932	4.89	5	10601	0.40	7	5	9	9
alu2	5	28675	1442.3	6	28751	2.65	8	6	10	12
apex7	4	5652	1.16	5	12191	0.48	10	4	9	13
example2	4	7308	2.30	5	15847	0.83	10	5	13	18
k2	9	226041	N.C.	10	227318	32.21	14	9	17	19
term1	4	5032	0.93	5	10804	0.44	8	5	8	10
too_large	5	32166	52.383	6	32375	134.50	10	6	11	13
vda	7	66194	N.C.	8	66608	74.44	12	8	13	14
Total ΣW	*	*	*	**50**	*	*	**79**	**48**	**90**	**108**

Table 1: Performance comparison between SDR and other routers with input pin doglegs allowed

	Placer	VPR	VPR	Altor	SPLACE
	Global Router	VPR	VPR	VPR	SROUTE
	Detailed Router	SDR	VPR	VPR	SROUTE
Circuits	9symml	7	5	6	7
	alu2	8	6	8	8
	apex7	6	4	9	6
	example2	7	5	10	7
	k2	11	9	14	11
	term1	6	4	7	5
	too_large	9	7	9	8
	vda	10	8	10	10
	Total ΣW	64	48	73	62

Table 2: Track number comparison when input pin doglegs are disallowed

predict the onset of "difficulty" in routing, the point at which many routes *must* deviate from simple patterns to embed.

7. Conclusion and Future Work

In this paper, we described a new strategy for SAT-based detailed FPGA routing. 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 result 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.

Two areas stand out for further work. One is simply to increase the amount of pattern flexibility for each net in our formulation, and explore how this impacts SAT-solve time versus layout quality. The other is to provide more feedback in the case where routing fails. Note that in a SAT formulation, a "no route" determination embeds no nets at all, unlike a typical net-at-a-time router which simply fails on the "difficult" nets. We believe it should by possible to add a set of auxiliary, "dummy" variables that allow each net to select to "opt out" of routing instead of embedding. By allowing some small, finite fraction (e.g., 1%) of nets to simply choose not to embed, we think it possible that we can offer more feedback to the designer when 100% completions proves to be impossible.

8. ACKNOWLEDGMENTS

We are grateful to Vaughn Betz and Guy Lemieux of the University of Toronto for providing valuable advice and benchmark circuits for our experiments. We are grateful to Texas Instruments for grants of the Sun workstations used for our experiments. This research was supported in part by NSF under grant 9404632.

9. REFERENCES

[1] M. Abramovici, M.A. Breuer and A.D. Friedman, *Digital Systems Testing and Testable Design,* Computer Science Press, 1990.

[2] M.J. Alexander, J.P. Cohoon, J.L. Ganley, and G. Robins, "Performance-Oriented Placement and Routing for Field-Programmable Gate Arrays," *Proc. European Design Auto. Conf.*, pp. 259-264, Sept. 1995.

[3] V. Betz and J. Rose, "VPR: A New Packing, Placement and Routing Tool for FPGA Research," *the Seventh Annual Workshop on Field Programmable Logic and Applications*, pp.213-222, 1997.

[4] M. A. Breuer, "A Class of Min-cut Placement Algorithms," *Proceedings of 14th Design Automation Conference,* pp.284-290, October 1997.

[5] S. Brown, J. Rose, and Z.G. Vranesic, "A Detailed Router for Field Programmable Gate Arrays," *IEEE Trans. CAD,* pp. 620-628, vol. 11, no. 5, May 1992.

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

[7] R.E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," *IEEE Trans. Computers*, pp. 677-691, 1986.

[8] R.E. Bryant, "Symbolic Boolean Manipulation with Ordered Binary Decision Diagrams," *ACM Computing Surveys*, vol. 24, no. 3, pp. 293-318, Sept. 1992.

[9] S. T. Chakradhar, V. D. Agrawal and S. G. Rothweiler, "A Transitive Closure Algorithm for Test Generation," *IEEE Transactions on Computer-Aided Design,* vol. 12, no. 7, pp.1015-1028, July 1993.

[10] P. K. Chan *et.al.*, "On Routability Prediction for Field Programmable Gate Arrays," *Proc. IEEE DAC,* June 1993.

[11] M. Davis and H. Putnam, "A Computing Procedure for Quantification Theory," *Journal of the Association for Computing Machinery,* vol. 7, pp. 201-215, 1960.

[12] DIMACS *http://DIMACS.Rutgers.EDU.*

[13] J. Dion, "Fast Printed Circuit Board Routing," *24th ACM/IEEE Design Automation Conf.*, June 1988.

[14] J. Giraldi and M. L. Bushnell, "Search State Equiva-

lence for Redundancy Identification and Test Generation," in *Proceedings of the International Test Conference*, pp. 184-193, 1991.

[15] J. Greene, V. Roychowdhury, S. Kaptanoglu, and A. El Gamal, "Segmented Channel Routing," *Proc. ACM/IEEE DAC*, pp. 567-572, 1990.

[16] A. Hashimoto, and J. Stevens, "Wire Routing by Optimizing Channel Assignment within Large Apertures," *Proceedings of 8th Design Automation Conference*, pp.155-169, 1971.

[17] W. Kunz and D. K. Pradhan, "Recursive Learning: An Attractive Alternative to the Decision Tree for Test Generation in Digital Circuits," in *Proceedings of the International Test Conference*, pp. 816-825, 1992.

[18] T. Larrabee, *Efficient Generation of Test Patterns Using Boolean Satisfiability*, Ph.D Dissertation, Department of Computer Science, Stanford University, STAN-CS-90-1302, February 1990.

[19] C. Y. Lee, "An Algorithm for Path Connections and its applications," *IRE Transactions on Electronic Computers*, 1961.

[20] G. Lemieux, S. Brown, and D. Vranesic, "On Two-Step Routing for FPGAs," *International Symposium on Physical Design*, pp.60-66, April 1997.

[21] P. C. McGeer, A. Saldanha, P. R. Stephan, R. K. Brayton and A. L. Sangiovanni-Vincentelli, "Timing Analysis and Delay-Fault Test Generation Using Path Recursive Functions," in *Proceedings of the International Conference on Computer-Aided Design*, pp. 180-183, 1991

[22] S. K. Nag and R. A. Rutenbar, "Performance-Driven Simultaneous Placement and Routing for FPGA's," *IEEE Transactions on CAD*, pp. 499-518, June 1998.

[23] J. S. Rose, W. M. Snelgrove, Z. G. Vranesic, "ALTOR: An Automatic Standard Cell Layout Program," *Canadian Conf. on VLSI*, pp. 169-173, 1985.

[24] J. P. M. Silva and K. A. Sakallah, "Dynamic Search-Space Pruning Techniques in Path Sensitization," in *Proc. IEEE/ACM Design Automaton Conference (DAC)*, pp. 705-711, June 1994.

[25] J. P. M. Silva and K.A. Sakallah, "GRASP--A New Search Algorithm for Satisfiability," *Proc. ACM/IEEE ICCAD*, Nov. 1997.

[26] P. R. Stephan, R. K. Brayton and A. L. Sangiovanni-Vincentelli, "Combinational Test Generation Using Satisfiability," Memorandum no. UCB/ERL M92/112, Department of Electrical Engineering and Computer Science, University of California at Berkeley, October 1992.

[27] J. S. Swartz, V. Betz, and J. Rose, "A Fast Routability Driven Router for FPGAs," *Proc. ACM Intl. Symp. FPGAs*, Feb 1998.

[28] N. Togawa, M. Yanagisawa, and T. Ohtsuki, "Maple-opt: A Performance-Oriented Simultaneous Technology Mapping, Placement, and Global Routing Algorithm for FPGA's," *IEEE Transactions on CAD*, pp. 803-815, Sept. 1998.

[29] R.G. Wood and R.A. Rutenbar, "FPGA Routing and Routability Estimation Via Boolean Satisfiability," *Proc. ACM International Symposium on FPGAs*, Feb. 1997.

[30] R. G. Wood and R. A. Rutenbar, "FPGA Routing and Routability Estimation Via Boolean Satisfiability," *IEEE Transactions on VLSI Systems*, pp. 222 - 231, June 1998.

[31] S. Wilton, "Architectures and Algorithms for Field-Programmable Gate Arrays with Embedded Memories," *Ph.D Dissertation*, University of Toronto, 1997.

[32] XILINX, *The Programmable Gate Array Data Book*, Xilinx, Inc., San Jose, California, 1993

[33] R. Zabih and D. A. McAllester, "A Rearrangement Search Strategy for Determining Propositional Satisfiability," in *Proceedings of the National Conference on Artificial Intelligence*, pp. 155-160, 1988.

[34] http://www.eecg.toronto.edu/~vaughn/challenge/challenge.html

[35] http://www.eecg.toronto.edu/~lemieux/sega/sega.html