Partial Binary Decision Diagrams

Whitney J. Townsend and Mitchell A. Thornton
Mississippi State University
Mississippi State, MS
wjt1@ece.msstate.edu and mitch@ece.msstate.edu

***Abstract*--Decision diagrams provide compact representations for discrete functions. There are some functions for which binary decision diagrams reach exponential size. Presented here is a method of representing Boolean functions as multiple partial decision diagrams.**

***Index Terms*-- binary decision diagrams, partial binary decision diagrams**

I. Introduction

Binary decision diagrams provide compact representation for discrete functions. [3, 5] For this reason, these diagrams have seen extensive use in VLSI CAD. For some functions however, the binary decision diagram representation grows exponentially large with respect to the number of inputs.

Partial binary decision diagrams have been studied in other works as a method of determining an efficient variable ordering for binary decision diagram construction [6,8,7]. Presented here is a technique for partitioning a binary decision diagram into multiple binary decision diagrams each representing a subset of the information contained by the complete binary decision diagram for the function.

The organization of this paper is as follows. Part II discusses background information pertaining to the construction and implementation of binary decision diagrams. Part III explains and illustrates partial binary decision diagrams. Part IV presents experimental results for benchmark circuits represented as multiple partial binary decision diagrams. Part V provides concluding remarks on the work presented and discusses future work.

II. Background

A *binary decision diagram (*BDD) is a directed acyclic graph, *G = (V, E).* Every BDD has two different types of vertices, terminal vertices and non-terminal vertices. The terminal vertices represent the Boolean values, 0 and 1, while the non-terminal vertices represent variables of the function represented by the BDD. Each non-terminal vertex has exactly two outgoing edges, one of which is labeled by the Boolean constant 1 (or then) and the other by the Boolean constant 0 (or else). The graph begins at a single non-terminal node, known as the root, which has no incoming edges. Two very important properties that a BDD may have are to be ordered and to be reduced. These properties allow a BDD representation to be canonical for a given variable ordering. An ordered BDD is one in which each variable is encountered no more than once in any path and always in the same order along each path. A reduced BDD observes the following two properties. First, there are no redundant nodes in which both of the two edges leaving the node point to the same next node present within the graph. If such a node exists it is removed and the incoming edges redirected to the following node. Second, isomorphic subgraphs are shared, that is, if two nodes point to identical subgraphs, rather than repeat both subgraphs, the two nodes point to the same subgraph. A BDD that is both ordered and reduced is called a *reduced ordered binary decision diagram* (ROBDD). In this paper all references to BDDs shall imply ROBDDs. A BDD for the function, $f = xy + z$, is shown in Figure 1.

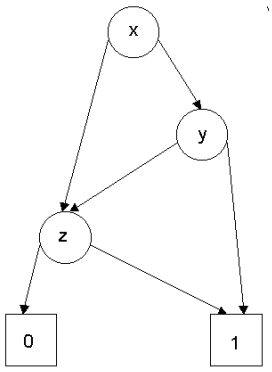

Figure 1. BDD example for function, $f = xy + z$

BDDs are produced by first creating individual BDDs for each variable of the function and then using the APPLY operation to build the BDD from the variable BDDs. The APPLY operation requires two BDDs and a Boolean operation to be performed. One efficient way to implement APPLY is to use the ITE operator (If-Then-Else). [1] The ITE operation is a recursive form of the Shannon decomposition theorem given below.

$$f = \overline{x}_1 f_0 \oplus x_1 f_1 \quad (1)$$

If $f = Z$, $x = f$, $f_0 = h$, and $f_1 = g$, then the Shannon decomposition shown in Equation 1 can be expressed as the following ITE in which, if ($f = 1$), then (g), else (h).

This work was supported in part by the NSF under grant CCR-0000891.

$$Z = ite(f, g, h) \quad (2)$$

The terminal cases of this recursion are as follows.

$$f = ite(1, f, g) = ite(0, g, f) = ite(f, 1, 0) \quad (3)$$

The complement of a BDD is formed by the following ITE expression.

$$ite(f, 0, 1) = \overline{f} \quad (4)$$

All two variable Boolean operators can be implemented as an ITE expression as shown in Table 1. An example of pseudo-code for ITE is shown in Figure 2.

While BDDs are compact representations for many functions, they can reach exponential size in regard to the number of inputs for some functions. There are several reasons why this occurs. One of the reasons is that a bad ordering was chosen for the variables when the BDD was created. Finding the best ordering for the variables during BDD creation has been proven to be NP-complete. [2] Therefore although heuristic techniques are used, exponential sizes can still occur. Another reason is that there exist circuits, such as multipliers, for which the BDD will always reach exponential size. [4]

TABLE 1. ITE FORMS

	Expression	ITE Expression
0000	0	0
0001	$f \bullet g$	$ite(f, g, 0)$
0010	$f \bullet \overline{g}$	$ite(f, \overline{g}, 0)$
0011	f	f
0100	$\overline{f} \bullet g$	$ite(f, 0, g)$
0101	g	g
0110	$f \oplus g$	$ite(f, \overline{g}, g)$
0111	$f + g$	$ite(f, 1, g)$
1000	$\overline{f + g}$	$ite(f, 0, \overline{g})$
1001	$\overline{f \oplus g}$	$ite(f, g, \overline{g})$
1010	$\overline{g}$	$ite(g, 0, 1)$
1011	$f + \overline{g}$	$ite(f, 1, \overline{g})$
1100	$\overline{f}$	$ite(f, 0, 1)$
1101	$\overline{f} + g$	$ite(f, g, 1)$
1110	$\overline{f \bullet g}$	$ite(f, \overline{g}, 1)$
1111	1	1

III. METHODOLOGY

A method for creating a *partial binary decision diagram* (pBDD) has been developed. This method is invoked during the creation of a BDD from a netlist description; therefore the entire BDD is never created. This method employs the notion of a third terminal node within the BDD that contains the unrepresented portion of the function, and is known as an *unknown* (U) terminal. An example of a function represented completely and then by two pBDDs is shown in Figure 3. Note that the multiple constant terminals shown were added to simplify the illustration. In the actual BDDs only two constant nodes are present.

```
ite (f, g, h) {
 if (terminal) {
    return result;
} else {
     let x be the top variable of (f, g, h);
    T = ite( f_x , g_x , h_x );
    E = ite( f_x̄ , g_x̄ , h_x̄ );
    if T = E, return T;
   R = newnode (x, T, E);
   return (R);
   }
}
```

Figure 2. ITE algorithm

Generation of the pBDDs is achieved by modifying the code implementing the ITE function that is used during diagram creation and additionally by restricting the number of nodes created. By altering these modifications it is possible to create several pBDDs each representing a portion of the functionality of the circuit.

IV. EXPERIMENTAL RESULTS

Experimental results presented here have been computed on a SUN Ultra 10. The modifications to the ITE algorithm have been implemented using the *Colorado University Decision Diagram* (CUDD) package. [9] The CUDD ITE functions are highly optimized and six separate locations were chosen within the code at which to make the modifications, producing six distinct pBDDs for each circuit.

In all over one hundred benchmark circuits were tested using the modified code. Table II. provides a summary of the results obtained for several benchmark circuits using the modified ITE code. These circuits were chosen for inclusion in Table II based upon the number of nodes in the complete BDD, selecting those circuits with over 500 nodes. The column labeled BDD shows the number of nodes resulting if the circuit was built completely, while each subsequent column shows the results from a different modification to the ITE algorithm during diagram creation.

V. CONCLUSION

A review of BDD construction and implementation techniques was presented. Two possible reasons why a BDD might reach exponential size were discussed as motivation for

the present work. A method for creating pBDDs was described in which each pBDD created represents a portion of the functionality of a circuit. This method redirects a portion of the functionality of the circuit to a third terminal node, U. Experimental results were shown for several circuits represented as multiple pBDDs.

Future work will involve improving the modifications to the ITE algorithm to decrease the overlap among the pBDDs and to prevent those occasions in which the altered ITE algorithm code actually generates a pBDD larger than the original complete BDD. The optimal solution would be to have a few small pBDDs that are capable of forming a disjoint set completely representing the functionality of the circuit.

REFERENCES

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

[2] B. Bollig and I. Wegener, "Improving the variable ordering of obdds is np-complete", *IEEE Trans. on Comp.*, 45:993-1002, 1996.

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

[4] R. E. Bryant, "On the complexity of vlsi implementations and graph representations of boolean functions with application to integer multiplication", *IEEE Trans. on Comp.*, 40:205-213, 1991.

[5] R. Drechsler and B. Becker. *Binary Decision Diagrams - Theory and Implementation.* Kluwer Academic Publishers, 1998.

[6] S. Friedman and K. Supowit, "Finding the optimal variable ordering for binary decision diagrams", in *Design Automation Conf.*, 348-356, 1987.

[7] A. Narayan, J. Jain, M. Fujita, and A. Sangiovanni-Vincentelli , "Efficient variable ordering and partial representation algorithms", in *Int'l. Conf. on VLSI Design*, 81-86, 1995.

[8] D. E. Ross, K. M. Butler, R. Kapur, and R. M. Mercer, "Fast function evaluation of candidate obdd variable orderings", in *European Conference of Design Automation*, 4-10, 1991.

[9] F. Somenzi. *CUDD: CU Decision Diagram Package Release 2.3.0*., http://vlsi.colorado.edu/~fabio/CUDD/cuddIntro.html, University of Colorado at Boulder, 1998.

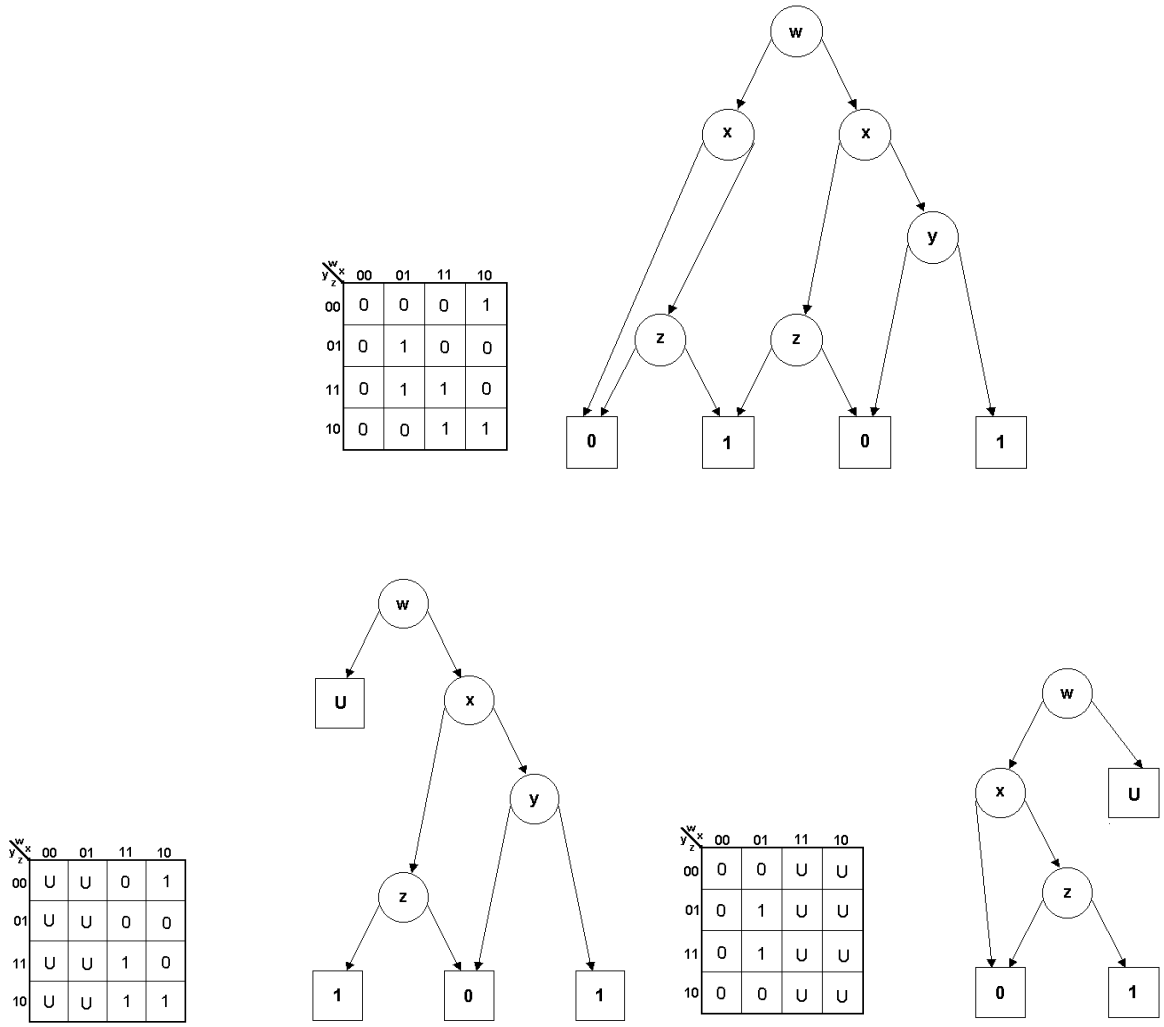

Figure 3. Completely Represented Function , Then As Two pBDDS

TABLE II. RESULTS

Circuit	BDD	pBDD1	pBDD2	pBDD3	pBDD4	pBDD5	pBDD6
alu4	933	83	294	81	73	271	87
apex1	1305	523	317	234	471	372	360
apex2	570	174	125	54	80	94	100
apex3	962	481	72	130	431	62	172
apex4	972	110	252	65	151	260	89
apex5	1095	115	369	224	234	442	200
apex6	744	405	89	320	410	61	301
bc0	587	44	113	58	30	101	47
cps	1096	471	519	396	349	454	438
dalu	1176	313	206	81	275	174	109
ex1010	1432	110	115	31	66	120	37
ex4	515	158	229	205	146	272	215
frg2	1396	405	211	226	462	212	260
intb	730	90	150	70	48	112	106
misex3	666	167	150	87	215	140	73
seq	1319	523	96	137	229	86	111
table3	786	72	430	168	117	404	112
table5	714	82	480	184	103	468	197
tial	929	77	205	105	101	205	101
vda	544	409	330	375	370	305	384
x1	626	288	437	352	197	452	348
x4	543	399	525	429	283	567	356