BDDMAP: a technology mapper based on a new covering algorithm

D.S. Kung, R.F. Damiano, T.A. Nix
IBM T.J. Watson Research Center, Yorktown Heights, NY 10598

D.J. Geiger
IBM Corporation, P.O. Box 950, Poughkeepsie, NY 12602

Abstract

We present a technology mapper which combines the strengths of rule-based heuristics and algorithmic techniques. Matching is mainly performed by rule-based heuristics which is complemented by functional matching based on OBDD's. The novel aspects of the covering algorithm include using an anticipative cost function, global cost propagation and handling of multiple output technology gates. We obtain favorable results when BDDMAP is benchmarked against the mapper of MisII, release 3.5, using a complete set of recommended MCNC examples. Topic number 4.1

1 Introduction

Technology mapping is the implementation of a Boolean network, henceforth referred to as the target network, using technology dependent gates from a prescribed library of primitives. The various approaches to this difficult mapping problem can be broadly divided into 4 categories: rule-based mapping [1], graph matching [2], direct mapping [3] and functional matching [4]. Technology mappers that use rule-based techniques [1] are usually fast but the rules tend to be ad hoc and heavily dependent on the library. In graph matching [2] the target network and the library cells are usually expanded into a two-input NAND/NOT representation. Since a two-input representation of a logic function is not canonical, library elements could have a large number of topologically inequivalent but functionally equivalent representations even if the target design is assumed to be free of redundancies. The target network is frequently partitioned into trees and then tree matching and dynamic programming algorithms can be applied to each tree partition to obtain a local optimal solution. As a post-process, some further local transformations such as inverter optimization can be applied across the partitions to improve the mapping results. While this algorithm is efficient on designs in which the tree partitions are relatively large and deep, the library is not fully utilized if the tree partitions contain very few gates. Moreover, more complex library gates (e.g. XOR) do not have a tree representation of NAND/NOT gates so a separate process must be used. In the functional matching approach [4] a library gate is a candidate for implementing a pattern of logic in the target network if both of them compute the same logic function. This boolean equivalence checking problem can be expedited by employing a canonical form to represent the logic functions. One disadvantage is that the canonical form is often storage expensive.

In this paper, we present a technology mapper, BDDMAP, which uses rule-based and functional matching. The covering algorithm is an improvement over the dynamical programming - tree partitioning method used in DAGON. Rule-based heuristics or functional matching is invoked to match the type of technology gates for which it is most efficient. We find this judicious division of labor provides the most efficient and effective matching process. The algorithmic part of BDDMAP lies in the covering process. The novel aspects of the covering algorithm are using an anticipative cost function, global cost propagation and handling of multiple output gates. In section 2 we discuss the mapping problem in the context of matching of patterns and covering of the target network. An overview of BDDMAP is given in section 3. The matching process, namely, the functional matching technique and the rule-based heuristics are discussed in section 4. This is followed by benchmark results and concluding remarks.

2 The Mapping Problem

The target logic network is assumed to consist of single output gates with standard Boolean functions. The basic entity for matching is a pattern of logic which is defined as follows. A pattern of logic, C, is a subgraph of the target network. The nets of the pattern with no source from within the pattern are the input nets of the pattern. The nets of the pattern with no sinks within the pattern are outputs of the pattern although other nets with fanouts outside the pattern could also be designated as outputs. Nets of the pattern with fanouts outside the pattern but are not themselves designated outputs of the pattern are copy nets. A pattern with n inputs and m outputs computes a logic function with the same inputs and outputs. If such a logic function is equivalent to a logic function available in the library, then a match is found in the pattern. A cover of the target logic network is a collection of matched patterns such that
the union of these matches includes the target network with the following constraint. Every input of a match is either a primary input of the target network or connected to the output of some match and every output of a match is either a primary output of the target network or connected to the input of some match. Overlapping among the matched patterns could occur at the copy nets and are resolved by copying the logic gates in the overlap. The technology mapping problem thus consists of the matching problem and the covering problem. The matching problem is to select a set of patterns and identify the matches among these patterns. The covering problem is to select a cover which minimizes the area or power or delay or a combination of these subject to constraints like testability and fanin/fanout restriction.

3 BDDMAP

BDDMAP is a general purpose technology mapper with good performance and high efficiency. BDDMAP is outlined as follows:

A. Matching
   i. pre-processes
   ii. functional matching
   iii. post-processes

B. Covering
   i. cost propagation
   ii. technology binding
   iii. post-processes.

It combines both rule-based and algorithmic techniques to handle the mapping problems in which libraries exhibit a wide variety of characteristics. The philosophy we adopt is to separate the algorithmic operations, namely, functional matching and covering and the rule-based operations, namely, the pre- and post-processes. Matching is done first and then the target network is covered after all the matches have been gathered. The separation of the mapping into these two stages allows a more global covering algorithm to be used.

The covering algorithm incorporates heuristics to attain a more global optimization. The cost of a match depends on the implementations adjacent to the match. The cost propagation from inputs to outputs are not terminated at fanout points so that the covering algorithm is more global in nature. Another new aspect of the covering algorithm is that multiple output matches are handled as an integral part of the algorithm. A detailed description of the covering appears in section 5.

4 The Matching Algorithm

In BDDMAP, we optimize the division of labor between functional matching and rule-based heuristics. The complex unsymmetric technology gates, the basic gates (NAND, NOR, AND, OR), the multiple output gates and the non-standard gates such as those in bipolar technologies are handled by the pre- and post-processes. Functional matching complements the rule-based techniques by matching some of the basic gates missed by the rule-based approach, such as XOR's, XNOR's and small unsymmetric gates like the 2-1 selector.

For each library element handled by functional matching, OBDD's are created to represent the logic function of that element. For library functions that are not totally symmetric, we create all the OBDD's which represent the inequivalent input orderings. Functional matching is only applied to technology gates with large amount of symmetry or small number of inputs to avoid the possible exponential blowup of OBDD's. As for generating OBDD's for the target network, we limit ourselves to subset of patterns of logic generated by a breadth-first traversal of the network. This subset of patterns is chosen because functions that are symmetrically decomposed can be matched easily. If the OBDD for a pattern is isomorphic to an OBDD in the library, a match is added to the list of matches associated with the gate at the root of the pattern.

The main functions of the pre- and post-processes for matching are rule-based matching and network priming. Complicated library elements such as adders, large multiplexors and decoders are matched efficiently using rule-based techniques such as those used in LSS [5]. For multiple output matches, the match is added to the lists of matches associated with the source box of every output net of the match. Matches for basic gates are most easily obtained while decomposing large basic gates into smaller ones. The matching of AOI's, OA's, AOI's and AOI's are best done as a post-process based on the NAND, NOR, AND and OR matches obtained thus far.

Network priming refers to selecting a decomposition of the target network so that better matches are exposed. This is particularly important for the matching of XOR's, AO's, OA's and their complements. No patterns in the target network in Fig. 1(a) can be matched to a two-input XOR. However, if the target network is decomposed as in Fig. 1(b) the XOR match (inside dash box) is exposed.

Some of the most important matches are obtained by adjusting the phase of the inputs and outputs of a pattern. The standard approach is to insert double inverters in the network, which increases the number of target gates substantially. In BDDMAP we circumvent this increase in complexity by the use of decomposition matches. A decomposition match is a network whose internal nodes consist of library elements. In
this application, the decomposition match consists of a main match and inverters to the some of the inputs and outputs of the main match. As an example, the function $d = AO21(a,b,c)$ can be matched by the decomposition match $[p = AO21(a,b,c), d = INV(p)]$ where $p$ is an internal net which will be realized if the match is used. In the cost calculation during covering, the cost of a decomposition match is assigned in such a way as to anticipate the cost upon implementation of the technology gates. A detailed description of the anticipated cost appears in section VII. These decomposition matches are generated after the the preprocesses and functional matching and are often derived from matches already found.

5 The Covering Algorithm

The covering algorithm is an extension of the dynamic programming technique for tree covering [2] and is optimal if the target network is a forest of trees. We emphasize that the network is not partitioned into trees so the matches are separated into two different classes. Matches with copies are the ones in which some nets internal to the match have fanouts to gates outside of the match and these nets are referred to as copy nets. Otherwise, the matches are said to have no copies.

In this paper we will concentrate on area optimization so the cost function is related to the area of the matches. We proceed to discuss the cost calculation algorithm for covering. If the target network is a tree, the principle of optimality implies that for every match $M$ at the root of the tree, the cumulative cost, $C_M$, of an optimal cover containing $M$ is the sum of the cost of $M$ and the cost of an optimal cover of each subtree rooted at the inputs of $M$. The best match at the root is the match $M^b$ such that the cumulative cost $C_{M^b}$ at the root is minimum and $C_M$ is the cost of an optimal cover for the tree. Therefore, computing the best match and minimum cumulative cost at each net gives the cost of an optimal cover for a tree. The following is the extension of the cost calculation to a general DAG. The cumulative cost at a net $j$, $C_j$, of a match with $j$ as an output and with no copies is simply

$$C_j = \frac{W}{m} + \sum_{i \in I} \frac{B_i}{f_i},$$

where $I$ is the set of input nets of the match, $W$ is the anticipated cost of the match, $m$ is the number of outputs of the match, $B_i$ is the best cumulative cost at net $i$ and $f_i$ is the fanout of net $i$. If a match has copies, we compute for each copy net, $l$, the set $S_l$ of input nets of the match that are in the cone of influence of $l$. Let $n_i$ be the number of times $l$ appears in the sets $S_l$ for each copy net. Then the best cumulative cost of the match is

$$C_j = \frac{W}{m} + \sum_{i \in I} \frac{B_i}{f_i + n_i},$$

The best match is the match with the minimum cumulative cost and this minimum cost is associated with net $j$ as the best cumulative cost. The initial condition is that the best cumulative cost at the primary input nets and register output nets is zero. The anticipated cost of a match is the cumulative cost at the output of the match if the best matches at the input nets of the match are realized and double inverters are removed. The use of decomposition matches and anticipated costs obviates the use of double inverter insertion at every net.

The above cost function has the following nice property:

Let the target network be transformed into a DAG by removing the registers. If all the matches for the target network have no copies, the cost of an optimal cover for the DAG is the sum of the best cumulative cost of all the primary outputs, using the cost calculation of eqn. 1.

For our case, matches are allowed to have copy nets and the fanout of the nets changes as gates are being copied. The optimal cover can no longer be easily derived from the best cover of subnetworks. Eqn. 2 attempts to attain a reasonable approximation to the optimal case by estimating the effect of using a match with copies. Note that eqn. 2 disfavors matches which contain copy nets with a high external fanout, which is in agreement with our intuition.

The following example serves to illustrate the above concepts. Let a model network and a toy library be displayed in Fig. 2. The best match and the best cumulative cost at some nets are as follows: $B_9$ - NAND2(1,2), $C_3 = 2$; $B_7$ - NAND2(3,2), $C_7 = 4$; $B_{10} - \{p = AO21(3,4,5), 10 = INV(p)\}$, $C_{10} = 5$; $B_{19} - \{13 = OR2(10, p), p = INV(10)\}$, $C_{19} = 6$.

In the calculation of $C_{19}$, $B_{19}$ is a decomposition match with copy net 7. so $W = 4$, $n_3 = 1$ and $f_3 = 1$. In the calculation of $C_{13}$, the anticipated cost $W = 3 = 2 = 1$ due to the anticipated double inverter removal.

The gates in the combinational network are topologically sorted from inputs to outputs. The best cumulative cost for each net is computed in this topol-
tical order. With all the best cumulative cost and best matches in place, the target network is bound to the technology gates chosen as best matches. The order of binding is according to the following queue. All the primary output nets and input nets to registers are put on the queue. The rest of the nets are put whenever all the gates to which they fanout are bound to technology. In the course of technology binding, matches that can no longer be realized are invalidated. In addition, the technology binding affects the fanout of the nets which in turn changes the cumulative cost of the matches. Therefore, we constantly update the cumulative costs and the best matches are replaced as more favorable ones take over.

6 Results

We present the results of BDDMAP on some benchmark examples [6] for the 1991 International Workshop on Logic Synthesis. The library we used is an IBM CMOS library which is similar to but more extensive than the MOSIS 2u standard cell library in the MCNC distribution [6]. We compare the area results (in terms of internal cells of BDDMAP with the technology mapper of MisII [7]). The technology independent optimizations are done using BooleDozer so the starting point for the MIS technology mapper and BDDMAP is exactly the same. Therefore only the performance of the technology mapping stage is being compared. The parameters for the MisII mapper is set such that logic duplication is allowed and multiple fanouts within the match is allowed. This setting should bring out the full strength of the MisII technology mapper of MisII [7]. The technology independent optimizations are done using BooleDozer so the starting point for the MIS technology mapper and BDDMAP is exactly the same. Therefore only the performance of the technology mapping stage is being compared. The parameters for the MisII mapper is set such that logic duplication is allowed and multiple fanouts within the match is allowed. This setting should bring out the full strength of the MisII technology mapper for area optimization. All the BDDMAP runs are performed using the same scenario (or script) so that no fine tuning is being performed. In addition, instead of limiting ourselves to examples which are favorable to BDDMAP, we benchmark all the recommended multi-level examples.

The recommended benchmarks cover a variety of model size and model types. There are ALU's, control logic, error correction logic, data encryption logic, multiplier and priority encoder. The mapping results are displayed in Table 1. In the thirteen examples, BDDMAP clearly performs better in 10 examples. 2 examples have a difference of 2 cells or less and MisII clearly performs better on 1 example, rot. On the average, BDDMAP performs better in area by 3.55 percent. The runtime of BDDMAP takes approximately twice as long as the MisII mapper but the largest model, des, takes only 10 minutes or so to run on a RISC6000 530 for BDDMAP.

7 Summary and Conclusions

It is common for technology libraries to contain gates which are as simple as a two input NAND or complex as a 32 bit adder. Pure algorithmic approaches, though aesthetically pleasing and conceptually unifying, are impractical in handling complex gates. Rule-based techniques serves well to complement the weaknesses of algorithmic techniques. BDDMAP aims to strike a balanced compromise between these two approaches. Multiple output gates and connectivity constraints for bipolar technologies are handled efficiently by the mapper. The competitiveness of BDDMAP is shown by the benchmark comparison with the mapper of MisII. The current mapper is primarily optimized for area and delay optimization will be incorporated in the future.

Acknowledgements

We would like to thank Louise Trevillyan and Len Berman for useful discussions and Lukas van Ginneken for a critical reading of the manuscript.

Table 1: Results from ISCAS Multi-level Examples

<table>
<thead>
<tr>
<th></th>
<th>BDDMAP</th>
<th>MisII</th>
</tr>
</thead>
<tbody>
<tr>
<td>C432</td>
<td>412</td>
<td>416</td>
</tr>
<tr>
<td>C1908</td>
<td>1150</td>
<td>1154</td>
</tr>
<tr>
<td>C6288</td>
<td>2312</td>
<td>2311</td>
</tr>
<tr>
<td>C7552</td>
<td>4808</td>
<td>4807</td>
</tr>
<tr>
<td>t481</td>
<td>1781</td>
<td>1844</td>
</tr>
<tr>
<td>rot</td>
<td>1435</td>
<td>1412</td>
</tr>
<tr>
<td>b9</td>
<td>241</td>
<td>239</td>
</tr>
<tr>
<td>dalu</td>
<td>2887</td>
<td>3037</td>
</tr>
<tr>
<td>des</td>
<td>6676</td>
<td>6903</td>
</tr>
<tr>
<td>k2</td>
<td>2065</td>
<td>2190</td>
</tr>
</tbody>
</table>

References


