Efficient Variable Ordering Heuristics for Shared ROBDD*

Pi-Yu Chung, Ibrahim N. Hajj, and Janak H. Patel
Coordinated Science Laboratory and
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign

Abstract

In this paper we describe several ordering heuristics for shared, Reduced and Ordered Binary Decision Diagrams(ROBDDs). These heuristics have been tested on ISCAS and MCNC benchmark circuits. In all examples, the ordering is accomplished in a few seconds and generates smaller shared ROBDDs than other previously proposed heuristics. The objective is to provide a fast way to generate shared ROBDDs of reasonable sizes, and the results could be used as a good initial solution to any semi-exhaustive ordering method [2, 3] to further reduce the sizes.

1 Introduction

Reduced and Ordered Binary Decision Diagrams (ROBDD) proposed by Bryant [4] have been shown to be a practical way for representing boolean functions. ROBDDs have played an important role in logic synthesis, design verification and design correction [5]. In this paper, we are interested in generating shared ROBDDs for gate-level combinational circuits. Boolean functions at different primary outputs are represented in terms of primary input variables using the same variable ordering so that common subgraphs can be shared by different functions to save space. The ROBDD package we use is the one developed by Brace et. al. at Carnegie Mellon University [6], which adopts complement edges for reducing diagram sizes. Fig. 1(a) shows the ROBDDs of functions AB and $A+B$ and (b) shows the shared ROBDD of these two functions.

It is known that the sizes of ROBDDs are very sensitive to the order of input variables. Several ordering heuristics have been published to provide solutions to this problem [2, 3, 7, 8, 9, 10]. In this paper, we adopt guidelines that have been suggested to reduce the ROBDD sizes, but with new interpretation, to derive a set of six new heuristics. Since the heuristics are fast, we apply all of them to a given circuit and choose the ordering that gives a minimum shared ROBDD size. Experimental results on ISCAS and MCNC benchmark circuits show the effectiveness of our approach. Some semi-exhaustive approaches have been suggested to further minimize the ROBDD sizes by exchanging variables [2] or by applying simulated annealing [3]. The ordering obtained by our approach can provide a good initial solution to these methods to further reduce the size.

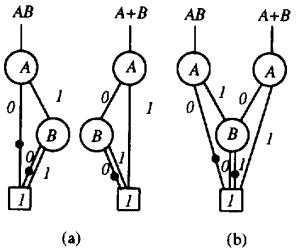


Figure 1: Shared ROBDD

2 Variable Ordering Guidelines

In this section, we state several guidelines that have been suggested in literature for finding a good variable ordering.

1. For a single-output tree network consisting of only primitive gates (AND, OR, NAND, NOR, XOR and NOT), one of the best ordering can be found by traversing the gates from the output to the inputs with depth-first order [7, 8]. Depth-first search has been considered a practical ordering method for most circuits.

*This research was supported in part by the Joint Services Electronics Program, contract number N00014-90-J-1270, and in part by Semiconductor Research Corporation, contract number SRC 92-DP-109.

2. The input that affects the output functions the most should be ordered first [10]. However, there is no exact solution to measure which input affects the output most.

3. Those inputs that are topologically near to one another in the circuit should be close in the order [10]. This rule is similar to the idea in [7] that the locations of the primary inputs in circuit diagrams imply a good ordering if the number of crosspoints is small.

In the following sections, we will adopt these guidelines to derive new heuristics.

3 PODEM-like Variable Ordering

PODEM is a test generation program [11], which finds a set of primary input (PI) assignments to justify a value v_k for line k. Given an *objective*(k, v_k), a backtracing procedure assigns values to a set of PIs that are likely to achieve the objective. The PODEM-like variable ordering assigns a set of objectives, i.e., to justify a value 1 at every primary output. The output that has the largest level, or the deepest depth from the primary inputs, is justified first. The backtrace routine is modified here so that it does not stop after the value is justified but keeps on backtracing until all the PIs that can reach the output are assigned. The input variable is ordered as it is assigned. The backtrace procedure is listed in Fig. 2.

This algorithm is depth-first in nature as proposed in guideline 1. Moreover, controllability is considered while choosing an edge to backtrace. The algorithm tends to order the input that is most likely to justify the output value first, which is a new way to interpret guideline 2. A variation to this algorithm is to disregard the value to be justified, but always backtrace first the gate input that is most uncontrollable.

3.1 Controllability Measures

We adopt two controllability measures, namely, level-based cost function and fanout-based cost function [12]. The controllability measures for a gate X are listed in Fig. 3, where n_i denotes the input nodes to the gate, f denotes the number of fanouts of the gate, and PI denotes primary inputs.

Example. Fig. 4 lists the cost function for a two-input multiplexer. To apply PODEM-like variable ordering using fanout-based cost function on this circuit, we assign the objective to be $(g4, 1)$. In backtracing

```
Backtrace(k, v_k)
begin
  if k is a PI, then append k to the ordering
  else
  begin
    while gate k has unvisited inputs
    begin
      if v_k = c ⊕ i
        then select the unvisited input j
          that is the most controllable
      else select the unvisited input j
          that is the most uncontrollable
      Backtrace(j, v_k ⊕ i)
    end
    return
  end
end
```

	AND	OR	NAND	NOR
Controlling value c	0	1	0	1
Inversion value i	0	0	1	1

Figure 2: Backtrace Procedure

$(g4, 1)$, we select the most controllable input to $g4$. Since $C1(g2)$ and $C1(g3)$ have the same value, $g2$ is chosen. In backtracing $(g2, 1)$, we select the most uncontrollable input to $g2$, which is C. Because C is a PI, so it is ordered first. Then by the depth-first-traversing, A and B are ordered. Fig. 5 shows the ROBDDs with arbitrary ordering (A, B, C) and with PODEM-like ordering (C, A, B).

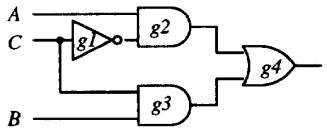
4 Distance-Based Variable Ordering

The distance-based variable ordering is inspired by guideline 3. The circuit diagram can be viewed as an undirected graph, where gates and primary inputs are treated as nodes, and an edge (u, v) exists if there is a wire connecting the output of gate u to the input of gate v. The distance between nodes u and v is the number of edges on the shortest path between u and v. In this heuristic, we find the all-pair distances among all primary inputs. The input which has the minimum sum of distances to the other inputs is ordered first because it is considered most likely to affect the output functions.

After the first input is ordered, we choose the unordered input which has the minimum distance to the ith input as the $(i + 1)$th input. If more than one unordered inputs have the same minimum distance to

Cost function	Measure	
Level-Based	$Level(PI) = 0$ $Level(X) = \max_i(Level(n_i)) + 1$	
Fanout-Based	$C0(PI) = C1(PI) = f - 1$	
	INV	$C0(X) = C1(n_i) + f - 1$ $C1(X) = C0(n_i) + f - 1$
	AND	$C0(X) = \min_i(C0(n_i)) + f - 1$ $C1(X) = \sum_i(C1(n_i)) + f - 1$
	OR	$C0(X) = \sum_i(C0(n_i)) + f - 1$ $C1(X) = \min_i(C1(n_i)) + f - 1$

Figure 3: Controllability Measures



	Level-Based	Fanout-Based	
		C0	C1
A	0	0	0
B	0	0	0
C	0	1	1
g1	1	1	1
g2	2	0	1
g3	1	0	1
g4	3	0	1

Figure 4: Example circuit

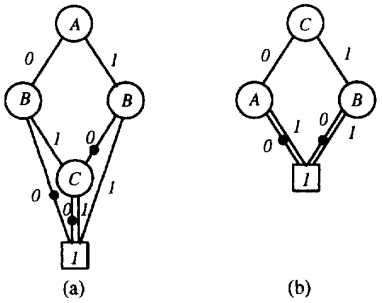
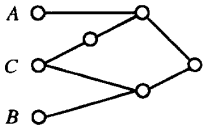


Figure 5: ROBDDs with different ordering



	A	B	C	SUM
A		4	3	7
B	4		2	6
C	3	2		5

Figure 6: All-pair distance

the ith input, two methods are used to break a tie.

1. For each of these inputs, we calculate the sum of the distances from it to every ordered input. The one with minimum sum is ordered first.

2. We choose the one that has the minimum distance to the $(i-1)$th input. If there is still a tie, the distances to the $(i-2)$th input are compared. Keep on this process until a tie is broken, or when the distances to all ordered inputs have been compared.

Example. We use the same example to demonstrate the distance-based variable ordering. Fig. 6 lists the all-pair distances among A, B and C and sum of distances for each. C has the minimum sum of distance to the other inputs, so C is ordered first. B is closer to C than A, so B is ordered before A. The ordering we get is (C, B, A), and the ROBDD is the same as the one in Fig. 5(b).

5 Experimental Results and Conclusions

We have tested the above heuristics on ISCAS and some large MCNC benchmark circuits. The depth-first-order heuristic proposed by Malik et. al. [8] and the dynamic weight assignment method proposed by Minato et. al. are also implemented for comparisons. We also include the ordering obtained by Mercer et. al. [3] using simulated annealing method as an "optimum" solution, although our heuristics give smaller ROBDDs for circuit C5315[1]. The sizes of shared ROBDDs representing all primary output functions are listed in Table 1 in terms of the number of nodes.

The execution time information is summarized as follows. The data are obtained from a SPARC-II station. The PODEM-like ordering (h1, h2, h3 and h4) took less than one CPU second in all cases. The distance-based methods (h5 and h6) have higher complexity because of calculating the all-pair distances, and took 35 CPU seconds for ordering the largest circuit *des*. The CPU times taken to build the ROBDDs are proportional to the ROBDD sizes. It took 200 CPU seconds to build the largest shared ROBDD (size 683676).

PODEM-like ordering heuristics are depth-first in nature, so their results are closer to, but better than, the results obtained by Malik's method in most cases.

[1]The cost function of the annealing process was targeted to reduce the size of OBDDs. The best ordering for an OBDD may not be the best ordering for an ROBDD[13].

Table 1: Shared ROBDDs sizes

Circuit	Mercer	Malik	Minato	PODEM-like				Distance-Based	
				Level-based		Fanout-based			
				U(h1)	C/U(h2)	U(h3)	C/U(h4)	TM1(h5)	TM2(h6)
C432	1859	39730	44834	46906	60701	46385	45203	**2602**	2653
C499	32979	40032	52518	40032	**39266**	40032	39498	41520	45042
C880	6398	15697	30948	**8292**	12552	16957	9538	28505	42012
C1355	32979	**40032**	52518	**40032**	59581	43403	63330	109692	97308
C1908	8433	33537	21411	17161	**12319**	31684	19971	27382	27798
C3540	90414	414356	408758	370109	261233	**196548**	422390	434010	496626
C5315	20686	38439	34112	35776	18740	683876	434969	42787	**11708**
b9		186	200	183	195	199	**182**	186	188
apex6		1157	1068	1162	1354	1119	1233	942	**666**
apex7		508	452	509	489	**449**	518	697	589
rot		7186	13124	7787	10351	**6627**	17316	15869	30945
des		7474	6072	7700	6766	6849	6112	**3810**	3828

Note: the minimum shared ROBDD sizes given by heuristics is illustrated in boldface. U : always choose the most uncontrollable. C/U : choose the most controllable or uncontrollable depending on value. TM1 : Tie breaking method 1. TM2 : Tie breaking method 2.

The distance-based methods are not very consistent, but give significantly smaller ROBDDs for circuits C432, C5315, *apex6* and *des*. Heuristic h3 ranks first in 3 out of 12 circuits, heuristic h1, h2, h5, h6 rank first each in two circuits and Malik's heuristic and h4 rank first in just one circuit. We conclude that it is not practical to try only one heuristic, but rather to try a set of fast heuristics on a given circuit and choose the best result.

References

[1] S. J. Friedman and K. J. Supowit, "Finding the Optimal Variable Ordering for Binary Decision Diagrams," *DAC-87,*, pp. 348-355, 1987.

[2] N. Ishiura, H. Sawada, and S. Yajima, "Minimization of Binary Decision Diagrams Based on Exchanges of Variables," *ICCAD-91,* pp. 472-475, 1991.

[3] M. R. Mercer, R. Kapur, and D. E. Ross, "Functional Approaches to Generating Orderings for Efficient Symbolic Representations," *DAC-92,* pp. 624-627, 1992.

[4] R. E. Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Trans. Computers,* vol. C-35, pp.677-691, Aug, 1986.

[5] P.-Y. Chung and I. N. Hajj, "ACCORD: Automatic Catching and Correction of Logic Design Errors in Combinational Circuits," *Proc. ITC-92,* pp. 742-751, 1992.

[6] K. S. Brace, R. L. Rudell, and R. E. Bryant, "Efficient Implementation of a BDD Package," *Proc. ACM/IEEE DAC-90,* pp. 40-45, 1990.

[7] M. Fujita, H. Fujisawa, and N. Kawato, "Evaluation and Improvements of Boolean Comparison Method Based on Binary Decision Diagrams," *ICCAD-88,* pp.2-5, 1988.

[8] S. Malik, A. R. Wang, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Logic Verification using Binary Decision Diagrams in a Logic Synthesis Environment," *ICCAD-88,* pp.6-9, 1988.

[9] K. M. Butler, D. E. Ross, R. Kapur, and M. R. Mercer, "Heuristics to Compute Variable Orderings for Efficient Manipulation of Ordered Binary Decision Diagrams," *DAC-91,* pp. 417-420, 1991.

[10] S. Minato, N. Ishiura, and S. Yajima, "Shared Binary Decision Diagram with Attributed Edges for Efficient Boolean Function Manipulation," *DAC-90,* pp. 52-57, 1990.

[11] P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits," IEEE Trans. on Computers, Vol C-30, No. 3, pp. 215-222, March, 1981.

[12] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital Systems Testing and Testable Design,* Computer Science Press, pp.215-218, 1990.

[13] M. R. Mercer, personal communication, Dec 1992.