MDD-Based Synthesis of Multi-Valued Logic Networks*

Rolf Drechsler
Albert-Ludwigs-University
Freiburg, Germany
drechsle@informatik.uni-freiburg.de

Mitch Thornton
Mississippi State University
Mississippi State, Mississippi
mitch@ece.msstate.edu

David Wessels
University of Arkansas
Fayetteville, Arkansas
wessels@engr.uark.edu

Abstract

A method for the synthesis of large Multi-Valued Logic Networks (MVLNs) using Multi-Valued Decision Diagrams (MDDs) is presented. The size of the resulting circuit is linear in the size of the original MDD. In contrast to previously presented approaches to circuit design using MDDs, here the nodes are not substituted by multiplexers. Instead, a small circuit is created representing the functionality of each edge in the graph. The resulting circuits have nice properties with respect to area/delay estimation and power dissipation. Experimental results are given to illustrate the efficiency of the approach.

1. Introduction

The use of *Decision Diagrams* (DDs) is becoming increasingly popular in the area of electronic design automation. DDs represent a function as a directed acyclic graph, and have generated great interest due to their ability to represent certain functions in a very compact form. DDs can be regarded as representing the function in a behavioral, rather than structural form. The structure of the DD does not necessarily impart any information about the structural representation of a corresponding circuit. This "behavioral" nature of DDs has led to their increasing popularity in synthesis and verification tools.

Several approaches to map DDs directly to a target architecture have been proposed (see e.g. [1, 2, 4]). The resulting circuits generally have the same approximate size as circuits generated by other synthesis tools such as SIS [17] and often have other desirable properties. Extensions to the multi-valued case have also been proposed [18]. The basic underlying idea of all these approaches is to substitute each node in the DD by a sub-circuit. For example, 2-input multiplexers may be used in the case of a BDD [2]. In this type of approach, the "flow" of information is reversed since BDDs are usually evaluated from the root node to the terminal vertices. However, after this type of circuit translation, the root node becomes the output of the circuit. Due to this transformation, some of the desirable properties are lost. As an example, during the evaluation of a BDD only one path is activated resulting in low switching activity but this is not the case in the corresponding translated circuit. Based on this property, an alternative approach to mapping BDDs has been presented [15]. The resulting circuits have very low power dissipation, while keeping most of the properties of BDDs.

Here, a method for the synthesis of *Multi-Valued Logic Networks* (MVLNs) based on *Multi-Valued DDs* (MDDs) is described. Following the ideas presented in [15] we show how MDDs can be mapped to netlists without reversing the information flow. The size of the resulting MVLNs is linear in the size of the MDD. Experiments for 3-valued circuits are given to show the efficiency of our approach.

The paper is structured as follows: In Section 2 MVLNs and MDDs are defined. The basics of our synthesis procedure are described in Section 3. In Section 4 experimental results are given. Finally the results are summarized.

2. Preliminaries

A description of the notation used and a brief review of the concepts in MVL are given in this section. We also describe and define logic networks and decision diagrams for MVL.

*This work was carried out during a visit of the first author at the University of Arkansas and was supported by DAAD grant 315/PPP/gü-ab and NSF grants CCR-9633085 and SBE-9815371.

2.1. Multi-valued logic networks

In general, a *Multi-Valued Logic Network* (MVLN) can be modeled as a directed acyclic graph $C = (V, E)$ with the property that each vertex, $v \in V$, is labeled with the name of a basic cell, the name of a *Primary Input* (PI) or the name of a *Primary Output* (PO). The collection of basic cells that are possible in the MVLN is given by a fixed library that contains MIN-, MAX-, INV- and LITERAL-gates as a minimum[1]. It is possible to include basic cells with arbitrary complexity and with a varying number of inputs. There is an edge, (u, v), in E from vertex u to v, if and only if an output pin of the cell associated with u is connected to an input pin of the cell associated with v. Edges also contain additional information to specify the pins of the source and sink nodes that they are connected to. Vertices have exactly one incoming edge per input pin. Nodes labeled as PI (PO) have no incoming (outgoing) edges.

To simulate a MVLN, each PI may assume the values of a given ordered finite set, $P = \{0, \ldots, k-1\}$, where k denotes the number of logic levels. The complement (INV-gate) of a signal x is defined as $\overline{x} = (k-1) - x$. A LITERAL-gate (a, b) $(a, b \in P, 0 \leq a \leq b < k)$ has one input and one output[2]. For a given input x, the behavior of such a gate is defined by:

$$f(x) = \begin{cases} k-1 & : \quad a \leq x \leq b \\ 0 & : \quad otherwise \end{cases}$$

It is also assumed that the characteristic functions are available for each of the primary inputs (i.e. the set of $J_j(x_i)$ values such that $J_j(x_i) = k-1$ if $x_i = j$, and $J_j(x_i) = 0$ otherwise).

2.2. Multi-valued decision diagrams

As is well-known, each Boolean function $f : \mathbf{B}^n \rightarrow \mathbf{B}$ can be represented by a *Binary Decision Diagram* (BDD) [7], which is a directed acyclic graph where a Shannon decomposition is carried out in each node. BDDs can be extended to represent functions $f : \mathbf{B}^n \rightarrow \{0, .., k-1\}$ and the resulting graphs are denoted as *Multi-Terminal BDDs* (MTBDDs). The operations on MTBDDs can be carried out as efficiently as in the two terminal case of BDDs. [3, 8]. In turn, MTBDDs can be extended to *Multi-Valued Decision Diagrams* (MDDs) [19] representing functions $f : \{0, .., k-1\}^n \rightarrow \{0, .., k-1\}$. In the case of MDDs, each internal node has k outgoing edges[3]. In [19] it has been shown that the efficient operations known for BDDs can also be carried out on MDDs using a *case*-operator instead of the *ite*-operator [5].

A DD is *ordered* if each variable (which is represented by a non-terminal graph vertex) 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 *reduced* if it does not contain vertices either with isomorphic sub-graphs or with all successors pointing to the same node.

In the remainder of this paper we only consider reduced, ordered MDDs.

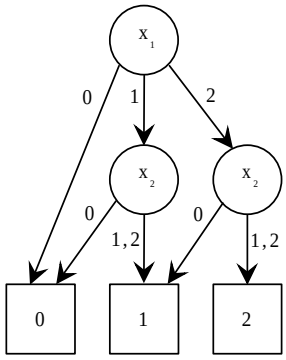

Figure 1. Diagram of Ordered and Reduced MDD Graph Structure

Example 1 *Figure 1 shows a diagram of a reduced and ordered MDD of the two-variable three-valued function f given by the following truth-vector:*

$$\mathcal{F} = [000011122]$$

When the MVLN is represented as a directed acyclic graph, a corresponding MDD can be created as follows:

1. Terminal nodes for the k constant functions are created.

2. For each PI of the MVLN, a graph vertex (variable) in the MDD is created, where the i-th outgoing edge points to the terminal node labeled i ($i \in \{0, .., k-1\}$).

3. The gates of the MVLN are visited in topological order and the corresponding MDD operation is carried out.

The topological order visit guarantees that all inputs of a gate are known before it is evaluated. After all MVLN nodes have been visited, MDDs for all POs will result.

[1] In the binary case, MIN- and MAX-gates correspond to AND- and OR-gates, respectively.

[2] These LITERAL-gates are also called *window literals*.

[3] In our application we restrict ourselves to the case that all variables are defined over the same set of values.

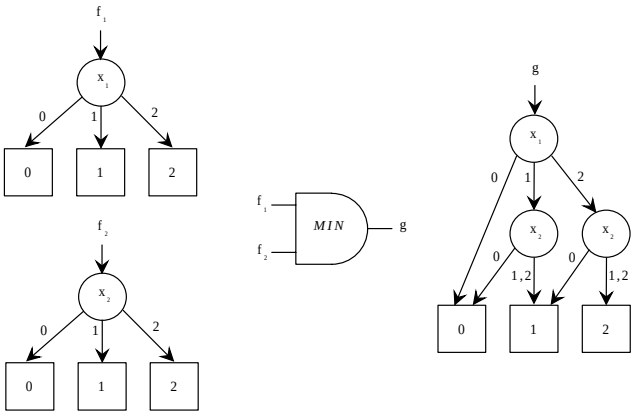

Figure 2. Symbolic simulation for MIN-gate

Example 2 *In Figure 2 a simple example for a three-valued simulation for a MIN-gate is shown. The input a_1 (a_2) corresponds to the MDD f_1 (f_2). The output of the gate b corresponds to the function that is represented by the MDD g.*

3. Synthesis of MVLNs from MDD

For the synthesis process, we assume an MDD representing a k-valued function of k-valued variables is given initially. The outgoing edges per node of an MDD are mapped to small sets of logic gates, producing a k-output circuit. If the function being computed is $f(X)$, then the k outputs of the resulting circuit correspond to the characteristic functions of f, (i.e. $J_0(f(X)), \ldots, J_{k-1}(f(X))$). The circuit outputs thus form a $1-of-k$ code, where the i^{th} output of the circuit is logically true if and only if the MDD would evaluate to logic value i.

In the remainder of this section, we first show the general technique for mapping a MDD to a MVLN. It is assumed that the basic types of logic gates that are available compute the MIN and MAX functions. It is also assumed that each characteristic function is available for each primary input. Next, we consider implementing the MDD as a binary logic circuit.

3.1. Mapping MDDs

Each edge in the MDD is translated into one or more gates as shown below by the following four steps.

1. The activating characteristic functions of input x_i serve as inputs to a single MAX gate, $MAX1$.

2. The predecessor edge values serve as inputs to a second MAX gate, $MAX2$.

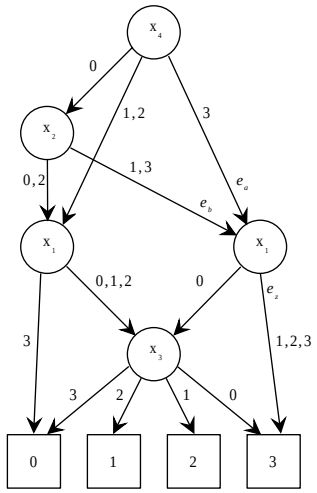

Figure 3. 4-input 4-value MDD

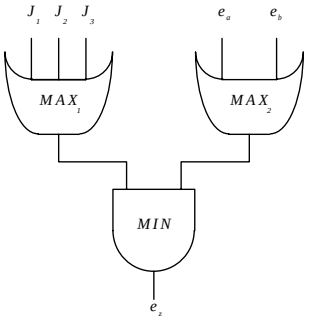

Figure 4. Translation of edge e_z

3. The outputs of the two MAX gates serve as input to a single two-input MIN gate.

4. The MIN gate produces the final value for the specified edge.

Example 3 *Consider the MDD in Figure 3. If a realization for edge e_z is to be constructed, the two incoming edges to this node are needed, e_a and e_b, as are the characteristic functions J_1, J_2, and J_3. The circuit fragment in Figure 4 shows the gates and edges relevant to edge e_z.*

Note that all single-input MAX/MIN gates are redundant and can be removed in a single pass following the translation. Furthermore, the MAX gate that determines if any predecessor edges are activated ($MAX2$ in Figure 4) can be shared across all output edges that are incident to the same node of the MDD.

Each node in the MDD has at least two incident output edges (otherwise the node is redundant), and has at most k such edges. This places the total number of gates required

for the ϵ output edges that are incident to a particular node at:

- ϵ two-input MIN gates,
- $0 \ldots 1 + (k/2)$ MAX gates (specifically $k + 1 - \sum_{j=1}^{\epsilon}(d_j - 1) - \sum_{d'_j s=1}(d_j)$ where d_j represents the number of distinct values of variable x_i which activate edge e_j)

The total number of gates is bounded by $k + 1$. This can be seen since the node under consideration requires 0 or 1 MAX gates as a collector from preceding edges and can have at most k outgoing incident edges, each of which requires one two-input MIN gate. Collapsing edges (i.e. multiple x_i values activate the same edge) reduces the number of MIN gates required, but requires a MAX gate for each set of collapsed edges to collect the activating characteristic functions.

3.2. Implementations in binary logic

The only gate types described for use above are MAX and MIN gates with the assumption that all of the characteristic functions of each primary input available. For the binary logic case ($k = 2$), each characteristic function gives an output of 0 or $k - 1$. A MAX or MIN gate operating on only values of 0 or $k - 1$ can only produce outputs of value 0 or $k - 1$, therefore the output of each gate throughout the network will also be either 0 or $k - 1$. Hence the translation to a binary system is entirely natural, using OR gates for MAX, and AND gates for MIN [15].

With respect to the characteristic functions, for each k-valued input, x_i, to the MDD, we need a component generating outputs $J_{i,0} \ldots J_{i,k-1}$ such that $J_{i,v} = 1$ if and only if $x_i = v$. If each x_i primary input is supplied using $log_2(k)$ inputs, then the set of characteristic functions for each input can be supplied using a $log_2(k) \rightarrow k$ decoder. The result is a binary $M \times k$-input, k-output circuit, in which exactly one of the k outputs is activated for each input combination.

Gate sets other than AND/OR can also easily be obtained [21]. The structure of the circuit resulting from this mapping technique is entirely composed of AND/OR gates. It is thus easily translated into a comparably-sized network of NAND gates, still resulting in output coded in a $1 - of - k$ format.

3.3. Properties of the resulting MVLN

Having covered the construction of the network, we now consider its applicability for common optimization techniques. One of the major goals of circuit synthesis systems is to enable the use of optimization techniques with regard to circuit delay, area and power dissipation. This section summarizes the application of various optimizations to circuits produced using the synthesis process described above.

3.3.1. Circuit area

Assuming circuit size is proportional to the number of logic gates in the circuit, one of the advantages of the proposed technique is that the size of the circuit produced is directly proportional to the size of the MDD. Thus techniques which minimize the size of the MDD [10] produce corresponding reductions in the resulting circuit size. It also allows the use of other aspects, (e.g. planarity [18]) to be considered during the circuit placement and routing phase. Compared to the standard mapping approach where each node is substituted with a multiplexer, this technique has the advantage that further reductions can occur through the removal of redundant logic gates.

3.3.2. Power dissipation

Power constraints are another major design factor in synthesis since reduction schemes can minimize heat dissipation and limit the drain on battery powered devices. Equation 1 expresses the dynamic power dissipation relation for CMOS technology for binary-valued logic.

$$V_{dd}^2 \cdot f_{clock} \cdot \sum_{i=1}^{N}(Sw_i \cdot c_i) + V_{dd} \cdot \sum_{i=1}^{N}(i_{sc}) \qquad (1)$$

For a circuit of N nodes, P_i is the probability of a $high$ output from node i, Sw_i is the switching probability of node i, c_i is the node capacitance, V_{dd} is the supply voltage, f_{clock} is the clock frequency and i_{sc} is the node short circuit current. The underlying assumption is that the circuit is switching between binary values. This assumption is valid for all but the primary inputs. Equation 1 also requires knowledge of the switching probability, Sw_i, for each circuit node.

If we make the assumption that each value from $0 \ldots k-1$ is equally likely for each primary input x_i at each step, and that the values for the different input variables are statistically independent, then we can calculate the Sw_i values as follows:

- The switching probability at a node is $Sw_i = (P_i \cdot (1 - P_i)) + ((1 - P_i) \cdot P_i) = 2(P_i - P_i^2)$, where P_i is the probability the node output is $high$.
- The probability of a $high$ value at the output of each characteristic function is $1/k$.
- For an MDD, no node can have two incoming edges activated simultaneously, and correspondingly in our circuit no MAX gate will ever have $high$ values on two different inputs simultaneously. Thus, the P_i

value for the output of a MAX gate is simply the sum of the P_i values of its inputs.

- The P_i value for the output of a MIN gate is naturally the product of the P_i values for all of its inputs.

By applying these rules on a traversal of the circuit we can easily compute the P_i values, and hence obtain the Sw_i values (with the statistically independent and uncorrelated assumptions) for each node in the circuit. This in turn allows us to apply the power estimation formula referred to above. Notice also that the power dissipation is likely to be much lower using our synthesis approach (analogously to the binary case [15]) compared to the standard method where the MDD nodes are substituted by multiplexers. Here, we profit from the fact that using our technique the flow of information is not reversed and similar to BDDs, exactly one path is activated for a given variable assignment.

The efficient computation of signal probabilities can also be used for testing the circuit. Based on the switching activity of each signal efficient built-in self-test can be constructed.

3.3.3. Delay calculations

In general circuits derived from DDs tend to have linear depth. However, the same properties which simplify the calculation of switching probabilities also simplify the problem of delay analysis in the circuit produced. The "false path" problem [9], which plagues circuit timing analysis, is avoided since the circuit structure corresponds directly to the structure of the MDD and each path is activated by some particular input assignment.

It is also possible to transform linear depth DDs to a *log(depth)* form as described in [14] [13] [20]. Performing transformations such as these before mapping can allow for further reductions in the resulting circuit delay.

4. Experimental results

We used the MDD package described in [10] to generate experimental results using the ISCAS89 benchmarks from [6]. This same set was also used in [10, 11]. Specifically, some of the combinational parts of the benchmarks were used since no standard benchmark set of large circuits is available for multi-valued designs. We interpreted the benchmarks as 3-valued circuits by transforming AND-gates into MIN-gates and OR-gates into MAX-gates. In doing so the resulting circuits do not contain LITERAL-gates. Therefore the strength of our results is limited. But we expect that they describe a trend which is also valid for multi-valued circuits containing LITERAL-gates. All measurements were performed on a *Sun Ultra 10*. A node limit of 250.000 nodes and a limit of 3.600 CPU seconds was used.

Table 1. Benchmark functions

name	PO	synthesis	
		MIN	MAX
s27	36	88	56
s208	1077	2698	1610
s298	299	754	442
s344	324	858	438
s400	474	1228	668
s444	491	1263	701
s510	904	2235	1381
s641	4040	9874	6286
s713	4050	9901	6299
s820	1217	3094	1774
s832	1226	3123	1781
s953	2545	6250	3930
s1238	46521	123365	62719
s1288	2085	5349	2991
s1494	2044	5247	2929

The results for 3-valued circuits are given in Table 1. The name of the benchmark is given in the first column and *PO* denotes the number of MDD nodes that were required for the representation of the POs. The variable ordering was chosen using interleaving [12]. In column *synthesis* we give the number of MIN gates and the number of MAX gates needed. These numbers are upper bounds computed by a traversal of the MDD. Further reductions are always possible due to the arguments given in Section 3 unless the fully reduced MDD representation was in the form of a $k-ary$ tree. As can be seen, four gates per node (or even less after reduction) are needed for a 3-valued function. Alternative methods that substitute the MDD nodes by look-up tables require at least 5-input cells when mapped to binary logic, three for the edges of the node and two for the control inputs. Further reductions can be expected, if more powerful minimization techniques are applied to the MDD such as sifting [16]. The circuit construction is very fast if the MDD is given and takes less than one CPU second for all examples, since only one traversal of the graph has to be carried out.

5. Conclusions

An MDD based method for synthesis of MVLNs has been presented. Instead of substituting each node by a multiplexer, an edge-mapping approach for MVLNs has been

developed. The resulting circuits have linear size with respect to the initial MDD. Furthermore, properties of the netlists have been discussed showing several advantages of this type of circuits over traditional approaches.

During the experimental study we observed that the number of MIN and MAX gates strongly depends on the variable ordering of the MDD and not only on the number of nodes (see e.g. *cs298* and *cs344* in Table 1). The focus of future work is to develop (dynamic) reordering algorithms that will effectively optimize the network (in terms of required circuitry) during MDD minimization. Furthermore, we are working on an efficient algorithm for removing all redundancies from the initially generated network.

References

[1] P. Ashar, S. Devadas and K. Keutzer. Path-delay-fault testability properties of multiplexor-based networks. *Integration the VLSI Jour.*, 15(1):1–23, 1993.

[2] B. Becker. *Synthesis for Testability: Binary Decision Diagrams*, volume 577 of *LNCS*. Symp. on Theoretical Aspects of Comp. Science, 1992.

[3] R.I. Bahar, E.A. Frohm, C.M. Gaona, G.D. Hachtel, E. Macii, A. Prado and F. Somenzi. Algebraic decision diagrams and their application. In *Int'l Conf. on CAD*, pages 188–191, 1993.

[4] P. Buch, A. Narayan, A.R. Newton and A.L. Sangiovanni-Vincentelli. On synthesizing pass transistor networks. In *Int'l Workshop on Logic Synth.*, 1997.

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

[6] F. Brglez, D. Bryan and K. Kozminski. Combinational profiles of sequential benchmark circuits. In *Int'l Symp. Circ. and Systems*, pages 1929–1934, 1989.

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

[8] E. Clarke, M. Fujita, P. McGeer, K. 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.

[9] D. Du, H. Yen and S. Ghanta. On the General False Path Problem in Timing Analysis. In *Design Automation Conf.*, pages 555–560, 1989.

[10] R. Drechsler. Verification of Multi-Valued Logic Networks. In *Multiple-Valued Logic - An International Journal*, Volume 3, Number 1, pp. 77-88, 1998

[11] R. Drechsler, R. Krieger and B. Becker. Random pattern fault simulation in multi-valued circuits. In *Int'l Symp. on multi-valued Logic*, pages 98–103, 1995.

[12] H. Fujii, G. Ootomo and C. Hori. Interleaving based variable ordering methods for ordered binary decision diagrams. In *Int'l Conf. on CAD*, pages 38–41, 1993.

[13] H. Hengster, R. Drechsler, S. Eckrich, T. Pfeiffer and B. Becker. AND/EXOR Based Synthesis of Testable KFDD-Circuits with Small Depth. In *Proc. of the Asian Test Symposium*, pages 148–154, 1996.

[14] N. Ishiura. Synthesis of Multi-level Logic Circuits from Binary Decision Diagrams. In *Proc. of SASIMI*, pages 74–83, 1992.

[15] L. Lavagno, P. McGeer, A. Saldanha and A.L. Sangiovanni-Vincentelli. Timed Shannon Circuits: A Power-Efficient Design Style and Synthesis Tool. In *Design Automation Conf.*, pages 254–260, 1995.

[16] R. Rudell. Dynamic variable ordering for ordered binary decision diagrams. In *Int'l Conf. on CAD*, pages 42–47, 1993.

[17] E. Sentovich, K. Singh, L. Lavagno, Ch. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephan, R. Brayton and A. Sangiovanni-Vincentelli. SIS: A system for sequential circuit synthesis. Technical report, University of Berkeley, 1992.

[18] T. Sasao and J.T. Butler. Planar multiple-valued decision diagrams. In *Int'l Symp. on multi-valued Logic*, pages 28–35, 1995.

[19] 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.

[20] R. Stanković and R. Drechsler. Circuit Design from Kronecker Galois Field Decision Diagrams for Multiple-Valued Functions. In *Int'l Symp. on Multiple-Valued Logic*, pages 275–280, 1997.

[21] M.A. Thornton and D.M. Wessels. Logic Synthesis Based on the Structure of an Ordered DD. In *Int'l Workshop on Logic Synthesis*, pages 21–25, 1999.