Microelectronics Journal 46 (2015) 706–715

Contents lists available at ScienceDirect

Microelectronics Journal

journal homepage: www.elsevier.com/locate/mejo

SAT based solutions for detailed routing of island style FPGA architectures

Shyamapada Mukherjee*, Suchismita Roy

National Institute of Technology, Durgapur 713209, India

ARTICLE INFO

Article history:
Received 21 May 2014
Received in revised form
20 April 2015
Accepted 12 May 2015
Available online 12 June 2015

Keywords:
FPGA
Detailed routing
Boolean satisfiability
Switch box architecture

ABSTRACT

Detailed routing solutions for island style FPGA architectures using Boolean satisfiability (SAT) based formulations have been proposed in this paper. Due to decreasing size of ICs and hence, the increasing complexity of the routing resource constraints, routing has been a big challenge in electronic design automation field. Our proposed techniques work on multi-pin net routing where all nets are considered for routing in their intact form whereas, most of the existing routing solutions decompose multi-pin nets into two-pin nets for detailed routing to ease the problem. However this approach, apart from increasing the number of nets in the circuits, may also introduce pin doglegging which, when not permitted by the architecture of FPGA, would require extra constraints to eliminate. Many detailed routers adopt sequential detailed routing approaches which are vulnerable to the net ordering problem which may cause a routable circuit to be erroneously classified as unroutable. Our proposed techniques avoid these pitfalls by keeping the multi-pin nets intact and solve all nets simultaneously using SAT. The SAT-based multi-pin net dogleg-free formulations presented here achieve significant improvement over existing SAT-based solutions with respect to the number of variables and clauses used, thereby achieving greater scalability and also display comparable and sometimes better routability results on benchmark circuits when compared with other detailed routing solutions.
Detailed routing is also significantly affected by the architecture of the switching blocks. This paper proposes SAT-based formulation for three different switch box architectures i.e. Subset, Wilton, and Universal switches. Our experiments clearly demonstrate how routing solutions for a circuit can differ significantly for different types of switch boxes.

1. Introduction

FPGAs have become an increasingly popular electronic circuit design medium with great improvement in their performance. FPGA detailed routing problem is a very important and interesting problem in the electronics physical design automation (EDA) arena. Several algorithms have been tested for solving this problem. Many of the routing algorithms are based on the net-at-a-time approach. Net ordering is a big issue in the net-at-a-time approach because invalid sequence of net selection may cause the algorithm to fail.

In contrast to the net-at-a-time approach, considering all nets simultaneously provides good results, since all possible paths are explored simultaneously. However, the complexity of the problem is greatly increased and it requires a powerful computation framework. Boolean Satisfiability (SAT) based detailed routing is an interesting approach that makes it possible to consider all nets concurrently. In our proposed algorithms the FPGA detailed routing problem is converted into a series of constraints which are expressed as a single Boolean function. This function is then solved by any available SAT solver like GRASP [14], RelSAT [11], SATO [26], Zchaff [25] etc. Any valid assignment of the Boolean variables which causes the function to evaluate to true, expresses valid routability. SAT not only provides a valid solution for the problem; it searches all possible paths for a complete routing solution. If it is unable to find a solution, it proves that the circuit is unroutable.

A wide variety of tactics have been tried to solve the FPGA routing problem. In [15], an iterative negotiation-based performance driven Pathfinder router is proposed which is based on the maze routing algorithm. Allocation of routing resources is decided on the basis of demand for the resources by the nets. Nets are allowed to share routing resources initially, but subsequently must negotiate with other nets to determine which net needs the shared resource most. This is implemented by a process of ripping out and rerouting each net which is guided by a cost function that assigns a cost to every routing resources based on its demand. VPR [4] is a popular and versatile FPGA routing and placement tool. The VPR router can do global routing or a combined global and

* Corresponding author.

http://dx.doi.org/10.1016/j.mejo.2015.05.003
0026-2692/© 2015 Elsevier Ltd. All rights reserved.

detailed routing. The routing algorithm used in VPR is an advancement on the iterative Rip-up and Reroute based maze routing used by Pathfinder. The latest VPR versions can work on heterogeneous and three dimensional (TPR) [1] FPGAs also. Gambit [12] is a tool for simultaneous placement and routing in which detailed routes are obtained by a Graph Colouring approach which is then solved by a sequential heuristic-based method to obtain reduced channel widths. SEGA [13] uses a Colour Graph Expansion (CGE) algorithm to expand 2-pin nets into a number of distinct net segments along with a cost function structure to make use of long segments to obtain a good routing solution. ROAD [2] is a sequential detailed router that finds routing solutions using a Bump & Refit (B & R) approach instead of a Ripup & Reroute (R&R) approach. In ROAD global routes of prior-routed nets are not changed but their assignments are altered in order to make space for the current net to be routed. ROAD-HOP [3] is a hop-based FPGA detailed routing technique which also follows the Bump & Refit (B & R) approach. All these approaches implement sequential routing and rely on various heuristics to overcome the net ordering problem. The FPGA routing tutorial [8] has a comparison of different FPGA routing techniques.

Boolean Satisfiability (SAT) based routing approaches route all nets simultaneously and are, hence, immune to the net ordering problem. However, this naturally leads to a huge increase in the complexity of the problem and SAT-based techniques have traditionally suffered from performance and scalability related issues. Several SAT-based techniques for detailed routing have been proposed earlier. Devadas was the first to formulate the routing problem—conventional 2-layer channel routing—as an equivalent SAT problem [6]. Wood and Rutenbar [23] extended the ideas in [6] to propose a Boolean formulation for routability estimation. The Boolean function is expressed as a binary decision diagram (BDD) which represents all possible routes for all nets simultaneously. In [16] authors have proposed a *route-based* FPGA detailed routing approach using Boolean SAT and analyze the performance improvement over a previously proposed *track-based* approach [18]. An extension of the classical 2-layer routing in [6] for segmented channel routing using SAT for row based FPGAs was proposed in [9]. In [19], the authors propose a SAT-based methodology for detailed routing using the graph colouring approach for 2-pin net routes where graph colouring is represented as a Constraint Satisfaction Problem (CSP) and the variables are represented using if–then–else structures. Hung et al. [10] present a SAT-based routability checking approach for three dimensional circuit layouts where additional constraints are created to permit nets originating from the same multi-pin net to share the same wire in a channel block.

In all the SAT-based techniques discussed above, routing has been done on two-pin nets for simplicity and ease of modelling. However, most nets in a real circuit are multi-pin rather than two-pin. In this paper, we explore SAT-based FPGA detailed routing on multi-pin nets which is suitable for real multi-pin net structures and dogleg-free routing. Multi-pin net routing removes the overhead of net decomposition. Moreover multi-pin net routing automatically eliminates pin doglegging [4,7]. Two-pin net detailed routing formulations may allow pin doglegging for both input and output pins where a net is routed on more than one track segment per logic pin in the same channel as shown in Fig. 1(a). However, many commercial FPGA architectures do not permit pin dogleg for input pins where logic pins are connected to net segments via multiplexers (Fig. 1(b)). In such cases, 2-pin net formulations would require extra constraints to be added to prevent pin dogleg.

Detailed routing is also significantly affected by the architecture of the switching blocks that lie at the junction of channels. We have experimented with routing with different architectures of switching blocks architectures and have shown that the more

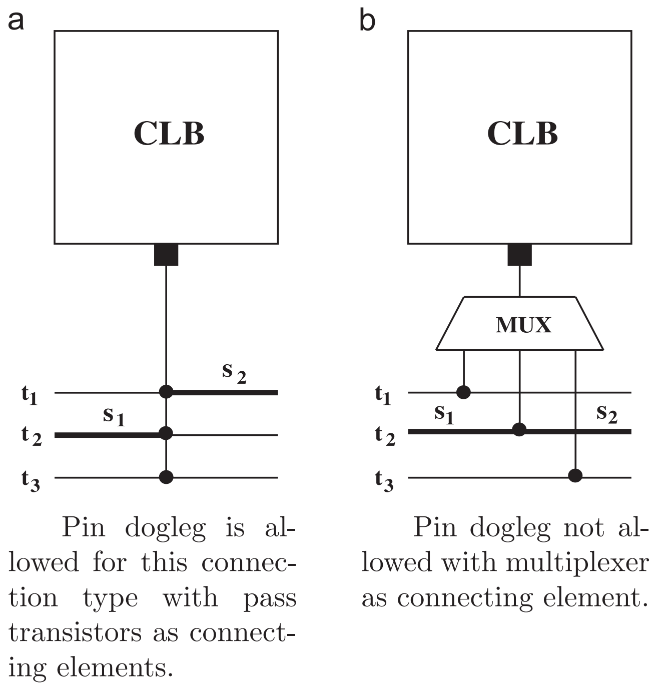
a
CLB
t_1
t_2
t_3
s_1
s_2
Pin dogleg is allowed for this connection type with pass transistors as connecting elements.
b
CLB
MUX
t_1
t_2
t_3
s_1
s_2
Pin dogleg not allowed with multiplexer as connecting element.

Fig. 1. The pin dogleg problem and its solutions. (a) Pin dogleg is allowed for this connection type with pass transistors as connecting elements. (b) Pin dogleg is not allowed with multiplexer as connecting element.

flexible switching block architectures give better results when compared to more restricted architectures. The restricted architecture of a switching block makes the problem unroutable or increases the number of tracks needed for routing. We have made a comparison between SAT-based multi-pin net and two-pin net routing and shown improvement of multi-pin net formulation over a two-pin net formulation in terms of number of Boolean variables and clauses required to formulate the Boolean routing function. Although the size minimization of the Boolean function is not the main goal of the routing problem, it helps to formulate a comparatively smaller function for a large circuit as the scalability of the solution improves. In many cases decreasing the number of tracks in a channel is not the only point to be considered in routing, rather, proving routability of all nets in a circuit under the given FPGA architecture is the prime focus. SAT-based techniques have the advantage of proving unroutability in a definite manner if the given circuit and the targeted architecture happen to be incompatible. This is a feature that non-SAT-based routers do not have.

In this paper, we propose two SAT-based approaches: *M*ulti *P*in *N*et *T*rack *E*ncoding (*MPNTE*) with different switching architectures and *M*ulti *P*in *N*et *G*raph *C*olouring *T*echnique (*MPNGCT*) which is a simplified formulation for detailed routing with *Subset* type switching architecture. MPNGCT is based on the graph node colouring paradigm with the restricted architecture of a *Subset* switching block (i.e., a net in a track can connect to the same track on the other three sides of a switching block), with the main objective being to prove the routability of a large circuit in a specified FPGA architecture.

We have formulated the detailed routing model based on the standard island style FPGA architecture layout [24,4] with a two dimensional array of configurable logic blocks (CLBs), connection blocks (CBs) and switching blocks (SBs). Three parameters channel width (W) i.e. the number of tracks in a channel, connection block flexibility (F_c) i.e. the number of tracks that each pin may connect to, and switching block flexibility (F_s) i.e. the number of other tracks that each wire segment entering an SB can connect to, define the routing capacity of the given FPGA architecture. The restrictions on the values of F_c and F_s make the detailed routing problem more complex. In other words the architectures of connection blocks and switching blocks directly affect the solution of the routing problem or more specifically, the minimum number of tracks required for complete detailed routing of a circuit. In this paper $F_c = W$ and $F_s = 3$ have been considered for formulating the Boolean routing functions.

The next section describes the multi-pin net track encoding (MPNTE) scheme. We propose SAT formulation for different switching structures and compare the detailed routing solutions obtained with the different structures. However, the most commonly used switch is the *Subset* switch and in Section 3, we propose a simplified graph-colouring based detailed routing solution for multi-pin nets (MPNGCT) which uses only *Subset* switches and shows significant improvement in both efficiency and scalability. Section 4 concludes the paper.

2. Detailed routing by track based encoding

In this section, we propose a Boolean formulation for solving the FPGA detailed routing problem using the *track-encoding* technique. The routing resources available for routing have a great impact on the routing solutions. The architecture of the switch boxes has a significant bearing on the detailed routing for island style FPGAs. For simplicity, Subset (or Disjoint) switches are used as the default switching box by most detailed routers. However, other switching box architectures may also be used.

In this section, SAT-based detailed routing formulation for multi-pin nets has been proposed with different switching box structures. *Track-encoding* is the technique which has been applied here to convert the routing problem into an equivalent SAT instance. The *track-encoding* based routing formulation named Multi-Pin Net Track Encoding (MPNTE) approach works on the segments of multi-pin nets where a segment is the longest part of a net running through a channel without changing its direction. Instead of considering an entire net at a time, MPNTE considers each segment of a net individually and creates a distinct *track-variable* for each of them. The domain of each *track-variable* is $0, 1, 2, \ldots, (W-1)$, where W is the predefined channel width based on which the Boolean function is created. This domain basically is the set of track indices in a channel. Each *track-variable* is encoded by $\lceil \log(W) \rceil$ Boolean variables. The value assigned to a *track-variable* in a satisfiable routing solution represents the track-index to which the corresponding segment is assigned. The detailed routing constraints are expressed by the following three Boolean conditions.

1. *Track assignment constraints*: They ensure that a net is assigned to any of the available tracks in a channel. For a single net segment it can be expressed in the following manner:

   $\mathit{TrackAsgCon}(\mathcal{S}_j(i)) : \bigvee_{r=0}^{W-1} (t_j(i) = r)$

   where $t_j(i)$ is the *track-variable* for the segment $\mathcal{S}_j(i)$ which is the jth segment of net n_i.
2. *Exclusivity constraints*: They restrict two distinct net segments of two different nets which are sharing at least one common connection block from being assigned to the same track. For all types of architectures the *exclusivity constraint* for any pair of distinct net segments sharing a common connection block can be expressed by the condition given below.

   $\mathit{ExclCon}(\mathcal{S}_j(i), \mathcal{S}_l(k)) : (t_j(i) \neq t_l(k))$

   where $t_j(i)$ and $t_l(k)$ are the *track-variables* for segments $\mathcal{S}_j(i)$ and $\mathcal{S}_l(k)$ of two distinct nets n_i and n_k, respectively, which reside in the same connection block.
3. *Switching constraints*: They guarantee that a net segment running through one channel can switch to an adjacent channel in a different direction when needed in keeping with the given architecture of a switch block.

The formulation of the switching constraints depends on the target architecture of the switching blocks on the FPGA, and is explained in detailed in the following subsections.

2.1. Switching structures

Three popular switching block architectures, Subset [21,13], Wilton [22] and Universal [5] are frequently used for routing in island-style FPGAs. Though the switching block flexibility ($F_s = 3$) is same for all three switches, the internal structures are different. Depending on the desired routing goal and FPGA architecture, different switches are used in different situations. The architecture of each switching block is shown in Fig. 2, where dotted lines indicate the potential connections which can be made using a particular switching block. Due to the topological variations of internal connections in switching blocks, a variation in routing results can be observed for a same circuit.

Fig. 2(a) shows that *Subset* switching block has a symmetric switching pattern i.e. each track numbered "*i*" on any side of the switch box can be connected to track numbered "*i*" only on the

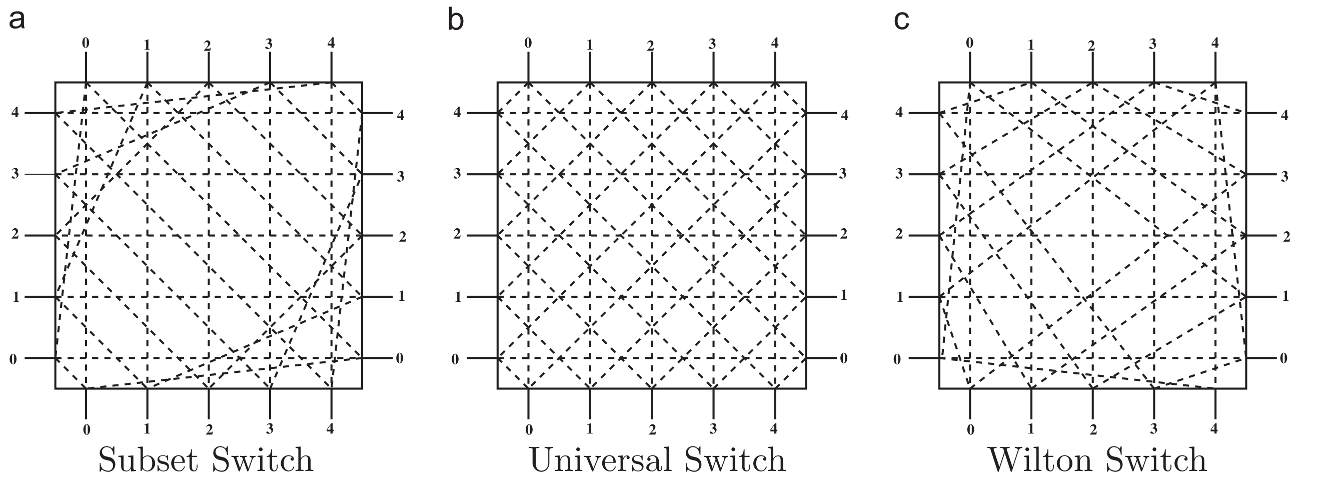

Fig. 2. Switch box architectures. (a) Subset switch. (b) Universal switch. (c) Wilton switch.

other three sides. This means that the routing fabric is divided into domains; if a wire is implemented using track "*i*", all segments that implement that wire are restricted to track "*i*". For example, an incoming signal to switch block at track 1 in Fig. 2(a) can only connect to track 1 in adjacent channels. This partitions the routing fabric into *W* subsets, and thus results in reduced routability or more channel width compared to other switch blocks.

In the *Universal* [5] block, the focus is on maximizing the flexibility of routing by allowing a net to switch from one track (index) to another. In Fig. 2(b) it is shown that a signal coming to track 0 can go out through track 0 at two sides but through track 4 at third side unlike in a *Subset* switch block.
The *Wilton* switch block is same as *Subset*, except that each diagonal connection has been rotated by one track [22]. This reduces the domains problem and results in many more routing choices for each connection. For example, an incoming signal to switch block at track 1 in Fig. 2(c) can connect to either track 1 or track 4 in adjacent channels. An increase in routing choices for a signal greatly helps in the presence of a contention during the routing process because if track 1 is already being used by another net then the track 4 could be used to make the required connections. This is no longer true if the *Subset* block is used, and the routing may fail. This makes routing with *Wilton* switch block more flexible than the *Subset* switch block.

The switch blocks were developed and evaluated for FPGA architectures consisting of only single-length wires/tracks i.e. non-segmented architectures. Real FPGAs, however, typically employ longer segments which connect distant switch blocks with less effective capacitive loading resulting in FPGAs with improved speeds. The longer segments have less effective capacitance because they need to pass through fewer switches compared to short segments, and the delay of a net increases quadratically with the number of switches that it passes through. All of the existing switch blocks can be used in segmented architectures, because for all types of switches, when a net travels through one vertical or horizontal channel the net does not change its track. Hence explicit switching is not required. Switching is needed when a net changes its direction from vertical to horizontal or vice versa.

2.2. Routing With subset Wilton and universal switches

This section describes the formulation of the *switching constraints* for various switching block structures. These constraints define which track will be assigned to a net when it changes direction. For a multi-terminal net, more than one *switching constraint* is formulated if the net changes direction more than once in its route.

The internal structure of a *Subset* switch is shown in Fig. 2(a). The internal structure of this switch restricts all segments of a net to be placed in the same track index in different channels. The formulation of *switching constraint* for this switch can be expressed in the following way using *track-variables*.

$$SwitchCon_{sub}(\mathcal{S}_j(i), \mathcal{S}_{j+1}(i)) : (t_j(i) = t_{j+1}(i))$$

where $t_j(i)$ and $t_{j+1}(i)$ are *track-variables* for the *j*th and $(j+1)$ th segments of net n_i and the net is switched from segment *j* to segment $(j+1)$.

The *Wilton* switch has a different type of internal structure. Fig. 2(c) shows the structure which allows segments of a net to be assigned to different tracks in their respective channels. The *switching constraints* for this switch are formulated based on the expression given below:

$$\bigvee_{i=0}^{W-1} \big[(t_{left} = i \wedge t_{right} = i) \vee (t_{top} = i \wedge t_{bottom} = i) \vee (t_{left} = i \wedge t_{top} = (W-i) \mathrm{mod}\ W) \vee (t_{top} = i \wedge t_{right} = (i+1) \mathrm{mod}\ W) \vee (t_{right} = i \wedge t_{bottom} = (2W-2-i) \mathrm{mod}\ W)) \vee (t_{bottom} = i \wedge t_{left} = (i+1) \mathrm{mod}\ W) \big]$$

where $t_{left}, t_{right}, t_{top}$ and t^{bottom} are the *track-variables* for some segment of a net with respect to the side of a switch where the segment is incident, as shown in Fig. 3. The above expression shows how two segments at different sides of a switch are connected and the possible assignment values of the corresponding *track-variables* expressed with respect to the sides of a switch box. The expression is a general form for *Wilton* switching and true for any value *W*.

The *switching constraints* using *track-variables* for the *Universal* switches can be constructed based on the expression given below:

$$\bigvee_{i=0}^{W-1} \big[(t_{left} = i \wedge t_{right} = i) \vee (t_{top} = i \wedge t_{bottom} = i) \vee (t_{left} = i \wedge t_{top} = (W-i-1)) \vee (t_{top} = i \wedge t_{right} = i) \vee (t_{right} = i \wedge t_{bottom} = (W-i-1)) \vee (t_{bottom} = i \wedge t_{left} = i) \big]$$

This is a general expression for connecting two segments of a net at two different sides of a *Universal* switch.

Example 1: The formulations of these constraints are clearly explained with a suitable example in this section. In Fig. 4, three

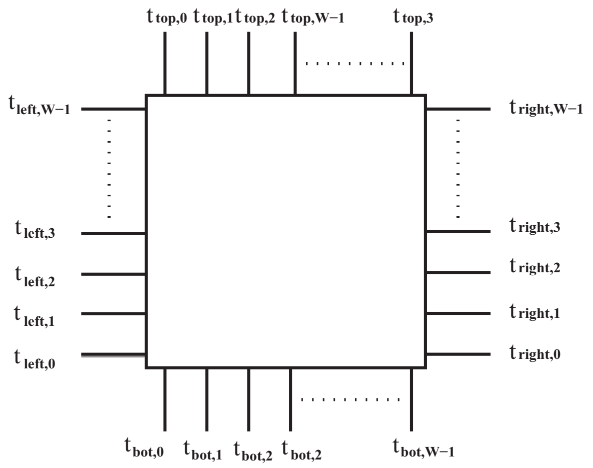

Fig. 3. General block diagram of a switch with track indices at four sides. $t_{x,i}$ denotes the *i*th track on side *x* of a switch.

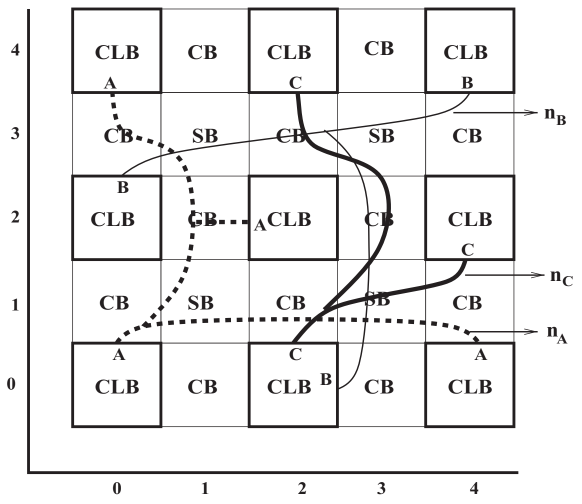

Fig. 4. Example of three multi-pin nets n_A, n_B and n_C with their global route through different routing channels.

Table 1
Track-variables for the net segments of the multi-pin nets shown in Fig. 4.

Net	Seg. No.	Seg. Sym.	Span	Track variable
	0	$S_0(A)$	C-Block(0,1)–C-Block(4,1)	$t_0(A)$
n_A	1	$S_1(A)$	C-Block(1,2)–C-Block(1,2)	$t_1(A)$
	2	$S_2(A)$	C-Block(0,3)–C-Block(0,3)	$t_2(A)$
	0	$S_0(B)$	C-Block(0,3)–C-Block(4,3)	$t_0(B)$
n_B	1	$S_1(B)$	C-Block(3,2)-C-Block(3,0)	$t_1(B)$
	0	$S_0(C)$	C-Block(2,1)–C-Block(4,1)	$t_0(C)$
n_C	1	$S_1(C)$	C-Block(3,2)–C-Block(3,2)	$t_1(C)$
	2	$S_2(C)$	C-Block(2,3)–C-Block(2,3)	$t_2(C)$

multi-terminal nets n_A, n_B, and n_C are routed by global router in the specified channels. Table 1 shows the various segments of the nets and the corresponding *track-variables* created for them. The *track assignment constraints* for net n_A, n_B, and n_C are expressed as follows assuming $W=5$.

$$TrackAsgCon(A): \bigvee_{r=0}^{4} (t_0(A)=r)$$

$$TrackAsgCon(B): \bigvee_{r=0}^{4} (t_0(B)=r)$$

$$TrackAsgCon(C): \bigvee_{r=0}^{4} (t_0(C)=r)$$

The Boolean formulation of *track assignment constraints* is shown below. Assume that *track-variable* $t_0(A)$ is encoded by three Boolean variables $b_{00}(A), b_{01}(A)$ and $b_{02}(A)$ (because $\lceil \log(W) \rceil = 3$). Therefore the Boolean formulation of this constraint is as below.

$$\begin{aligned}
TrackAsgCon(A): &\Big[\overline{b_{02}(A)b_{01}(A)b_{00}(A)} \vee \overline{b_{02}(A)b_{01}(A)}b_{00}(A) \\
&\vee \overline{b_{02}(A)}b_{01}(A)\overline{b_{00}(A)} \vee \overline{b_{02}(A)}b_{01}(A)b_{00}(A) \vee b_{02}(A)\overline{b_{01}(A)b_{00}(A)}\Big] \\
&= \overline{\Big[b_{02}(A)\overline{b_{01}(A)}b_{00}(A) \wedge b_{02}(A)b_{01}(A)\overline{b_{00}(A)} \wedge b_{02}(A)b_{01}(A)b_{00}(A)\Big]} \\
&= \Big[\Big(\overline{b_{02}(A)} \vee b_{01}(A) \vee \overline{b_{00}(A)}\Big) \wedge \Big(\overline{b_{02}(A)} \vee \overline{b_{01}(A)} \vee b_{00}(A)\Big) \\
&\wedge \Big(\overline{b_{02}(A)} \vee \overline{b_{01}(A)} \\
&\vee \overline{b_{00}(A)}\Big)\Big]
\end{aligned}$$

SB(1,1) in Fig. 4, best describes the *switching constraints* for different types of switch blocks for the given problem. At SB(1,1) net A switches from the horizontal connection block CB(0,1) to the vertical connection block CB(1,2). To show the differences in the formulations of the *switching constraints* for different switching structures we take only one switch block SB(1,1) as an example. We have given two forms of the *switching constraints* using *track-variables* and corresponding Boolean variables. To represent the Boolean formulations of this constraint, we assume that the *track-variables* $t_0(A)$ and $t_1(A)$ for the net segments through CB(0,1) and CB(1,2) are encoded by Boolean variables as $[b_{02}(A)b_{01}(A)b_{00}(A)]$ and $[b_{12}(A)b_{11}(A)b_{10}(A)]$, respectively. The formulation of the *switching constraint* for net A at switch SB(1,1) for different switches is as follows.

Subset switch: The *switching constraint* formulation using *Subset* switch defines that the values assigned to $t_0(A)$ and $t_1(A)$ should be equal within the range $[0-(W-1)]$ and can be expressed as given below:

$$SwitchCon_{sub}(\mathcal{S}_0(A), \mathcal{S}_1(A)) : [t_0(A) = t_1(A)]$$

The Boolean representation of this constraint is as follows:

$$\begin{aligned}
&SwitchCon_{sub}(\mathcal{S}_0(A), \mathcal{S}_1(A)): \\
&[(b_{00}(A) \odot b_{10}(A)) \wedge (b_{01}(A) \odot b_{11}(A)) \wedge (b_{02}(A) \odot b_{12}(A))] \\
&= \Big[\Big(\overline{b_{00}(A)} \vee b_{10}(A)\Big) \wedge \Big(b_{00}(A) \vee \overline{b_{10}(A)}\Big) \wedge \Big(\overline{b_{01}(A)} \vee b_{11}(A)\Big) \\
&\wedge \Big(b_{01}(A) \vee \overline{b_{11}(A)}\Big) \wedge \Big(\overline{b_{02}(A)} \vee b_{12}(A)\Big) \wedge \Big(b_{02}(A) \vee \overline{b_{12}(A)}\Big)\Big]
\end{aligned}$$

where $\odot$ is a Boolean logical equivalence symbol.

Wilton switch: In Fig. 4, the horizontal segment $\mathcal{S}_0(A)$ of net A is incident to the left of SB(1,1) and exits from the top of SB(1,1). The internal connection topology of a *Wilton* switch decides which tracks the segments $\mathcal{S}_0(A)$ and $\mathcal{S}_1(A)$ may be assigned to. The possible track assignments, represented as ordered pairs (left, top) are (0,0), (1,4), (2,3), (3,2) and (4,1)

$$\begin{aligned}
&SwitchCon_{wil}(\mathcal{S}_0(A), \mathcal{S}_1(A)): \\
&\quad[(t_0(A)=0 \wedge t_1(A)=0) \vee (t_0(A)=1 \wedge t_1(A)=4) \\
&\quad\vee (t_0(A)=2 \wedge t_1(A)=3) \vee (t_0(A)=3 \wedge t_1(A)=2) \\
&\quad\vee (t_0(A)=4 \wedge t_1(A)=1)]
\end{aligned}$$

The corresponding Boolean formulation of the above expression is given below:

$$\begin{aligned}
&SwitchCon_{wil}(\mathcal{S}_0(A), \mathcal{S}_1(A)): \\
&\Big[\Big(\overline{b_{02}(A)} \wedge \overline{b_{01}(A)} \wedge \overline{b_{00}(A)} \wedge \overline{b_{12}(A)} \wedge \overline{b_{11}(A)} \wedge \overline{b_{10}(A)}\Big) \\
&\vee \Big(\overline{b_{02}(A)} \wedge \overline{b_{01}(A)} \wedge b_{00}(A) \wedge b_{12}(A) \wedge \overline{b_{11}(A)} \wedge \overline{b_{10}(A)}\Big) \\
&\vee \Big(\overline{b_{02}(A)} \wedge b_{01}(A) \wedge \overline{b_{00}(A)} \wedge \overline{b_{12}(A)} \wedge b_{11}(A) \wedge b_{10}(A)\Big) \\
&\vee \Big(\overline{b_{02}(A)} \wedge b_{01}(A) \wedge b_{00}(A) \wedge \overline{b_{12}(A)} \wedge b_{11}(A) \wedge \overline{b_{10}(A)}\Big) \\
&\vee \Big(b_{02}(A) \wedge \overline{b_{01}(A)} \wedge \overline{b_{00}(A)} \wedge \overline{b_{12}(A)} \wedge \overline{b_{11}(A)} \wedge b_{10}(A)\Big)\Big]
\end{aligned}$$

Universal switch: Similarly, the expression for the *switching constraints* using *Universal* is given below. The possible track assignment for the segments $\mathcal{S}_0(A)$ and $\mathcal{S}_1(A)$ represented pair wise is (0, 4), (1, 3), (2, 2), (3, 1) and (4, 0).

$$\begin{aligned}
&SwitchCon_{univ}(\mathcal{S}_0(A), \mathcal{S}_1(A)): \\
&\quad[(t_0(A)=0 \wedge t_1(A)=4) \vee (t_0(A)=1 \wedge t_1(A)=3) \vee (t_0(A)=2 \wedge t_1(A) \\
&\quad=2) \vee (t_0(A)=3 \wedge t_1(A)=1) \vee (t_0(A)=4 \wedge t_1(A)=0)]
\end{aligned}$$

$SwitchCon_{univ}(\mathcal{S}_0(A), \mathcal{S}_1(A))$ can be expressed in the form of a Boolean expression as given below:

$$\begin{aligned}
&SwitchCon_{univ}(\mathcal{S}_0(A), \mathcal{S}_1(A)): \\
&\Big[\Big(\overline{b_{02}(A)} \wedge \overline{b_{01}(A)} \wedge \overline{b_{00}(A)} \wedge b_{12}(A) \wedge \overline{b_{11}(A)} \wedge \overline{b_{10}(A)}\Big) \\
&\vee \Big(\overline{b_{02}(A)} \wedge \overline{b_{01}(A)} \wedge b_{00}(A) \wedge \overline{b_{12}(A)} \wedge b_{11}(A) \wedge b_{10}(A)\Big) \\
&\vee \Big(\overline{b_{02}(A)} \wedge b_{01}(A) \wedge \overline{b_{00}(A)} \wedge \overline{b_{12}(A)} \wedge b_{11}(A) \wedge \overline{b_{10}(A)}\Big) \\
&\vee \Big(\overline{b_{02}(A)} \wedge b_{01}(A) \wedge b_{00}(A) \wedge \overline{b_{12}(A)} \wedge \overline{b_{11}(A)} \wedge b_{10}(A)\Big) \\
&\vee \Big(b_{02}(A) \wedge \overline{b_{01}(A)} \wedge \overline{b_{00}(A)} \wedge \overline{b_{12}(A)} \wedge \overline{b_{11}(A)} \wedge \overline{b_{10}(A)}\Big)\Big]
\end{aligned}$$

Boolean formulation of *switching constraint* for different switching block structure is shown above. These formulations are converted to CNF and conjuncted with other clauses to form the complete function. The *exclusivity constraints* are required to be formulated among nets (A,B), (B,C) and (A,C) pair wise, according to Figs. 4 and 6. The formulation of the *exclusivity constraint* between net A and B can be explained with the help of Fig. 4. Both the nets are sharing CB(0,3) as a common routing region. The corresponding *track-variables* for the segments $\mathcal{S}_2(A)$ and $\mathcal{S}_0(B)$ running through CB(0,3) are $t_2(A)$ and $t_0(B)$ as listed in Table 1. The formulation of the constraint can be expressed as follows.

$$\begin{aligned}
&ExclCon(\mathcal{S}_2(A), \mathcal{S}_0(B)) : [t_2(A) \neq t_0(B)] \\
&\quad = \neg\left[\bigvee_{i=0}^{4}(t_2(A) = i \wedge t_0(B) = i)\right] \\
&\quad = \left[\bigwedge_{i=0}^{4}\neg(t_2(A) = i \wedge t_0(B) = i)\right] \\
&\quad = \left[(b_{20}(A) \vee b_{21}(A) \vee b_{22}(A) \vee b_{00}(B) \vee b_{01}(B) b_{02}(B))\right] \wedge \left[\left(\overline{b_{20}(A)} \vee b_{21}(A) \vee \right.\right. \\
&\quad \left.\left. \vee\, b_{22}(A) \vee \overline{b_{00}(B)} \vee b_{01}(B) \vee b_{02}(B)\right)\right] \wedge \left[\left(b_{20}(A) \vee \overline{b_{21}(A)} \vee b_{22}(A) \vee b_{00}(B) \right.\right. \\
&\quad \left.\left. \vee\, \overline{b_{01}(B)} \vee b_{02}(B)\right)\right] \wedge \left[\left(\overline{b_{20}(A)} \vee \overline{b_{21}(A)} \vee b_{22}(A) \vee \overline{b_{00}(B)} \vee \overline{b_{01}(B)} \vee b_{02}(B)\right)\right] \\
&\quad \wedge \left[\left(b_{20}(A) \vee b_{21}(A) \vee \overline{b_{22}(A)} \vee b_{00}(B) \vee b_{01}(B) \vee \overline{b_{02}(B)}\right)\right] \wedge \left[\left(\overline{b_{20}(A)} \vee b_{21}(A) \right.\right. \\
&\quad \left.\left. \vee\, \overline{b_{22}(A)} \vee \overline{b_{00}(B)} \vee b_{01}(B) \vee \overline{b_{02}(B)}\right)\right]
\end{aligned}$$

where $t_2(A)$ and $t_0(B)$ are encoded by the Boolean variables $b_{20}(A), b_{21}(A), b_{22}(A)$ and $b_{00}(B), b_{01}(B), b_{02}(B)$.

An iterative process is used to calculate the minimum channel width W_{min} with three different classes of constraints required to create the *track-based* formulation of the detailed routing problem. This process is shown in the flowchart given in Fig. 5. The value for I shown in the flowchart is obtained from global routing information for the circuit under consideration for routing.

2.3. Experimental results

The proposed SAT-based *dogleg-free* detailed routing formulation, MPNTE on multi-pin nets has been experimented with using the standard MCNC benchmark circuits. We have used VPR [20] for placement and global routing for all the circuits. Along with a circuit description in .blif format, a special FPGA architecture file in .arch form is fed into VPR which produces global routes for all the nets in the circuit in .r format. The relevant properties of these circuit are listed in Table 2. The table summarizes the size of the circuits as a two dimensional array structure ("X × Y"), actual number of CLBs used ("# C"), the number of multi-pin nets ("# N"), and the total number of distinct net segments ("# NS"). A segment of a net is the longest part of multi-pin net running through a channel without changing its direction. We construct the routing formulations on the multi-pin nets obtained from the global routing results produced by VPR.

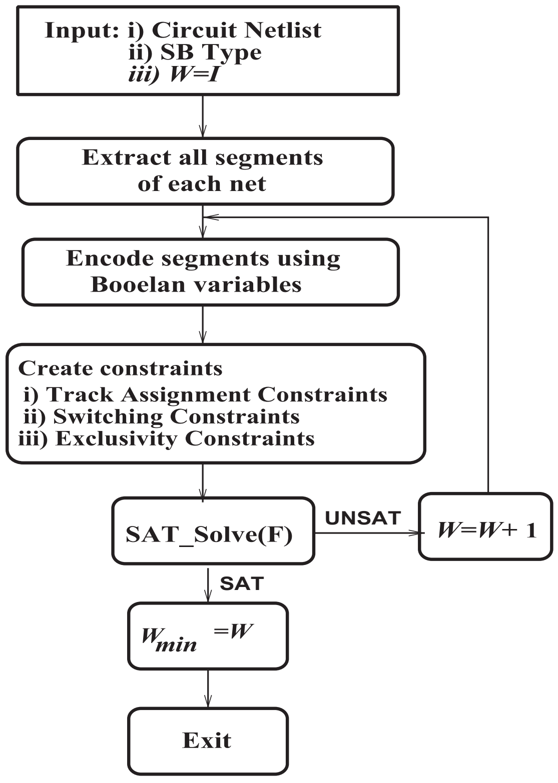
Input: i) Circuit Netlist
ii) SB Type
iii) W=I
Extract all segments of each net
Encode segments using Booelan variables
Create constraints
i) Track Assignment Constraints
ii) Switching Constraints
iii) Exclusivity Constraints
SAT_Solve(F)
UNSAT
W=W+ 1
SAT
Wmin =W
Exit

Fig. 5. Flowchart describing the working principle of MPNTE.

Table 2
MCNC benchmark circuits. The size of the circuit is defined as a 2-dimensional array structure ($X \times Y$), the actual number of CLBs used (#C), the number of multi-pin nets (#N), and total distinct net segments (#NS) (A segment of a net is the longest part of multi-pin net running through a channel without changing its direction).

Circuit	X × Y	#C	#N	#NS
9symml	10 × 10	97	106	236
alu2	15 × 15	197	207	512
apex7	11 × 11	102	151	324
C499	10 × 10	74	115	296
C880	14 × 14	174	234	540
example2	19 × 19	120	223	536
C1355	10 × 10	74	115	376
term1	10 × 10	88	122	276
too-lrg	14 × 14	187	225	568
vda	18 × 18	291	308	812
k2	23 × 23	519	564	2302
e64	17 × 17	274	339	910
9sym	12 × 12	144	153	432
misex3c	24 × 24	549	563	1882
alu4	40 × 40	1522	1536	5583
bigkey	54 × 54	1707	1936	6604
des	63 × 63	1591	1847	7865

The technique described above is implemented using C and Zchaff [25] as the underlying SAT solver. All experiments were conducted on a Core2Duo Fedora 12 system with 8 GB RAM. Using the final global routing solution produced by VPR, a Boolean function is formulated and solved by Zchaff in an iterative way on different values of W till the minimum value W_{min} is obtained which causes the formulated Boolean function to evaluate to TRUE and hence gives a complete detailed routing solution for all the multi-pin nets in the circuit.

The detailed routing constraints for MPNTE are constructed based on the different switch box architectures separately. In all cases, switching structures are given for switching block flexibility $F_s = 3$ and connection block flexibility $F_c = W$. The Boolean formulation of the constraints depends on a particular value of W which is the channel width under consideration for finding routing solution for a circuit. The initial value of W for the first iteration is obtained from the global routing information produced by VPR. VPR uses a routing procedure in which the global routes are changed adaptively in order to get a good final detailed routing. Hence, we have extracted the global routing information from the final detailed routes given by VPR. The value of W is changed iteratively until we get the minimum value which causes the function to evaluate to TRUE and gives a complete detailed routing solution for a circuit.

The experimental routing results obtained by MPNTE are listed in Table 3. The table shows the routing results for the benchmark circuits along with the running time in seconds required for each case. For every switch type, switching block flexibility is 3, but routing solutions are different. The difference in channel width is due to the overall flexibility of the switches. *Subset* switch is more restricted than the other two switches. It restricts all segments of a net to be assigned to the same track for the entire span of the net in different channels. *Wilton* switch is more flexible than the other switches and hence it provides better results in terms of the channel width required for routing.

Table 3
Minimum channel width (W_{min}) obtained for detailed routing with different switching architectures.

Circuit	Subset		Wilton		Universal	
	W	CPU time	W	CPU time	W	CPU time
9sym	6	0.06	5	0.07	6	0.12
9symml	5	0.04	5	0.02	5	0.05
apex7	5	0.02	4	0.02	5	0.05
alu2	7	6.12	6	11.50	6	9.17
C499	7	1.32	6	1.73	6	1.59
C880	7	0.97	6	0.86	7	2.12
C1350	5	1.12	5	3.46	5	7.20
example2	6	0.56	5	1.23	5	0.95
term1	5	0.18	4	0.52	5	0.72
too-lrg	8	2.53	7	2.13	7	6.63
vda	9	2.58	8	4.27	9	6.51
k2	10	12.32	9	17.22	10	15.90
e64	8	2.60	8	3.41	8	7.25
alu4	10	4121.13	10	6319.27	10	5941.30
misex3c	11	12.13	10	71.50	10	43.12

3. Graph colouring for detailed routing by SAT

Even though, all the three switching architectures described in the previous section are available, the *Subset* switch is, by far, the most commonly used in actual implementations, in spite of being the least flexible architecture. In this section, we propose another Boolean formulation for solving the FPGA detailed routing problem using *Subset* switching only. This technique is based on the graph node colouring concept and is more compact and scalable than MPNTE. Hence, the objective of the proposed technique is to implement the detailed routing technique when *Subset* switches are the only available option such that bigger circuits can be handled.

The graph colouring problem is a well-known NP-complete problem and can be solved by Boolean satisfiability. In this proposed approach, the detailed routing problem is first converted into a graph colouring problem which is then solved by SAT. This technique is called Multi-Pin Net Graph Colouring Technique (MPNGCT). Any valid satisfying assignment of Boolean variables in the function assigns a specific colour to each node of the graph such that no two adjacent nodes are assigned the same colour. The minimum number of colours needed to colour the entire graph is the *Chromatic Number* ($\mathbb{C}_{min}$). In this approach, different nets which share the same connection block are represented by adjacent nodes and must be assigned different colours (tracks) to maintain electrical disconnection. A solution to the graph colouring problem implicitly provides a solution to the routing problem which assigns each net to a distinct track in the FPGA layout and the $\mathbb{C}_{min}$ defines the minimum channel width (W_{min}) for complete detailed routing for the entire circuit.

The Multi-Pin Net Graph Colouring Technique (MPNGCT) for detailed routing is scalable for large circuits because it uses smaller number of variables and simpler CNF clauses when compared to other SAT-based routing techniques. The size of the Boolean function generated by MPNGCT for a circuit is smaller in comparison with MPNTE and other SAT-based detailed routing approaches w.r.t. the number of CNF clauses and total number of literals. This technique uses *Subset* switches implicitly and compels each segment of a multi-pin net to be placed on the same track index in the channels through which it passes. This restriction on net segments decreases the flexibility of routing.

This technique is formulated for an FPGA architecture with switching block flexibility $F_s = 3$ with *Subset* switch and the connection block flexibility $F_c = W$. The conversion of FPGA detailed routing problem into a graph colouring problem is done by the two-step recursive process as detailed below:

1. *Node creation*: Create a distinct node for each individual multi-pin net in the circuit.
2. *Edge creation*: Create an edge between two nodes when their corresponding multi-pin nets are overlapping in any channel in their span.

This transformation is shown by the example given below:

Example 2: The *GraphCreation* process can be clearly illustrated using the circuit given in Fig. 4. In the figure, three multi-pin nets n_A, n_B, and n_C are shown with their global routes. Three nodes labelled *A*, *B*, and *C* have been created for nets n_A, n_B, and n_C respectively. The edges AB, BC, and AC have been added to the graph because of sharing of CB(0,3) between net n_A and n_B, CB(3,2) between net n_B and n_C and CB(2,1) between n_A and n_C respectively. The resultant graph is shown in Fig. 6.

Algorithm 1. *ConstraintCreation.*

```
Input: Graph $G(V,E)$, Set $\mathbb{C} = I$
Output: Chromatic Number $\mathbb{C}_{min}$(Minimum channel width
    ($W_{min}$))
1: Boolean function $\mathcal{F} = \Phi$
2: $\mathcal{B} = \lceil \log(\mathbb{C}) \rceil$
3: $\forall v_i \in V$, create $\mathcal{B}$ Boolean variables $b_0(i), b_1(i), \ldots, b_{\mathcal{B}-1}(i)$
4: $\forall v_i \in V$, construct and AND with $\mathcal{F}$ a set of CNF clauses for
    $ColAsgCon(v_i)$ using Boolean variables for $v_i$
5: $\forall e_{ij} \in E$ construct and AND with $\mathcal{F}$ a set of CNF clauses for
    $AdjColCon(v_i, v_j)$ using Boolean variables for $v_i$ and $v_j$
6: if (SAT($\mathcal{F}$)) goto(8)
7: $\mathbb{C} = \mathbb{C} + 1$ goto(1)
8: $\mathbb{C}_{min} = \mathbb{C}$
9: Exit
```

In the next phase, the nodes of the created graph are assigned colours with minimum possible number of colours which is determined using Boolean satisfiability. To formulate the graph colouring problem as a SAT problem, a colour-variable C_v is created for each node v of the graph. Each colour-variable may have any value in the domain $\{0, 1, \ldots, (\mathbb{C}-1)\}$ where $\mathbb{C}$, the predefined number of colours is used to paint all the nodes of the graph. The value of $\mathbb{C}$ is changed iteratively until $\mathbb{C}_{min}$ is obtained. The solution for colouring the nodes of the graph follows two important constraints which can be defined as follows:

1. *Colour assignment constraint:* Each node should be coloured by a valid available colour. For example, if C_x, a colour-variable has

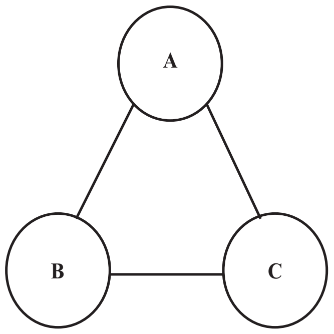

Fig. 6. Graph colouring example for the global routing given in Fig. 4. Three nodes *A*, *B*, and *C* have been created for three multi-pin nets n_A, n_B, and n_C respectively. The edges have been created according to the *adjacency colour constraints*.

Table 4
Constraints formulas and their CNF representations for graph colouring of the problem in Fig. 6. Three colour-variables C_a, C_b, and C_c have been created.

Constraint	Node (s)	Formula	CNF
ColAsgCon()	A	$(C_a=0 \vee C_a=1 \vee C_a=2)$	$[\overline{b_0(A)} \vee \overline{b_1(A)}]$
	B	$(C_b=0 \vee C_b=1 \vee C_b=2)$	$[\overline{b_0(B)} \vee \overline{b_1(B)}]$
	C	$(C_c=0 \vee C_c=1 \vee C_c=2)$	$[\overline{b_0(C)} \vee \overline{b_1(C)}]$
AdjColCon( )	A, C	$(C_a \neq C_c)$	$[b_0(A) \vee b_1(A) \vee b_0(C) \vee b_1(C)]$
			$\wedge [\overline{b_0(A)} \vee b_1(A) \vee \overline{b_0(C)} \vee b_1(C)]$
			$\wedge [b_0(A) \vee \overline{b_1(A)} \vee b_0(C) \vee \overline{b_1(C)}]$
	A, B	$(C_a \neq C_b)$	$[b_0(A) \vee b_1(A) \vee b_0(B) \vee b_1(B)]$
			$\wedge [\overline{b_0(A)} \vee b_1(A) \vee \overline{b_0(B)} \vee b_1(B)]$
			$\wedge [b_0(A) \vee \overline{b_1(A)} \vee b_0(B) \vee \overline{b_1(B)}]$
	B, C	$(C_b \neq C_c)$	$[b_0(B) \vee b_1(B) \vee b_0(C) \vee b_1(C)]$
			$\wedge [\overline{b_0(B)} \vee b_1(B) \vee \overline{b_0(C)} \vee b_1(C)]$
			$\wedge [b_0(B) \vee \overline{b_1(B)} \vee b_0(C) \vee \overline{b_1(C)}]$

been created for node x and the domain of the colour value is $\{0, 1, \ldots, (d-1)\}$, the constraint can be expressed as *ColAsgCon* (x) : $\bigvee_{r=0}^{d-1} C_x = r$ where d is the number of colour under consideration to formulate the problem.

2. *Adjacency colour constraint:* The adjacent nodes should not be coloured by the same colour. We assume that two colour-variables C_x and C_y are created for two adjacent nodes x and y respectively. Therefore this constraint can be written as follows:
   AdjColCon(x, y) : $(C_x \neq C_y)$

These two types of constraints implicitly ensure the two detailed routing constraints. The *ColAsgCon*() ensures track assignment of the net for which the node has been created, to a particular available track in channel(s) throughout the length of the net. *AdjColCon*() guarantees that two electrically distinct nets overlapping in any channel along their length are assigned to different tracks for detailed routing. The entire multi-pin net is being considered as a single node of the graph and a single colour assignment to the node implicitly ensures that all the segments of the net are being assigned to same track index through out its span in different channels. In Table 4, graph colouring constraints *ColAsgCon*() and *AdjColCon*() are shown in the form of colour-variables and their corresponding CNF representations for the problem in Fig. 6. Colour-variables C_a, C_b and C_c are created for node A, B and C respectively. For the formulation of the CNF clauses and a monolithic Boolean function for the graph colouring problem, each colour-variable is encoded by $\lceil \log(\mathbb{C}) \rceil$ binary variables, where $\mathbb{C}$ (here, $\mathbb{C}=3$ assumed) is the number of colours considered for colouring the nodes of the graph. Using these binary variables each graph colouring constraint can be represented by a set of CNF clauses. The conjunction of all the constraints in the form of CNF clauses creates the complete Boolean function as follows:

$$\mathcal{F} : \left[\bigwedge_{1 \le i \le N} ColAsgCon(i) \right] \wedge \left[\bigwedge_{\forall e_{pq} \in E} AdjColCon(p, q) \right]$$

where, we assume N is the number of nodes in the graph $G=(V, E)$, e_{pq} is an edge in E between nodes p and q $(p, q \in V)$, *ColAsgCon*(i) is the CNF form of the colour assignment constraints for node i, *AdjColCon*(p, q) is the CNF form of the adjacency colour constraint for any edge e_{pq}. Algorithm 1 describes the formulation of the Boolean function $\mathcal{F}$ by constructing the set of CNF clauses for *Colour assignment constraints* and *Adjacency colour constraints*. The initial value I for $\mathbb{C}$ is obtained from the minimum track width needed for global routing for the circuit under consideration. This value is increased iteratively until a satisfiable solution is obtained which gives a complete detailed routing solution for the circuit under consideration under the constraints of the target FPGA architecture and $\mathbb{C}_{min}$ is obtained.

3.1. Experimental results

We have implemented MPNGCT technique on the benchmark circuits listed in Table 2 with the same system configuration as was used for MPNTE in Section 2.1.

In Table 5, we have shown the detailed routing results obtained by this technique with the columns headed by “W” (minimum channel width which meets routability), “Variables” (number of Boolean variables), “Clauses” (number of CNF clauses), and “CPU Time” required for total computation for various benchmark circuits.

In Table 6, we compare our proposed techniques MPNTE and MPNGCT with an existing SAT-based detailed routing formulation R-SAT [16]. The comparison is based on the number of Boolean variables and clauses required to formulate the Boolean function for the given detailed routing problem using *Subset* switching in all cases, even though R-SAT [16] is based on a 2-pin net formulation with pin dogleg, but is included here as a comparison

Table 5
Results for circuits that were routed by our Multi-Pin Net Graph Colouring Technique (MPNGCT) based detailed routing.

Circuit	W	Variables	Clauses	CPU time
9symml	5	318	2968	0.012
alu2	7	621	7452	1.76
apex7	5	453	4224	0.003
C499	7	345	4180	0.06
C880	7	936	8424	0.61
C1355	5	345	3170	0.13
example2	6	669	7136	0.02
term1	5	366	2416	0.01
too-lrg	8	900	19,124	0.11
vda	9	1232	16,076	0.71
k2	10	2256	36,864	2.51
e64	8	1356	16,272	0.86
9sym	6	459	4896	0.01
misex3c	11	2252	39,973	2.62
alu4	10	6144	101,388	2182.18
bigkey	8	5808	122,928	2852.64
des	8	5541	188,650	4302.82

Table 6
Number of variables and clauses used for SAT-based detailed routing formulations using subset switching.

Circuit	MPNTE		MPNGCT		2-Pin R-SAT	
	Var.	Cls.	Var.	Cls.	Var.	Cls.
9symml	728	6372	318	2968	1554	29,119
alu2	1536	11,253	621	7452	4080	83,902
apex7	972	7235	453	4224	1500	11,695
C499	788	6685	345	4180	1872	18,870
C880	1620	17,281	936	8424	4592	61,745
C1355	1128	8242	345	3170	NA	NA
example2	908	16,436	669	7136	2664	27,684
term1	628	10,263	366	2416	808	3299
too-lrg	1704	17,553	900	19,124	3663	50,373
vda	3248	37,642	1232	16,076	6498	130,997
k2	9208	94,832	2256	36,864	12,570	338,927
e64	2730	64,116	1356	16,272	NA	NA
9sym	1296	7437	459	4896	NA	NA
misex3c	7428	72,286	2252	39,973	NA	NA
alu4	22,332	231,388	6144	101,388	NA	NA
bigkey	NA	NA	5808	122,928	NA	NA
des	NA	NA	5541	188,650	NA	NA

with another SAT-based technique. The table demonstrates that the proposed multi-pin net detailed routing techniques MPNTE and MPNGCT use significantly smaller number of Boolean variables and clauses than the 2-pin route-based formulation. The difference in number of clauses and variables is because, in 2-pin net routing formulation, separate routing constraints are formulated for individual 2-pin nets even when they belong to the same multi-pin net, whereas in multi-pin net routing, this duplication does not appear. Smaller number of variables and clauses are the reasons for an improvement in searching time. A smaller Boolean function certainly improves the scalability of the technique. Of course, MPNGCT uses fewer variables and clauses, in general than the *track-encoding* techniques since separate track variables are not required for each net segment.

In Table 7, we compare the performances of our routers with other detailed routing techniques with respect to the minimum channel width required for complete detailed routing. The scope of the comparison is limited by the fact that most existing routers allow pin-doglegging at connection blocks which gives an advantage with respect to minimum channel width required for detailed routing [17]. The results given in this table are for dogleg free routing with *Subset* switching. The channel widths for MPNTE, MPNGCT and VPR are obtained after running them on our own system. But results for SEGA and Gambit are obtained from [4] and [12], respectively. All the routers in this table use VPR for placement and global routing. The “NA” entries under “Gambit” and “SEGA” denote that the corresponding information is not available in the referenced papers. The detailed routes obtained from VPR clearly show that it follows 2-pin net routing which allows VPR to place different segments of a multi-pin net on different track indices. This indicates that VPR uses *Subset* switching for 2-pin net structures and not for the entire multi-pin net as in our techniques. Hence, our techniques work on a more restricted architecture and this explains why they require more tracks than VPR for some circuits (alu2, C499, example2, too-lrg) even though SAT based techniques explore the complete solution space and report the best possible solution.

In Table 8, a comparison of the run times of the various routers are given. The CPU times are on the basis of experiments conducted on a Core2Duo Fedora 12 System with 8 GB RAM. To get the actual time VPR spends for detailed routing and thus to compare it with runtime of our techniques, we obtained VPR's global routing time and subtracted it from VPR's global+detailed routing time. In spite of this difference, the results (CPU time) give a fair indication that our SAT-based approaches perform on par with VPR (usually better than VPR in the case of MPNGCT) which is a popular state-of-art router available today. This is particularly significant in view of the fact that SAT-based routers have traditionally suffered from performance issues owing to the fact that all nets are to be solved simultaneously which increases the complexity of the problem.

Table 7
Minimum channel width obtained with various detailed routers. NA denotes information not available.

Circuit	Detailed router				
	MPNTE	MPNGCT	Gambit	VPR	SEGA
9symml	5	5	14	5	NA
alu2	7	7	19	6	NA
apex7	5	5	11	5	NA
C499	7	7	NA	6	NA
C880	7	7	NA	7	NA
C1355	5	5	NA	6	NA
example2	6	6	15	5	NA
term1	5	5	9	5	NA
too-lrg	8	8	18	7	NA
vda	9	9	26	9	NA
k2	10	10	NA	10	NA
e64	8	8	NA	8	NA
9sym	6	6	NA	6	NA
misex3c	11	11	NA	10	17
alu4	10	10	NA	10	16
bigkey	NA	8	NA	7	9
des	NA	8	NA	7	11

Table 8
Run time comparisons among different routers. NC denotes not completed within specified time.

Circuit	CPU time		
	MPNTE (Subset)	MPNGCT	VPR (Subset)
9symml	0.04	0.012	0.25
alu2	6.12	1.76	6.32
apex7	0.02	0.003	1.32
C499	1.32	0.06	2.89
C880	0.97	0.61	2.76
C1355	1.12	0.13	1.62
example2	0.56	0.02	1.65
term1	0.18	0.01	2.13
too-lrg	2.13	0.11	1.32
vda	2.21	0.71	1.54
k2	12.32	2.51	12.11
e64	2.60	0.86	2.55
9sym	0.06	0.01	1.87
misex3c	12.13	2.62	12.29
alu4	4121.14	2182.18	5211.12
bigkey	NC	2852.64	4464.22
des	NC	4302.82	6532.54

4. Conclusion

This paper introduces two SAT-based formulations for FPGA detailed routing for multi-pin net structures. Both these techniques avoid net decomposition in formulating the detailed routing constraints in the form of CNF clauses. The graph colouring based formulation (MPNGCT) uses fewer number of variables and clauses and hence reduces the size of the Boolean function, which improve the speed and scalability of the technique. This is particularly useful when obtaining a routable circuit is the requirement and minimizing channel width is not necessarily the primary focus. Due to its scalability, this technique is useful for proving the routability of larger benchmark circuits. The formulations for FPGA detailed routing on three different switching architectures for multi-pin net structures have also been explored. The solutions obtained show the routing capability of different switch boxes in terms of routability and minimum channel width. The *Wilton* switch gives better results than other two switching types owing primary to its greater flexibility. The techniques proposed here assume bidirectional dogleg free routing. Even though we have considered $F_s = 3$ and $F_c = W$ for our routing algorithms, both techniques can be easily modified for use with any other values of F_s and F_c.

References

[1] C. Ababei, H. Mogal, K. Bazargan, Three-dimensional place and route for fpgas, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 25 (6) (2006) 1132–1140.
[2] H. Arslan, S. Dutt, Road: an order-impervious optimal detailed router for fpgas, in: 21st Proceedings of International Conference on Computer Design (ICCD 2003), VLSI in Computers and Processors, 13–15 October 2003, San Jose, CA, USA, IEEE Computer Society, San Jose, California, 2003, pp. 350–356.
[3] H. Arslan, S. Dutt, An effective hop-based detailed router for fpgas for optimizing track usage and circuit performance, in: Proceedings of the 14th

ACM Great Lakes Symposium on VLSI 2004, Boston, MA, USA, April 26–28, 2004. ACM, Boston, Massachusetts, 2004, pp. 208–213.
[4] V. Betz, J. Rose, Vpr: a new packing and placement and routing tool for fpga research, in: Proceedings of 7th Annual Workshop Field Programmable Logic and Applications, 1997, pp. 213–222.
[5] Y. Chang, D.F. Wong, C.K. Wong, Universal switch modules for fpga design, ACM Trans. Des. Autom. Electron. Syst. 1 (1996) 80–101.
[6] S. Devadas, Optimal layout via boolean satisfiability, in: Proceedings of the International Conference on Computer-Aided Design (ICCAD), ACM/IEEE, November 1989, pp. 294–297.
[7] FPGA, Actel Family Databook, 1991.
[8] D.F. Gomez-Prado, A Tutorial on fpga Routing, 2006.
[9] W.N.N. Hung, X. Song, E.M. Aboulhamid, A. Kennings, A. Coppola, Segmented channel routability via satisfiability, ACM Trans. Des. Autom. Electron. Syst. (TODAES), October 9(4), 2004.
[10] W.N.N. Hung, X. Song, T. Kam, L. Cheng, G. Yang, Routability checking for three-dimensional architectures, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 12 (12) (2004) 1371–1374.
[11] R.J. Bayardo Jr., R. Schrag, Using csp look-back techniques to solve real world sat instances, in: Proceedings of the 14th International Conference on Artificial Intelligence, 1997, pp. 203–208.
[12] J. Karro, J. Cohoon, Gambit: a tool for the simultaneous placement and detailed routing of gate arrays, in: Proceedings of FPL 2001, LNCS 2147, Springer-Verlag, Berlin, Heidelberg, 2002, pp. 243–253.
[13] G. Lemiux, S. Brown, A detailed router for allocating wire segments in fpgas, in: Proceedings of International on Physical Design WorkShop. ACM, Lake Arrowhead, California, 1993, pp. 215–226.
[14] J.P. Marques-Silva, K. Sakallah, Grasp: a search algorithm for propositional satisfiability, IEEE Trans. Comput. 48 (5) (1999).
[15] L.E. McMurchie, C. Ebeling, Pathfinder: a negotiation-based path-driven router for fpgas, in: Proceedings of the International Symposium on FPGAs, ACM/IEEE, February 1995.
[16] G. Nam, F. Aloul, K. Sakallah, R. Rutenbar, A comparative study of two boolean formulations of fpga detailed routing constraints, IEEE Trans. Comput. 53(6), June 2004.
[17] G. Nam, K. Sakallah, R. Rutenbar, A new fpga detailed routing approach via search-based boolean satisfiability, IEEE Trans.Comput.-Aided Des. Integr. Circuits Syst. 21(6), June 2002.
[18] G.-J. Nam, K.A. Sakallah, R.A. Rutenbar, Satisfiability-based layout revisited: detailed routing of complex fpgas via search-based boolean sat, in: Proceedings of the 1999 ACM/SIGDA Seventh International Symposium on FPGAs, ACM, New York, NY, USA, 1999, pp. 167–175.
[19] M.N. Velev, P. Gao, Comparison of boolean satisfiability encodings on fpga detailed routing problems, in: Proceedings of the Conference on Design Automation and Test in Europe, DATE '08, ACM, New York, NY, USA, 2008, pp. 1268–1273.
[20] Vpr430, Vpr fpga placer and router. ⟨http://www.eecg.toronto.edu/~vaughn/vpr/vpr.html⟩, 2000.
[21] N.H.E. Weste, K. Eshraghian, Principles of CMOS VLSI design: a systems perspective (second edition), Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1992.
[22] S.J.E. Wilton, Architectures and algorithms for field-programmable gate arrays with embedded memory, Technical Report, 1997.
[23] R.G. Wood, R.A. Rutenbar, Fpga routing and routability estimationvia boolean satisfiabily, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 6 (June (2)) (1998) 222–231.
[24] Xilinx, Xilinx fpga product. ⟨http://www.xilinx.com/products/xc4000e.htm⟩, 2008.
[25] Zchaff, Sat solver zchaff. ⟨http://www.princeton.edu/~chaff/zChaff.html⟩, 2008.
[26] H. Zhang, Sato: an efficient propositional prover, in: Proceedings of the International Conference on Automated Deduction, 1997, pp. 272–275.