A Comparative Study of Two Boolean Formulations of FPGA Detailed Routing Constraints

Gi-Joon Nam†, Fadi Aloul†, Karem Sakallah† and Rob Rutenbar‡
†Department of EECS, University of Michigan, {criteria, faloul, karem}@eecs.umich.edu
‡Department of ECE, Carnegie Mellon University, rutenbar@ece.cmu.edu

ABSTRACT

A Boolean-based router expresses the routing constraints as a Boolean function which is satisfiable if and only if the layout is routable. Compared to traditional routers, Boolean-based routers offer two unique features: (1) simultaneous embedding of all nets regardless of net ordering, and (2) ability to demonstrate routing infeasibility by proving the unsatisfiability of the generated routing constraint Boolean function. In this paper, we introduce a new Boolean-based FPGA detailed routing formulation that yields an easy-to-evaluate and more scalable routability Boolean function than the previous methods. The routability constraints are expressed in terms of a set of "route" variables each of which designating a specific detailed route for a given net. Experimental results clearly show the superiority of this formulation over an earlier formulation that expressed the constraints in terms of "track" variables.

1. INTRODUCTION

Largely due to the significant improvement in their density and performance over the last few years, Field-Programmable Gate Arrays (FPGAs) have become an increasingly attractive design medium. Yet the majority of FPGA synthesis algorithms are primarily adapted from ASIC layout techniques. Iterative improvement placers [4], annealing-based placement algorithms [3], maze-style [12] and channel-style routers [11] are all common examples in this class. More specifically, most FPGA routing algorithms are based on a "net-at-a-time" paradigm and are formulated as a search for paths in a graph that models the FPGA routing fabric. A circuit is considered to have been successfully routed when each of its nets has been mapped to a path in this graph while minimizing a suitable cost function (e.g. delay). The routers described in [2, 3, 5, 14, 16, 25] are representative of this general approach, and have proved to be quite effective. Unlike ASICs, however, FPGAs have fixed routing resources and such algorithms might thrash in a vain attempt to find a routing solution when none exists. Routability estimators [6, 8], simultaneous placer/routers [17], and routing tactics that heuristically abandon the layout when unroutability appears inevitable [21] have been proposed as possible remedies to this problem. Nevertheless, it is still a practical impossibility to definitively determine the routability of a given FPGA placement.

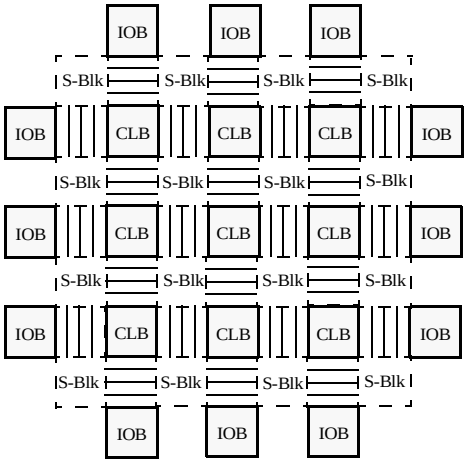
A recent alternative algorithmic paradigm for FPGA routing renders a circuit's routing constraints as a large, but atomic, Boolean function whose satisfying assignments correspond to feasible routing solutions. In contrast to the net-at-a-time algorithms, this *Boolean-based routing* approach considers all circuit nets simultaneously, and can be used to prove unroutability. This approach was first reported in [24] where the routing Boolean function was represented symbolically as a Binary Decision Diagram (BDD) [7]. Using a BDD had the advantage of capturing all possible routing solutions as traversals from the BDD's root node to the "1" leaf node. Additionally, unroutable layouts corresponded to the degenerate BDD consisting of the "0" leaf node. The only problem with the BDD representation had to do with scale: some large FPGA routing instances led to BDDs that grew too large to fit within available computer memory, even when advanced variable ordering heuristics were employed. A compromise channel-at-a-time approach was, therefore, adopted: the routing constraints were decomposed in order to produce smaller BDDs that captured the routability in each of the FPGA's channels separately. Boundary conditions were added to insure that the solutions of the independent channel routing functions can be stitched together to produce a complete routing solution.

The first successful effort to model and solve the routing Boolean function for an entire FPGA was reported in [19]. Instead of a BDD, the routing constraints were represented in conjunctive normal form (CNF) and fed to a Boolean satisfiability (SAT) solver. Using advanced techniques for implicit systematic search in the *n*-dimensional Boolean space of the function's input variables, the SAT approach overcame the memory explosion problem of BDDs and led to the successful representation and solution of larger routing functions. The first SAT-based results from routing entire FPGAs were quite competitive compared to other published FPGA routers. Still, several routing instances required excessive run times to solve; worse yet, some instances were unsolvable even after the SAT solver ran for a substantial amounts of time (e.g., 24 hours.)

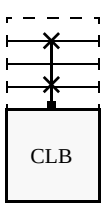
In this paper, we analyze the formulation method presented in [19] and introduce a revised formulation that encodes the routing constraints more efficiently leading to demonstrably easier-to-solve SAT instances. The rest of the paper is organized as follows. Section 2 presents the routing architectural model we employ. The previous Boolean SAT formulation of the FPGA detailed routing problems is described in Section 3. In Section 4, we describe the revised formulation method and compare it to the original formulation. An experimental evaluation of the two formulations is presented in Section 5, and Section 6 summarizes the paper's conclusions and future directions of this approach to FPGA routing.

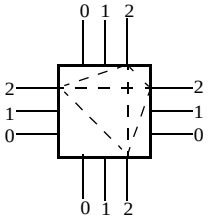
2. FPGA Routing Architecture

Our detailed routing formulations are based on the *island-style* FPGA architecture. This style is one of the most commonly used layout models in the literature [3, 5, 14, 19], and can be easily adapted to reflect a



(a) Island-style FPGA Routing Model





(b) Connection Block, $F_C = 2$ (c) Switching Block, $F_S = 3$

Figure 1. Island-style FPGA Routing Architecture Model.

variety of commercial FPGAs. As depicted in Figure. 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*). A vertical (horizontal) channel is defined as a set of tracks between two consecutive columns (rows) of CLBs; wire segments connect CLB pins to the adjacent routing tracks. C- and S-blocks contain programmable switches and form the routing resources: C-blocks connect CLB pins to channel tracks; S-blocks are surrounded by C-blocks and allow signals to either pass straight through or to make 90-degree turns. I/O cells reside on the boundary of the array.

The routing capacity of a given FPGA architecture is conveniently expressed by three 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 $F_C \leq W$ is the number of tracks in adjacent channels that each CLB logic pin may connect to. The S-block flexibility F_S is the total number of other tracks that each wire segment entering an S-block can connect to. For the example FPGA in Figure. 1 these parameters are $W = 3$, $F_C = 2$, and $F_S = 3$.

For our experiments, we configured this general layout model to mimic the Xilinx® XC4000E/X™ series architecture [28]. In this specific FPGA architecture, each CLB logic pin can connect to any channel track (i.e., $F_C = W$), and each wire segment entering an S-block can connect to the same-numbered tracks on each of the other three sides (i.e., $F_S = 3$). We also assume that every wire is fully segmented (i.e., it spans only one block length) based on the results of the global router [3] we used.

3. Track-Based Routing Constraint Model

The SAT formulation of FPGA detailed routing constraints introduced in [19] can be viewed as a net-to-track assignment problem. Each net in the layout is represented by a set of "track" variables that indicate the indices of the horizontal and vertical tracks over which the net might be routed. A routability Boolean function is then defined over these variables to enforce two types of constraints:

- **Connectivity constraints** to insure the existence of a conductive path for each two-pin connection through the sequence of C- and S-blocks specified by a global router. These constraints model the routing flexibility available in the C- and S-blocks.
- **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.

In the sequel we will refer to this formulation as the *track-based routing constraint model* to emphasize the fact that it is defined over a set of variables that represent the tracks available for routing, and to distinguish it from the new formulation introduced in Section 4.

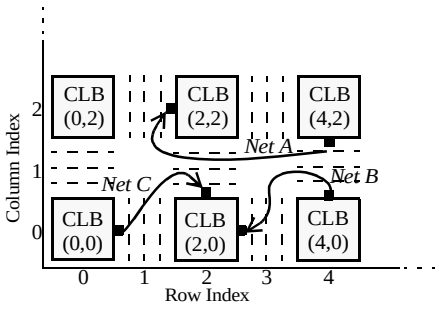
An example that illustrates this formulation is shown in Figure. 2 for an FPGA with $W = F_C = F_S = 3$. Each net is assumed to have been assigned a global route (sequence of C- and S-blocks) by a global router (Figure. 2-a). Track variables are then created for each net to model its possible assignment to specific tracks in each of the channels specified by its global route. For instance, two track variables are associated with net A: *AH* and *AV* to indicate its track assignment in horizontal channel 1 and vertical channel 1 respectively. These track variables are multi-valued with a domain $\{0, \ldots, W-1\}$. In the actual formulation, each of these multi-valued variables is encoded by $\lceil \log_2 W \rceil$ Boolean variables using the standard decimal-to-binary encoding.

The construction of the connectivity and exclusivity constraints is depicted in Figure. 2-b and c. The connectivity constraint for a given net restricts the net's track variables to those values that insure 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 through S-block(1,1) which has a flexibility $F_S = 3$ (see Figure. 1.) The exclusivity constraints in this example insure that nets A and B as well as A and C are assigned to different track numbers in horizontal channel 1. Figure. 2-d shows the actual CNF representation of an exclusivity constraint between two 3-valued track variables. In general, an exclusivity constraint between two track variables leads to a set of W CNF clauses each consisting of $2 \cdot \lceil \log_2 W \rceil$ literals. The Boolean function that models the routability of these three nets is simply the conjunction of all the connectivity and exclusivity requirements:

$$R(X) = Conn(A) \wedge Conn(B) \wedge Conn(C) \wedge Excl(H1) \qquad (1)$$

where X is a vector of Boolean variables that encode the track variables AH, AV, BH, BV, CH, and CV.

Compared to previous conventional routing methods, this method has the following unique properties:



Net A: [pin 0 of CLB(4,2), C-block(4,1), S-block(3,1), C-block(2,1), S-block(1,1), C-block(1,2), pin 1 of CLB(2,2)]
Net B: [pin 2 of CLB(4,0), C-block(4,1), S-block(3,1), C-block(3,0), pin 3 of CLB(2,0)]
Net C: [pin 3 of CLB(0,0), C-block(1,0), S-block(1,1), C-block(2,1), pin 2 of CLB(2,0)]

enum {0, 1, 2} *AH, AV,* // Net A track variables
BH, BV, // Net B track variables
CH, CV; // Net C track variables

(a) Global routing configuration for nets A, B and C, and corresponding variable declarations.

$$Conn(A) = [(AH \equiv 0) \vee (AH \equiv 1) \vee (AH \equiv 2)] \wedge [AV = AH] \wedge [(AV \equiv 0) \vee (AV \equiv 1) \vee (AV \equiv 2)]$$

$$Conn(B) = [(BH \equiv 0) \vee (BH \equiv 1) \vee (BH \equiv 2)] \wedge [BV = BH] \wedge [(BV \equiv 0) \vee (BV \equiv 1) \vee (BV \equiv 2)]$$

$$Conn(C) = [(CV \equiv 0) \vee (CV \equiv 1) \vee (CV \equiv 2)] \wedge [CH = CV] \wedge [(CH \equiv 0) \vee (CH \equiv 1) \vee (CH \equiv 2)]$$

(b) Connectivity constraints.

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

(c) Exclusivity constraints.

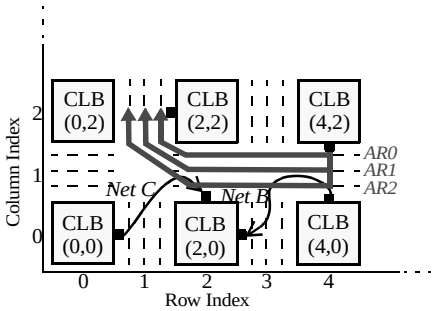
$$X \equiv [X_1 X_0] \quad Y \equiv [Y_1 Y_0]$$

$$X \neq Y = \overline{(X \equiv 0 \wedge Y \equiv 0) \vee (X \equiv 1 \wedge Y \equiv 1) \vee (X \equiv 2 \wedge Y \equiv 2)}$$
$$= \overline{(X \equiv 0 \wedge Y \equiv 0)} \wedge \overline{(X \equiv 1 \wedge Y \equiv 1)} \wedge \overline{(X \equiv 2 \wedge Y \equiv 2)}$$
$$= (X_0 \vee X_1 \vee Y_0 \vee Y_1) \wedge (\overline{X_0} \vee X_1 \vee \overline{Y_0} \vee Y_1) \wedge (X_0 \vee \overline{X_1} \vee Y_0 \vee \overline{Y_1})$$

(d) CNF representation of an exclusivity constraint between track variables *X* and *Y*.

Figure 2. Example of track-based detailed routing formulation.

- **Simultaneous net embedding:** The conventional *net-at-a-time* routing approach is notorious for being dependent on net ordering because previously-routed nets act as obstacles to the yet-to-be-routed nets. In this method, all the routing constraints are considered concurrently by a Boolean SAT solver, making net ordering irrelevant.



bool *AR0, AR1, AR2,* // Net A route variables
BR0, BR1, BR2, // Net B route variables
CR0, CR1, CR2; // Net C route variables

(a) Global routing configuration for nets A, B and C and three possible detailed routes for net A.

$$Live(A) = (AR0 \vee AR1 \vee AR2)$$
$$Live(B) = (BR0 \vee BR1 \vee BR2)$$
$$Live(C) = (CR0 \vee CR1 \vee CR2)$$

(b) Liveness constraints.

$$Excl(Resource(4, 1, 0)) = (\overline{AR0} \vee \overline{BR0})$$
$$Excl(Resource(4, 1, 1)) = (\overline{AR1} \vee \overline{BR1})$$
$$Excl(Resource(4, 1, 2)) = (\overline{AR2} \vee \overline{BR2})$$
$$Excl(Resource(2, 1, 0)) = (\overline{AR0} \vee \overline{CR0})$$
$$Excl(Resource(2, 1, 1)) = (\overline{AR1} \vee \overline{CR1})$$
$$Excl(Resource(2, 1, 2)) = (\overline{AR2} \vee \overline{CR2})$$

(c) Exclusivity constraints.

Figure 3. Example of route-based detailed routing formulation.

- **Flexible formulation ability:** This method can readily accommodate the constraints of any routing fabric. For example, using an FPGA with different C- and S-block architectural parameters requires only a slight modification of the connectivity constraint function $Conn(X)$.
- **Routability decision:** The unsatisfiability of the generated routing constraint Boolean function $R(X)$, as proven by a Boolean SAT solver, directly implies that there is no feasible routing solution with the given placement and global routing configuration. On the other hand, each assignment to the Boolean vector X that satisfies $R(X)$ corresponds to a complete feasible detailed routing solution for that placement and global routing configuration.

Despite these advantages, this formulation still leads to routing instances that cannot be solved in a reasonable amount of time (e.g. one day.) The formulation described next is motivated by the desirability of generating "easier" SAT instances that are scalable to larger FPGA routing problems.

4. Route-Based Routing Constraint Model

In the route-based formulation [18], the routability of a netlist is directly modeled in terms of Boolean variables that represent all of the detailed routes admissible by the given global routing solution. This choice of variables leads to a simpler set of constraints than those described above and enables the solution of larger routing instances.

This formulation is illustrated in Figure. 3. Within the global routing region specified for net A, for example, there are only three possible detailed routes indicated by the three Boolean variables $AR0$, $AR1$, and $AR2$. A similar set of routes and corresponding route variables is created for nets B and C. A particular route is included in the final routing solution if its corresponding Boolean variable is assigned the logic value 1, and is excluded as a routing option otherwise. With this choice of variables, the FPGA detailed routing problem is transformed from a track assignment to a "routability checking" problem. The routability of a netlist in terms of these "route" variables can now be expressed with two types of constraints:

- **Liveness constraints** to ensure that each two-pin connection has at least one detailed route selected in the final routing solution. The liveness constraint for a given two-pin connection has a simple form, namely an OR over the connection's F_C route variables (see Figure. 3-b). For a netlist with n two-pin connections, liveness constraints yield a set of n CNF clauses, each containing F_C positive literals.
- **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 semantically identical to those described earlier for the track-based formulation but have a much simpler CNF representation (see Figure. 3-c). For example, $Excl(Resource(4, 1, 0)) = (\overline{AR0} \vee \overline{BR0})$ indicates that the routing resource, track segment 0 of C-block(4,1), can only be used by either detailed route 0 of net A or detailed route 0 of net B, but not both. In general, if k different detailed routes from different nets are competing for the same routing resource, a set of $k \cdot (k-1)/2$ exclusivity constraints are created to insure that at most one of those detailed routes are assigned to that resource. Each of those constraints, in turn, is a simple CNF clause consisting of two complemented literals.

One possible concern of the route-based formulation is that when $F_S \geq 3$, the number of feasible detailed routes within a global routing path grows exponentially. Most of modern FPGA routing architecture, however, employ a switching block of type $F_s = 3$. With even more flexible switching block type, it is possible to bound the number of detailed routes with a global routing region to overcome this problem.

Each two-pin connection forms a complete path from a source pin to a sink pin and no further constraint is needed to stitch them together into a single multi-pin net.

Table 1: Benchmark Circuits.

Cir.	X x Y	#C	#N	#2p	Cir.	X x Y	#C	#N	#2p
9symml	9 x 9	70	79	259	exam2	19 x 19	120	205	444
alu2	12 x 12	143	153	510	k2	19 x 19	358	404	1257
apex7	11 x 11	77	126	300	term1	8 x 8	54	88	202
C499	10 x 10	74	115	312	too_lrg	13 x 13	148	186	519
C880	14 x 14	174	234	656	vda	15 x 15	208	225	722

The routability of a netlist for a given placement and global routing configuration is expressed by a single Boolean function which is the conjunction of all liveness and exclusivity constraints:

$$R(X) = [Live(n) \wedge Excl(r)] \ for \ n \in all \ Nets, \quad r \in all \ Resources \tag{2}$$

where X is a vector of Boolean variables that represent the possible detailed routes for each of the nets. With $F_c = W$ and $F_s = 3$ architectural assumption, the route-based formulation is truly equivalent to the track-based formulation, and preserves all the advantages of the previous Boolean-based routing formulation. In addition, for most circuits it requires fewer variables and is expressed in terms of a simpler set of CNF constraints as will be shown in the next section.

5. Experimental Results

We experimentally tested the effectiveness of the route-based formulation method on the standard MCNC benchmark circuits downloadable from [26]. The relevant properties of each of these circuit, listed in Table 1, include the size of the target FPGA CLB arrays (column "X x Y"), the actual number of CLBs used by the circuit ("#C"), the number of multi-pin nets ("#N"), and the corresponding number of two-pin connections that are routed individually ("#2p"). It should be noted that our formulation is based on two-pin connections; a multi-pin net whose global route is specified as a steiner-tree is decomposed into a set of two-pin connections prior to construction of the routability function. Such a decomposition facilitates pin-dogleggings at CLB output pins which generally yields layouts with fewer tracks [19].

The first experiment we conducted compares the performance of the track- and route-based detailed routing formulations. Using the placements and global routing solutions generated by VPR [3], routing functions based on templates Eq. (1) and Eq. (2) were pro-

Table 2: Performance Comparison Between Two Boolean Formulations of FPGA Detailed Routing.

Name	W	Track-based Formulation [19]						Route-based Formulation						R?	Speedup
		V	CL	Dec	Conf	G. Time	S. Time	V	CL	Dec	Conf	G. Time	S. Time		
9symml	**6**	2604	36994	11883	9910	0.74	471.12	1554	29119	347	0	0.67	1.94	Yes	242.85
	5	2604	32450	13896	11687	0.69	521.39	1295	24309	344	272	0.59	6.11	No	85.33
apex7	**5**	1983	15358	606	220	0.37	3.63	1500	11695	568	0	0.31	1.51	Yes	2.40
	4	1322	10940	445	336	0.25	3.59	1200	9416	293	191	0.23	1.38	No	2.60
exam2	**6**	3603	41023	1347	708	0.94	24.12	2664	27684	993	1	1.20	5.65	Yes	4.27
	5	3603	36344	12613	11063	0.83	531.42	2220	23144	1331	1245	1.00	25.16	No	21.12
term1	**4**	746	3964	322	134	0.11	0.51	808	3290	207	6	0.07	0.11	Yes	4.63
	3	746	3517	71	48	0.11	0.19	606	2518	12	12	0.07	0.03	No	6.33
C499	**6**	2070	22470	11381	10141	0.47	255.34	1872	18870	395	18	0.43	1.51	Yes	169.10
	5	2070	19908	11755	10696	0.42	322.76	1560	15777	372	355	0.38	4.66	No	69.26

Table 3: Size Distribution of the CNF Formulas.

Name	W	#Clauses of Size									
		Track-based Formulation					Route-based Formulation				
		1	2	3	4	>4	1	2	3	4	>4
9symml	**6**	0	4522	0	0	32472	0	28860	0	0	259
	5	0	5390	0	0	27060	0	24050	0	0	259
apex7	**5**	0	3488	0	0	11870	0	11395	0	0	300
	4	0	1444	0	0	9496	0	9116	0	0	300
exam2	**6**	0	5743	0	0	35280	0	27240	0	0	444
	5	0	6944	0	0	29400	0	22700	0	0	444
term1	**4**	0	684	0	3280	0	0	3088	0	0	202
	3	0	1057	0	2460	0	0	2316	0	202	0
C499	**6**	0	2958	0	0	19512	0	18558	0	0	312
	5	0	3648	0	0	16260	0	15465	0	0	312

duced for decreasing values of the channel width W. These routability Boolean functions were subsequently evaluated by the GRASP SAT solver [15]. In any practical situation, the main interest is to know whether designs can fit into target FPGAs or not, rather than to optimize the number of routing tracks used. However, we performed the experiments under the optimization scenario to facilitate the comparisons with other conventional routers later. Table 2 summarizes the results of the last two "iterations" in this process for six of the twelve benchmark circuits: the minimum channel width for which the benchmark circuit was still routable, and the maximum channel width for which it was proven to be unroutable (for the remaining benchmarks not listed in Table 2, the minimum width for routability could not be found using the track-based formulation; the performance of the route-based formulation on these benchmarks is discussed later). The columns in this table record, for each benchmark circuit, the following data: the assumed channel width ("W"), the number of Boolean variables and CNF clauses in the routability function ("V" and "CL"), the number of decisions and conflicts during the SAT search for a solution ("Dec" and "Conf"), and the CPU time spent for generating routability Boolean functions ("G.Time") and actual SAT search by GRASP ("S.Time"). Column "R?" indicates whether the routability function had a feasible solution; the computational advantage of the route-based formulation is shown in column "Speedup" which is the ratio of the CPU times taken by GRASP to solve the respective routability functions. The experiment was conducted on a Pentium III PC running Debian linux 2.2.18 with 512 MB of physical memory. The GRASP SAT solver was configured to use the "DLCS" decision heuristic.

We can observe immediately that typical solution times of the route-based formulation are much faster than the track-based method achieving 66x speedups on average. The actual numbers of decisions and conflicts during the SAT search validate the achieved speedups. In both formulations, the number of Boolean variables of the routability functions can be computed in the following way: assuming that the benchmark circuit has total n two-pin connections, the total number of Boolean variables in the routability function is exactly $n \cdot W$ in the route-based method whereas it is approximately $n \cdot \lceil \log_2 W \rceil \cdot m$ in the track-based formulation, where m is the average number of channels each global route of a two-pin connection passes through. Table 2 empirically demonstrates that the route-based formulation generates the routability function with few variables and clauses in all cases but one. The *term1* circuit with 4 tracks per channel is the only example that the route-based formulation requires more Boolean variables than the track-based formulation. This is because *term1* is a simply routable circuit requiring only 4 different global route channels, on average, per two-pin connection.

Table 4: The route-based formulation results for cases where the track-based one was unable to solve.

Circuit	W	V	Cl	Dec	Conf	G.Time	S.Time	R?
alu2	8	4080	83902	986	2	1.87	13.02	Yes
	7	3570	73478	9014	8968	1.66	1191.6	No
C880	7	4592	61745	1143	81	1.41	13.82	Yes
	6	3936	53018	40327	39546	1.21	52364.70	No
	5	3280	44291	612	598	1.02	21.83	No
k2	11	13827	372694	4199	16	8.93	280.61	Yes
	10	12570	338927	2902	51	8.01	186.08	Yes
	9	11313	305160	N.C[a]	N.C	7.19	N.C	N.C
	8	10056	271393	N.C	N.C	6.33	N.C	N.C
	7	8799	237626	17823	9873	5.81	4870.56	No
too_lrg	7	3633	50373	828	19	1.11	8.63	Yes
	6	3114	43251	1669	1184	0.93	59.01	No
	5	2595	36129	1163	1122	0.82	36.93	No
vda	9	6498	130997	1387	3	3.00	31.78	Yes
	8	5776	116522	25924	24861	2.72	6146.20	Yes
	7	5054	102547	23333	22098	2.28	12383.50	No

a. N.C stands for "Not Complete". Thus, the data is not available.

The numbers of variables and clauses, however, are not sufficient to explain the order-of-magnitude reduction in runtime. The analysis of clause size distribution (Table 3) reveals that the route-based formulation is mostly composed of 2-literal CNF clauses whereas the track-based formulation is composed mostly of clauses containing 3 or more literals. It is a well-known fact that 2-SAT has *P* class complexity while 3-SAT or higher SAT problems are *NP-complete* [10]. This is because, in 2-SAT, it is feasible to construct an implication graph to deduce solutions efficiently, which is impossible with higher order of SAT problems. Accordingly, the performance gain can be justified by the CNF clauses structure of the route-based constraint function, which are close to 2-SAT problem (albeit it is not exactly 2-SAT due to liveness constraint CNF clauses). Overall, this experiment suggests that it is more straightforward to find solutions for Boolean SAT instances from the route-based formulation.

In Table 4, we show the performance of the route-based formulation method over routing problems where the track-based method was not able to solve them within 24 hours limit--i.e, no conclusion was drawn whether given circuits are routable or not--. The number of decisions and conflicts are larger than those numbers in Table 2 suggesting that these are harder SAT instances. For only two cases, --*k2* with $W = 9$ and $W = 8$ -- even the route-based method could not solve the routability Boolean SAT problems. One interesting observation is that it is most difficult to make the routability decision (either routable or unroutable) with marginal track counts per channel. When a target FPGA has ample routing tracks per channel, the corresponding FPGA detailed routing solution should be straightforward to find. Similarly, if there are significantly insufficient routing resources, it is relatively easy to prove unroutability. The range of these marginal track counts--which form the boundary between routable and unroutable deci-

Table 5: Track number comparison with other conventional routers; VPR [3], SEGA [14], SROUTE [23], TRACER [13], IKMB [2], GBP [25] and FPR [1].

Placer	VPR			SPLACE	ALTOR [20]			FPR
G. Router				SROUTE	TRA-CER	IKMB	GBP	
D. Router	SAT	VPR	SEGA					
9symml	6	5	6	7	6	8	9	9
alu2	8	7	9	8	9	9	11	10
apex7	5	4	6	6	8	10	11	9
exam2	6	6	6	7	10	11	13	13
term1	4	4	6	5	7	8	10	8
too_lrg	7	6	9	8	9	10	12	11
k2	10	9	11	11	14	15	17	17
vda	8	8	10	10	11	12	13	13
C499	6	6	7	*[a]	*	*	*	*
C880	7	7	8	*	*	*	*	*
Total Σ W	**67**	**62**	**78**	**75**	**87**	**96**	**109**	**103**

a. These data were not available. For "Total W" calculation, we used the smallest values available in the table for the corresponding circuit.

sions-- seems to be narrow, usually 1 track per channel based on our experimental results. Typically, when the SAT-based approach cannot find a routing solution with T tracks per channel, the new routing SAT instance with T + 1 tracks per channel seems be straightforward. This opens an interesting possibility that the SAT-based routing approach can serve as an estimator of the difficulty of routing problems. If a routing instance cannot be solved within a hard execution time limit, the problem is considered very difficult, and most likely, a more flexible routing architecture is needed.

Finally, in Table 5, we compare performance of various FPGA detailed routers including our route-based formulation method shown under "SAT" heading. The table shows the number of tracks required to successfully route each benchmark circuit with the specified placement and global routing programs under track minimization scenario. The most meaningful comparison in this table is between the route-based method, VPR and SEGA because they differ only in how detailed routing was done. The data for the rest of routers are provided only for reference. Our route-based routing method shows the second best results among them, next to VPR [3] while beating the rest of them. Interestingly however, for 4 cases out of those 5 circuits (shaded cells in the table), our method proved the *unroutability* whereas VPR was able to successfully route them with the same numbers of tracks per channel. This difference is due to the fact that our method performs only detailed routing while VPR does both global and detailed routing. Thus VPR can change the global routing configuration when it cannot find a detailed routing solution easily. A more fair comparison is to extract the final detailed routing results from VPR and use them as a global routing solution for the route-based detailed routing formulation. Indeed, the route-based formulation approach was able to produce the exactly same routing results as VPR except one case. The route-based approach gave up finding a detailed routing solution of *k2* with 9 tracks per channel after 24 hours.

6. Conclusions and Future Work

In this paper, we introduced an improved Boolean SAT-based FPGA detailed routing formulation. This work differs from other SAT-based FPGA routing approaches because the routability constraints are expressed in terms of a set of "route" variables each of which designating a specific detailed route for a given net. The comparative experiments demonstrate that the route-based formulation yields an easy-to-evaluate and more scalable routability Boolean function than the previous methods.

Despite the success of Boolean methods so far, they still cannot compete with traditional routers on large-scale FPGAs. One possible approach to extend the applicability to larger FPGAs is to limit the number of detailed routes considered per connection. Instead of considering all the detailed routes within given global routing regions, only a small number of promising detailed routes can be selected based on projected routing congestions. In this way, we can not only reduce the size of the generated routability Boolean function, but also incorporate multiple global routing paths concurrently in order to escape from bad global routes.

7. References

[1] M. J. Alexander, J. P. Cohoon, J. L. Ganley, and G. Robins, "Performance-Oriented Placement and Routing for Field-Programmable Gate Arrays", *EDAC*, pp. 80 - 85, 1995.

[2] M. J. Alexander and G. Robins, "New Performance-Driven FPGA Routing Algorithms", *IEEE Trans. on CAD*, vol. 15, no. 12, pp. 1505 - 1517, Dec. 1996

[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, Oct. 1977.

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

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

[10] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H.Freeman and Company, 1979.

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

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

[13] Y.-S. Lee, A. Wu, "A Performance and Routability Driven Router for FPGAs Considering Path Delays", *Design Automation Conference*, pp. 557 - 561, 1995.

[14] G. Lemieux and S. Brown, "A Detailed Router for Allocating Wire Segments in FPGAs," *Proc. ACM Physical Design Workshop*, California, Apr. 1993.

[15] J. P. Marques-Silva and K. Sakallah, "GRASP: A Search Algorithm for Propositional Satisfiability", *IEEE Trans. on Computers*, vol. 48, no. 5, May 1999.

[16] L. E. McMurchie and C. Ebeling, "PathFinder: A Negotiation-Based Path-Driven Router for FPGAs," Proc. *ACM/IEEE International Symp. Field Programmable Gate Arrays*, Feb. 1995.

[17] S. K. Nag and R. A. Rutenbar, "Performance-Driven Simultaneous Placement and Routing for FPGAs", *IEEE Trans. on CAD*, pp. 499 - 518, June 1998.

[18] G. Nam, S. Kalman, J. Anderson, R. Jayaraman, S. Nag and J. Zhuang, "A Method and Apparatus for Testing Routability", *U.S. patent pending.*

[19] G. Nam, K. A. Sakallah, and R. A. Rutenbar, "Satisfiability-Based Layout Revisited: Detailed Routing of Complex FPGAs Via Search-Based Boolean SAT", *International Symposium on FPGAs*, Feb. 1999.

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

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

[22] S. M. Trimberger, *Field-Programmable Gate Array Technology*, Kluwer Academic Publishers, Boston, MA, 1994.

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

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

[25] Y-L. Wu and M. Marek-Sadowska, "Routing for Array-Type FPGAs", *IEEE Trans. on CAD*, vol. 16, no. 5, pp. 506 - 518, May 1997.

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

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

[28] http://www.xilinx.com/products/xc4000e.htm