IEEE International Symposium on Circuits and Systems (ISCAS'1999)
Orlando, June 1999

EXACT CHANNEL ROUTING USING SYMBOLIC REPRESENTATION

Frank Schmiedle *Rolf Drechsler* *Bernd Becker*

Institute of Computer Science, Albert-Ludwigs-University, 79110 Freiburg im Breisgau, Germany
{schmiedl,drechsle,becker}@informatik.uni-freiburg.de

Abstract

The layout problem in VLSI-design can be broken up into the subtasks partitioning, placement and routing. As well as the others of these tasks, the routing problem is NP-complete and so in general it is impossible to find an optimal solution to it in an efficient way. However, for problem instances of small size, it may be possible not only to generate the best routing with respect to a certain cost function, but even to determine all *legal solutions to the problem.*

We consider the channel routing problem and show how to solve it exactly using symbolic methods i.e. MDDs (Multi-valued Decision Diagrams). The basic operations on MDDs can be carried out very efficiently and thus the necessary time-consuming computations can be speeded up considerably. In our approach, wires in the channel are described as functions of multi-valued variables, and so the grid points and also the set of legal solutions can be represented by MDD-nodes. Furthermore, routability is transformed into satisfiability of these functions. By this we can also decide if a channel is routable because satisfiablity is trivial for MDDs. It will be shown that the best routing can be extracted from the set of all possible solutions very easily. Finally, experimental results are given to demonstrate the feasibility of our approach.

1. INTRODUCTION

Channel routing is one fundamental subtask in the layout process of VLSI-design. As it is NP-complete in general it cannot be solved exactly within reasonable time bounds for large problem instances. Several heuristics can be used to obtain a fairly good result in many cases, but often especially in applications of practical interest, solutions are still far from being optimal.

In this context, it has been observed that often a majority of the amout of time needed by existing routing tools is being spent on wiring an only small part of the routing area while routing the rest of the channel is done very fast. For example, this happens if the channel contains many multi-terminal nets with lots of terminals and the routing strategy is to connect terminals with pre-computed focus points, i.e. for each net a centroid is computed and the terminals of the net are connected to that point. Usually these focus points are close to each other. In such cases, an exact routing for the small region that contains some or even all of the focus points can be extremly helpful.

To succeed in getting exact solutions to our routing problem, we introduce a transformation from a channel description into symbolic representation of corresponding functions. Here we use MDDs [7] which are a variant of BDDs *(Binary Decision Diagrams)* [2] and are able to deal with multi-valued variables. For this data structure it is known that basic operations, e.g. logical AND, can be carried out efficiently and so we gain as well in storage as in computation time using this method. There have been other works that use Boolean functions and BDD representation in order to obtain exact solutions on routing, but they all made restrictions on the routing model, like using FPGAs [8] or allow only one horizontal segment per net [4]. In our approach, for the first time free 3-dimensional routing is combined with such symbolic methods. It is free in the sense that any possible way in the channel can be used to connect two points.

In Section 2, some basic definitions are introduced. Afterwards, our channel representation is described in Section 3. In Section 4, we show how to generate the set of solutions and some first experimental results are shown in Section 5.

2. PRELIMINARIES

2.1. Multi-valued Decision Diagrams

As well-known, each Boolean function $f : \mathbf{B}^m \mapsto \mathbf{B}$ can be represented by an *ordered Binary Decision Diagram* (BDD) [2], i.e. a directed acyclic graph where a Shannon decomposition is carried out in each node.

Obviously, BDDs can be extended to represent functions $f : \mathbf{B}^m \mapsto \{0,..,h\}$ and the resulting graphs are denoted as *Multi-Terminal BDDs* (MTBDDs). The operations on MTBDDs can be carried out as efficiently as in the case of two terminals [3].

It is straightforward to extend MTBDDs to *Multi-valued Decision Diagrams* (MDDs) [7] representing functions $f : \{0,..,k\}^m \mapsto \{0,..,h\}$. For this, each internal node has k outgoing edges[1]. In [7] it has been shown that the ef-

[1] In our application we restrict ourselves w.l.o.g. to the case that all variables are defined over the same set of values.

ficient operations known for BDDs can also be carried out on MDDs using a *case*-operator instead of the *ite*-operator [1].

A DD is called *ordered* if each variable is encountered at most once on each path from the root to a terminal and if the variables are encountered in the same order on all such paths. A DD is called *reduced* if it does not contain vertices either with isomorphic sub-graphs or with all successors pointing to the same node.

As we need functions $f : \{0, .., k\}^m \mapsto \mathbf{B}$, our MDDs like BDDs only contain the terminal nodes 0 and 1.

2.2. Channel Routing

The goal of channel routing is to connect modules in ICs (integrated circuits) in an appropriate way. "Appropriate" here means that certain specified tuples of terminals of the modules have to be connected with each other and there must not be any short circuits (so-called *violations*) between the different connections. The main optimization task is to minimize the cumulated length of wires or alternatively the area needed for connecting. In channel routing, terminals can only be located at two opposite sides of the possible routing area. Many strategies, often graph-based, exist to find a routing that is as small as possible, for example see [6].

Our model of the channel considers this possible routing area as a 3-dimensional matrix M. x-direction corresponds to *columns*, y-direction to so-called *tracks* and z-direction is used to represent the different *layers*. The size of M determines the *length, width* and *height* of the channel. So we break up the channel into some grid points and allow only to connect a grid point with one or some of its neighbour points. There are no further restrictions, like the number of horizontal segments used for a connection for example. However, if needed, e.g. because of a special technology used for routing, such restrictions can easily be introduced as additional logical constraints.

To specify the terminals to connect we need a *netlist*. That is a list of tuples of terminals and each tuple defines one *net*. Of course, every grid point in M may at most be occupied by one net.

3. SYMBOLIC REPRESENTATION OF CHANNEL AND CONNECTIONS

Let N be the set of nets that have to be connected. Furthermore, $a(x, y, z)$ denotes the grid point at column x, track y and layer z in the channel.

We decide to represent every grid point by an MDD variable m_{xyz} ($x = 1 \ldots \textit{length}, y = 1 \ldots \textit{width}$, $z = 1 \ldots \textit{height}$). Let $|N| = k$, then the set of legal values for these variables is $\{0, \ldots, k\}$. $m_{xyz} = 0$ means that $a(x, y, z)$ is not occupied by any net whereas that point contains a wire segment that belongs to net n iff $m_{xyz} = n$. In this way we are able to express each specified wiring situation in the channel by giving just the values of the set of points of M. We call two variables $m_{x_1y_1z_1}$ and $m_{x_2y_2z_2}$ neighbours, iff their corresponding points in the matrix are neighbours, i.e. if $|x_1 - x_2| + |y_1 - y_2| + |z_1 - z_2| = 1$.

The terminals can be described as a triple of coordinates in the matrix. To connect two of them, it is necessary to assign some variables in the matrix the value corresponding to the net to connect. Afterwards, there has to be a sequence of neighbour variables starting at one terminal and ending at the other one. Additionally all the variables in the sequence *must* have this value. As an effect a specified two-point-connection r_n that is a part of a routing of net n can be viewed as the conjunction of exactly these variables in the given sequence:

$$\begin{aligned} r_n &= (m_{x_1y_1z_1} = n) \quad \wedge \ldots \wedge (m_{x_iy_iz_i} = n) \\ &\qquad \wedge \ldots \wedge (m_{x_ey_ez_e} = n) \end{aligned} \tag{1}$$

$m_{x_1y_1z_1}$ and $m_{x_ey_ez_e}$ correspond to the terminals and the $m_{x_iy_iz_i}$'s describe a path through the channel that connects them.

In the case of multi-terminal nets resp. connections, additional sequences of variables are needed to connect further terminals with one arbitrary already routed wire segment of the net. At the symbolic level, multi-terminal-connections can still be described by equation (1). We only have to add a further term to the conjunction for every variable contained in the additional sequences. Conjunction is a basic operation in MDDs and for this equation (1) can efficiently be represented by an MDD.

4. GENERATION OF SOLUTIONS

In the previous section, we showed how to express connections in a channel by MDDs. In order to generate the set of *all* possible routing solutions, we have to provide an MDD-description that fulfils the following conditions:

1. It has to contain not only one specified, but *any* possible routing path of a net.
2. Constraints of *all* nets have to be involved in *one* single MDD.
3. It is necessary to ensure that every grid point in the channel is occupied at most by one net.

Every variable m_{xyz} can be assigned exactly one value, and by this no channel point can be occupied by more than one net. In that way we can elegantly prevent the occurence of violations by using MDDs.

If the MDDs that represent all possible routes for every single net are known, they can be conveniently combined by another conjunction:

$$s = \bigwedge_{n \in N} \tilde{n} \tag{2}$$

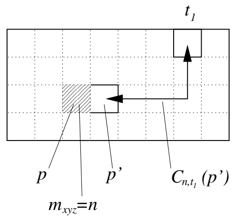

Figure 1: relationship between neighbour points in M

In equation (2), s denotes the set of possible routing solutions and $\tilde{n}$ the set of possible wirings for net n. So again if the MDD representations of the $\tilde{n}$'s are given, we can generate the desired MDD for s by basic MDD operations. A pleasant feature in our approach is the fact that the conflicts that occur when combining the single net routing conditions in one big MDD by equation (2) are removed implicitly by the MDD reduction mechanism.

Finally, we have to know how to construct the MDD for $\tilde{n}$. To do this, we choose one point t in the channel and introduce a function $C_{n,t} : M \mapsto \mathbf{B}$. We define:

$$C_{n,t}(p) :\Leftrightarrow p \text{ is connected with } t \text{ via net } n \tag{3}$$

Let $t_1 \ldots t_{max}$ be the terminals of net n. Then, the following holds:

$$\tilde{n} = \bigwedge_{i=2}^{max} C_{n,t_1}(t_i) \tag{4}$$

In order to get $C_{n,t_1}(t_i)$ for all $i \in \{2, \ldots, max\}$ we need to compute $C_{n,t_1}(p)$ for all $p \in M$. To do so, we assume that $p = a(x, y, z)$ and consider the relationship given in equation (5), that is also illustrated in Figure 1 for a better understanding:

$$\begin{aligned} C_{n,t_1}(p) \quad \Leftrightarrow \quad & (m_{xyz} = n) \wedge \\ & (\exists p' : C_{n,t_1}(p') \vee (p = t_1)) \\ & \text{where } p' \text{ is neighbour of } p \end{aligned} \tag{5}$$

To get $C_{n,t_1}(p)$ for all $p \in M$ we start an iteration with the following initialization: $C^0_{n,t_1}(t_1) = (t_1 = n)$ and $C^0_{n,t_1}(p) = \textit{false}$ for all $p \neq t_1$. In iteration l we compute $C^l_{n,t_1}(p)$ for all $p \in M$ according to (6).

$$\begin{aligned} C^l_{n,t_1}(p) \quad = \quad & C^{l-1}_{n,t_1}(p) \vee \\ & (\bigvee_{p'} (C^{l-1}_{n,t_1}(p') \wedge (m_{xyz} = n))) \\ & \text{where } p' \text{ is neighbour of } p \end{aligned} \tag{6}$$

We continue until in iteration l_{max} the following is true:

$$\forall p \in M : C^{l_{max}}_{n,t_1}(p) = C^{l_{max}-1}_{n,t_1}(p)$$

Then a fixed point is reached and finally we have:

$$\forall p \in M : C^{l_{max}}_{n,t_1}(p) = C_{n,t_1}(p) \tag{7}$$

Now we can easily generate $\tilde{n}$ using equation (4). Afterwards this procedure is repeated for the other nets. Having the results for all nets, we can compute s via (2). In many cases, this computation is very expensive, but *if* it can be done within reasonable time bounds, we are able to represent the set of *all* legal solutions to our channel routing problem by *one* MDD.

Equality checks on MDDs can be carried out within constant time, and the MDD representation of *false* does only contain one node. So it is trivial to check satisfiability of an MDD. But checking the satisfiability of s is exactly to decide if there is a solution for routing at all. In that way a very efficient method to check routability is provided here.

Of course there is an interest in the best solution to a problem or maybe also in the number of possible solutions. This information can be extracted from the MDDs by well-known graph-based search algorithms efficiently, too.

5. EXPERIMENTAL RESULTS

We implemented a system for exact routing that makes use of the MDD package presented in [5]. In this section, we present preliminary experimental results. Our computations were carried out on a SUN-Ultra-Sparc with 1GB main memory. We give the results for three examples, they are shown in Table 1 and Table 2. To demonstrate the behaviour of our system for one routable channel and one without any legal solutions, we consider two basic examples and call them `solvable` and `unsolvable`. The channel characteristics for these examples and for the third example `easy`, too, are shown in Figure 2.

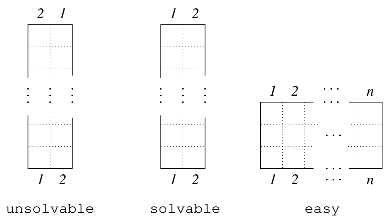

Figure 2: channel characteristics of our examples

The channels consist of two columns in one layer. We parametrize the number of tracks and there are two two-terminal nets to connect. In `unsolvable`, terminals of different nets are located in opposite to each other, so that no legal solution is possible while in `solvable` terminals of the same net belong to the same column.

Of course, in `unsolvable` no MDD nodes are needed for the final result because there are no solutions for this

unsolvable										
n	2	3	4	5	6	7	8	9	10	11
time[s]	0,06	0,09	0,2	0,58	2,53	8,75	30,53	106,11	490,01	1733,07

Table 1: results for `unsolvable`

solvable		
n	time[s]	# nodes
1	0,13	0
2	4,58	36
3	195,18	132
4	11702,3	290

easy		
n	time[s]	# nodes
1	0,06	3
2	0,08	6
3	0,5	9
4	25,42	12

Table 2: results for `solvable` and `easy`

problem and so the MDD only consists of the zero terminal node. In all our examples the number of MDD nodes is low. The reason for this is that the MDD that represents the final set of solutions in general is much smaller than the MDDs for the intermediate results we construct before. In fact, we have to allocate more nodes during computation than the result MDD contains. This legitimates the amount of time that is needed to generate the MDD representing the solution set. Within 1 hour computation time, we were able to deal with up to 11 tracks. The representation of the solutions of `solvable` of course requires several MDD nodes. It is not surprising that the number of MDD-nodes is expotentially increasing in the number of tracks used, but nevertheless computation within 4 hours is possible for channels with up to 4 tracks. Storage is much less critical as computation time here.

We have made the observation that significiantly more computation time and MDD nodes are needed if the channel for the problem is sparse, i.e. if in a legal solution there can still be many grid points that are not occupied by any net. A reason for this is that in sparse channels, there are much more alternatives for routing and so there are more possible solutions, too. Their representation requires bigger MDDs and by this the duration of computation is increased enormously. As a consequence our approach is suitable especially if the channel is dense and in combination with low size such channels are of special interest for exact routing (cf. Section 1).

Table 2 shows the results of channel `easy` that has *height*=1, *width*=3 and parametrized *length* (see Figure 2). For every column, we have one net with two terminals that are located at the ends of the column. Here we can see that the final solution contains exactly as much nodes as there are grid points, i.e. we have reached a lower bound in this case. It would be possible to solve this problem with even more columns, but the MDD package we used allows only 5 outgoing edges per node and so we have to restrict to problems with less than five nets.

6. CONCLUSIONS AND FUTURE WORK

In this paper we developed a new approach for the channel routing problem. We use symbolic representation (MDDs) and profit considerably from the efficient operations that are provided on this data structure. For the first time it has been demonstrated how to create a feasible representation of the set of *all* legal solutions to the channel routing problem. We also showed that routability and the best solution can easily be extracted from this solution set. Some promising experimental results have been given.

At the moment, we are in progress and adapt our system in order to provide the possibility of using a state-of-the-art MDD package. Further on the order in which the nets are considered is important because we compute the final set of solutions by adding net constraints on the fly directly after they have been constructed. Thus, optimizing this ordering may bring further improvements.

7. REFERENCES

[1] K.S. Brace, R.L. Rudell, and R.E. Bryant. Efficient implementation of a BDD package. In *Design Automation Conf.*, pages 40–45, 1990.

[2] R.E. Bryant. Graph - based algorithms for Boolean function manipulation. *IEEE Trans. on Comp.*, 35(8):677–691, 1986.

[3] E. Clarke, M. Fujita, P. McGeer, K.L. McMillan, J. Yang, and X. Zhao. Multi terminal binary decision diagrams: An efficient data structure for matrix representation. In *Int'l Workshop on Logic Synth.*, pages P6a:1–15, 1993.

[4] S. Devadas. Optimal layout via Boolean satisfiability. In *Int'l Conf. on CAD*, pages 294–297, 1989.

[5] R. Drechsler. Verification of multi-valued logic networks. In *Int'l Symp. on multi-valued Logic*, pages 10–15, 1996.

[6] T. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout*. Teubner, Wiley, 1990.

[7] A. Srinivasan, T. Kam, S. Malik, and R.E. Brayton. Algorithms for discrete function manipulation. In *Int'l Conf. on CAD*, pages 92–95, 1990.

[8] R.G. Wood and R.A. Rutenbar. FPGA routing and routability estimation via Boolean satisfiability. *IEEE Trans. on VLSI Systems*, 6(2):222–231, 1998.