Binary Decision Diagram with Minimum Expected Path Length

Yi-Yu Liu Kuo-Hua Wang† TingTing Hwang C. L. Liu

Department of Computer Science, National Tsing Hua University, Hsinchu 300, Taiwan
†Dept. of Computer Science and Information Engineering, Fu Jen Catholic University, Taipei, Taiwan

Abstract

We present methods to generate a Binary Decision Diagram (BDD) with minimum expected path length. A BDD is a generic data structure which is widely used in several fields. One important application is the representation of Boolean functions. A BDD representation enables us to evaluate a Boolean function: Simply traverse the BDD from the root node to the terminal node and retrieve the value in the terminal node. For a BDD with minimum expected path length will be also minimized the evaluation time for the corresponding Boolean function. Three efficient algorithms for constructing BDDs with minimum expected path length are proposed.

1. Introduction

Binary Decision Diagram (BDD) was introduced in 1978 [1] as an effective way to represent Boolean functions that leads to efficient manipulation [2] and implementation [3]. Given a random input pattern, we can traverse the BDD to determine the outputing. There are previous works on minimizing the number of nodes in a BDD [4] [5] [6]. However, the number of nodes is not directly related to the evaluation time for a Boolean function. Rather, the evaluation time is directly related to the total expected path length of a BDD.

In a logic circuit, the probabilities of occurance of the input variables are different. It follows that the evaluation time for different input combinations are also different. Consequently, a BDD with minimum expected path length will also have minimum evaluation time. To obtain a BDD with minimum expected path length, we need to develop (1) methods for computing the expected path length of a BDD (2) heuristics for ordering the variables in a BDD to minimize the expected path length.

This paper is organized as follows. Besides Section 1 being an introduction, we will discuss basic concept of BDDs in Section 2. In Section 3, we will propose three methods to compute the expected path length of an ROBDD. Section 4 deals with heuristics to determine variable ordering so that the resultant ROBDD will have minimum expected path length. Finally, experiment results will be presented.

2. Preliminary

2.1. ROBDD

A BDD is a directed, acyclic graph which consists of nodes and edges. There are two types of nodes, terminal and non-terminal. A terminal node is a Boolean value $\{0\}$ or $\{1\}$, and a non-terminal node is an input Boolean variable. Each non-terminal node contains two outgoing edges pointing to other nodes, which is referred to as the *parent-node* of these nodes. For convenience, we distinguish the two edges as *then-edge* and *else-edge*. A node which is connected from a then-edge is the *then-child* of its parent-node and from an else-edge the *else-child*.

A Reduced Ordered BDD (ROBDD) [2] is a BDD with a chosen variable ordering in all paths from the root node to the terminal nodes and possesses the following properties: (1) if both the then-child and else-child of a non-terminal node are identical, the non-terminal node will be removed from the path, and (2) if there are two or more identical subtrees, only one will be retained and all parent-nodes which have the same subtree will share one subtree as their children.

The formula representation of an ROBDD is "if *parent-node* then *then-child* else *else-child*", which is called an *ITE* operator (IF-THEN-ELSE). By using the ITE operator, Boolean operations can be easily performed on ROBDDs. Figure 1 gives an ROBDD for the function $F = A + B \cdot C'$ with the input variable ordering $A < B < C$. Note that for the same function, different ordering of input variables will result in different ROBDDs.

2.2. Expected Path Length of ROBDD

Since the evaluation time of a ROBDD is directly related to the path length of ROBDD, we want to shorten the path lengths with high probability, i.e., to minimize the expected path length of the ROBDD. It is necessary that we be able to compute a expected path length of a given ROBDD.

To begin with, we define the expected length of the ith path in ROBDD as $P_i \cdot L_i$, where P_i is the probability of the ith path and L_i is the path length of the ith path. The total Expected Path Length (EPL) of a ROBDD rooted at node r is defined as

$$EPL(r) = \sum_{i=1}^{m} P_i \cdot L_i \quad (1)$$

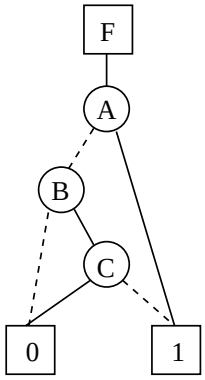

Figure 1. An ROBDD example.

where m is the number of paths of the ROBDD. Following this formula, to calculate the expected path length will need directly enumerate all paths. This approach is impractical because the number of paths may grows exponentially. In the following section, we will propose three efficient methods to calculate expected length: bottom-up method, top-down method, and middle-way method.

3. Three Methods for Computing EPL

Since it is impractical to enumerate all paths in a ROBDD. We present three efficient methods in this section for computing the EPL of a ROBDD. They are bottom-up method, top-down method, and middle-way method. The first and second methods need to be executed only once over all nodes. After initializing, the third method will be activated to improve the performace of our algorithm.

3.1. Bottom-up Method

Bottom-up method is formulated by which the expected path length can be calculated from terminal nodes up to root node. Since at each non-terminal node, the expected path lengths of its two child-nodes are already computed, we can compute the expected path length of the non-terminal node based on the expected path lengths for its child-nodes. Before we proceed to develop the formula, we define the following notations as:

- [$BEPL(a)$] the expected path length of all paths from a to terminal nodes
- [BP_i^a] the probability of the ith path from node a to a terminal node
- [BL_i^a] the path length of the ith path from node a to a terminal node

Now, let a be a non-terminal node with its then-child a_1 and else-child a_0 as shown in Figure 2. The expected path lengths of all paths from terminal nodes up to a_1 and a_0 can supposedly be computed as

$$BEPL(a_1) = \sum_{i=1}^{m_{a_1}} BP_i^{a_1} \cdot BL_i^{a_1} \quad (2)$$

$$BEPL(a_0) = \sum_{i=1}^{m_{a_0}} BP_i^{a_0} \cdot BL_i^{a_0} \quad (3)$$

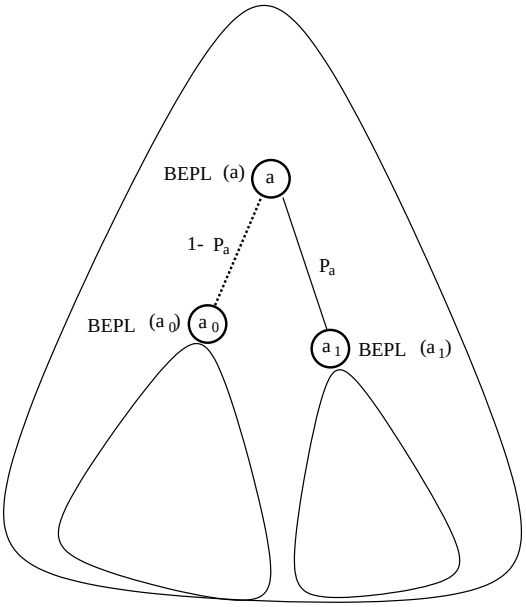

Figure 2. Bottom-up method for calculating the expected path length.

where m_{a_1} and m_{a_0} are the total number of paths from terminal nodes up to node a_1 and a_0, respectively.

Let the probability from node a to a_1 be p_a. Then, the probability from node a to a_0 will be $1 - p_a$. The expected path length from terminal nodes up to nodes a is computed as

$$\begin{aligned} BEPL(a) &= \sum_{i=1}^{m_{a_1}} p_a \cdot BP_i^{a_1} \cdot (BL_i^{a_1} + 1) + \\ &\quad \sum_{i=1}^{m_{a_0}} (1 - p_a) \cdot BP_i^{a_0} \cdot (BL_i^{a_1} + 1) \\ &= p_a \cdot (\sum_{i=1}^{m_{a_1}} BP_i^{a_1} \cdot BL_i^{a_1} + \sum_{i=1}^{m_{a_1}} BP_i^{a_1}) + \\ &\quad (1 - p_a) \cdot (\sum_{i=1}^{m_{a_0}} BP_i^{a_0} \cdot BL_i^{a_0} + \sum_{i=1}^{m_{a_0}} BP_i^{a_0}) \end{aligned} \quad (4)$$

Note that the total path probability from a non-terminal node to terminal nodes is 1. That is,

$$\sum_{i=1}^{m_{a_1}} BP_i^{a_1} = \sum_{i=1}^{m_{a_0}} BP_i^{a_0} = 1 \quad (5)$$

From the identity of equations (2) (3) (5), we can reduce equation (4) to

$$\begin{aligned} BEPL(a) &= p_a \cdot (BEPL(a_1) + 1) + \\ &\quad (1 - p_a) \cdot (BEPL(a_0) + 1) \\ &= p_a \cdot BEPL(a_1) + \\ &\quad (1 - p_a) \cdot BEPL(a_0) + 1 \end{aligned} \quad (6)$$

From equation (6), we follow that the expected path length of a non-terminal node at a can be computed by following steps :

step1 multiply the probability of the then-child by the expected path length of the then-child which is computed in the previous stage and is readily used.

step2 multiply the probability of the else-child by the expected path length of the else-child which is computed in the previous stage and is readily used.

step3 sum results of step1, step2, and constant 1

With boundary condition that both the path length of terminal nodes 0 and 1 are 0, we can recursively calculate the expected length of the root of an ROBDD by bottom-up method.

3.2. Top-down Method

Reversely, top-down method is used to calculate the expected path length from root node down to terminal nodes. Since at each non-terminal node, the expected path lengths of its parent nodes are already computed, we can compute the expected path length of an non-terminal node based on the expected path lengths for its parent nodes. Before we proceed to develop the formula, we define the following notations as:

[$TEPL(b)$] the expected path length of all paths from root to non-terminal node b

[TP_j^b] the probability of the jth path from root to node b

[STP^b] the sum of probabilities of all paths from root to node b

[TL_j^b] the path length of the jth path from root to node b

Obviously, from the above definitions, equations (7) (8) hold. That is, the summation of probabilities of all paths from root to a node b_k is

$$STP^{b_k} = \sum_{j=1}^{m_{b_k}} TP_j^{b_k} \tag{7}$$

where m_{b_k} is the total number of paths from root to node b_k. Moreover, the expected path length from root to a node b_k is

$$TEPL(b_k) = \sum_{j=1}^{m_{b_k}} TP_j^{b_k} \cdot TL_j^{b_k} \tag{8}$$

Now, let b be a non-terminal node with its n parent-nodes labeled b_1 to b_n and the probabilities from parent nodes b_1 to b_n to node b be p_{b_1} to p_{b_n}, respectively, as shown in Figure 3. We are to compute the expected path length of all paths from root to node b.

First, the summation of probabilities of all paths from root node to node b is

$$STP^b = \sum_{i=1}^{n} p_{b_i} \cdot STP^{b_i} \tag{9}$$

where n is the number of parent-nodes of node b. Now, the expected path length from root to node b is

$$\begin{aligned} TEPL(b) &= \sum_{i=1}^{n} \sum_{j=1}^{m_{b_i}} p_{b_i} \cdot TP_j^{b_i} \cdot (TL_j^{b_i} + 1) \\ &= \sum_{i=1}^{n} \sum_{j=1}^{m_{b_i}} (p_{b_i} \cdot TP_j^{b_i} \cdot TL_j^{b_i} + p_{b_i} \cdot TP_j^{b_i}) \end{aligned} \tag{10}$$

By equations (7) (8) and (9), equation (10) can be reduced to

$$\begin{aligned} TEPL(b) &= \sum_{i=1}^{n} (p_{b_i} \cdot TEPL(b_i) + p_{b_i} \cdot STP^{b_i}) \\ &= \sum_{i=1}^{n} p_{b_i} \cdot TEPL(b_i) + STP^b \end{aligned} \tag{11}$$

From equation (11), we follow that the expected path length of a non-terminal node at b can be computed by following steps:

step1 for all parent nodes, multiply the probability of parent-node to the node b by the expected path length from root to the parent node which is computed in the previous stage and is readily used.

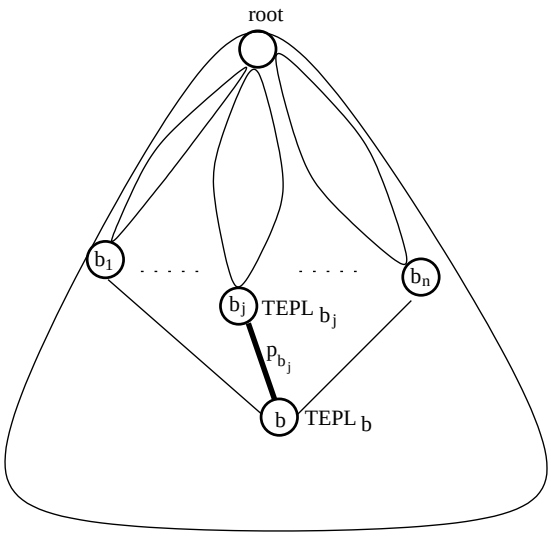

Figure 3. Top-down method for calculating the expected path length.

step2 sum probabilities of all paths from root to node b by equation (9), where the sum of probabilities of all path from root to a parent-node is already computed and readily used.

step3 sum results of step1 and step2

Based on the assumptions that the path length of root is 0 and the probabilities from all paths to root is 1, we can calculate the expected length of the terminal nodes of an ROBDD by top down method. Finally, the expected path length of the whole ROBDD is the summation of the expected path lengths of the two terminal nodes.

3.3. Middle-way Method

The third method is used to compute the expected path length of an ROBDD assuming that the expected path lengths of all non-terminal nodes at level $h-1$ have been computed by top-down method and the expected path lengths of all non-terminal nodes at level h have been computed by bottom-up method. Let C be the cut between levels $h-1$ and h and there are l edges crossing C that connect the nodes at levels $h-1$ and h as shown in Figure 4. To compute the expected path length of the whole ROBDD, we have to consider all paths connected by l edges. First, we consider how to compute the expected path length connected by a single crossing edge k. In this case, edge k connects the top end-node b_k and the bottom end-node a_k as shown in Figure 4. Suppose that the probability of edge k be p_k and there are n paths from root node to node b_k, and m paths from terminals nodes to a_k as shown in Figure 4. Then, the expected path length of all paths, $MEPL(k)$, which pass through edge k is

$$\begin{aligned} MEPL(k) &= \sum_{i=1}^{m} \sum_{j=1}^{n} p_k \cdot BP_i^{a_k} \cdot TP_j^{b_k} \cdot \\ &\quad (BL_i^{a_k} + TL_j^{b_k} + 1) \\ &= p_k \cdot (\sum_{i=1}^{m} \sum_{j=1}^{n} BP_i^{a_k} \cdot TP_j^{b_k} \cdot BL_i^{a_k} + \\ &\quad \sum_{i=1}^{m} \sum_{j=1}^{n} BP_i^{a_k} \cdot TP_j^{b_k} \cdot TL_j^{b_k} + \\ &\quad \sum_{i=1}^{m} \sum_{j=1}^{n} BP_i^{a_k} \cdot TP_j^{b_k}) \end{aligned}$$

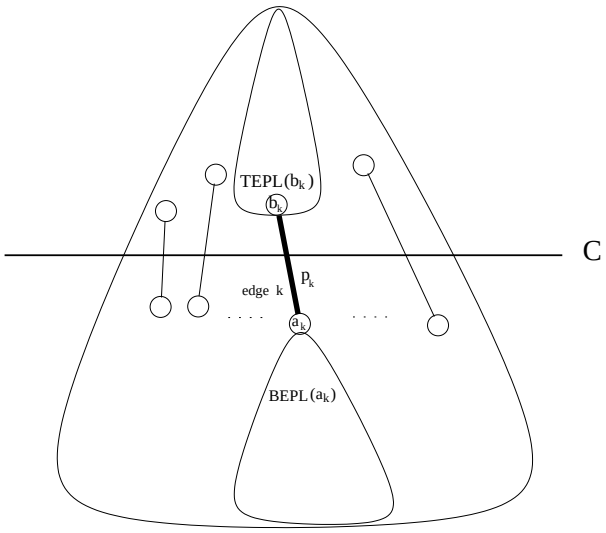

Figure 4. Middle-way method for calculating the expected path length.

$$= p_k \cdot (\sum_{j=1}^{n} TP_j^{b_k} \cdot \sum_{i=1}^{m} BP_i^{a_k} \cdot BL_i^{a_k} + \sum_{i=1}^{m} BP_i^{a_k} \cdot \sum_{j=1}^{n} TP_j^{b_k} \cdot TL_j^{b_k} + \sum_{i=i}^{m} BP_i^{a_k} \cdot \sum_{j=1}^{n} TP_j^{b_k}) \quad (12)$$

By equations (2)(5)(8)(9), equation (12) can be reduced to

$$MEPL(k) = p_k \cdot (STP^{b_k} \cdot BEPL(a_k) + TEPL(b_k) + STP^{b_k}) \quad (13)$$

From equation (13), we follow that the expected path length of all paths passing through edge k connecting node b_k and a_k can be computed by following steps:

step1 sum the expected path length from root of all paths to node b_k, the expected path length of all paths from node a_k to terminals and twice of the total path probability to node b_k. All of them are already computed and are readily used.

step2 multiply the result of step1 and the probability of edge k

Finally, the total expected length of the whole ROBDD is

$$EPL = \sum_{k=1}^{l} MEPL(k)$$

4. Algorithms for Finding Low Cost ROBDD

Our algorithm begins with an initial ordering. Then, an iterative loop is entered to modify the initial ordering. In each iteration, a new ordering is determined and the old ROBDD is restructured based on the new variable ordering. Next, the cost for the new ROBDD is computed. The loop continues till the stopping criterion is met. The procedure is shown in Figure 5. The detailed description is explained as follows.

In step 1, an initial ordering needs to be determined to construct an initial ROBDD. It can be determined by constructing a minimum size ROBDD. In this case, algorithms

Algorithm *find-least-cost*(f)
Input: f = Boolean function;
Output: return $cost$ and $the\ corresponding\ ROBDD$;
Begin
(1) find an initial variable ordering for ROBDD of f;
(2) build an initial ROBDD;
(3) compute the cost of the ROBDD;
(4) **while** (not stop) **do**
(5) find new variable ordering;
(6) restructure the ROBDD according to the new variable ordering;
(7) compute the cost of the new ROBDD;
(8) update the cost function;
(9) check if the stopping criterion is set;
endwhile
(10) **return** $cost$ and $the\ corresponding\ ROBDD$;
End

Figure 5. The *find-least-cost* algorithm

proposed in [5] [6] can be used. Step 3 is to compute the cost of the initial ROBDD. For the computation of the expected path length of the ROBDD, bottom-up and top-down methods proposed in Section 3 are used and all computed data for each node is stored at the node. Step 5 finds a new input ordering. Window permutation algorithm [5] [6] which finds a local minimum in a window of size k and sifting algorithm [4] which looks for a suitable position for each variable in each iteration can be used. With the new input variable ordering, step 6 restructures the old ROBDD to a new ROBDD. Transposition operator [7] is used which restructures an ROBDD by several simple ROBDD operations. For example, if the order of x_i and x_j are swapped in a new ordering, the following formula can be used.

$$f_{x_i \leftrightarrow x_j} = ITE(x_i \odot x_j, f, f_{x_i \leftarrow \bar{x_i}, x_j \leftarrow \bar{x_j}})$$

Note that swapping is not limited to adjacent variables.

In step 7, the cost for the new ROBDD is computed. For the expected length of the new ROBDD, similar to step 6, it needs not be computed from scratch. Instead, we will use the information for the old ROBDD to calculate the new expected length of ROBDD. Suppose that we have a new ordering that variables at levels i and j $(i < j)$ are swapped as shown in Figure 6. Since the part of ROBDD above level i and the part of ROBDD below level j will not be changed as seen in Figure 6, we do not have to recompute them. We only need to compute the new expected path length for the nodes in the middle. On the one side, the bottom-up method calculates the expected path length from nodes at level $j+1$ up. On the other side, the top-down method calculates the expected path length from nodes at level $i-1$ down. When variables on the two sides meet in the middle, the middle-way method can be used. By using middle-way method, it prevents much more redundant calculations than top-down and bottom-up methods.

5. Experimental Results

Our experiment is performed on a SUN-Ultra Enterprise 150. Software platform is based on the ROBDD package in

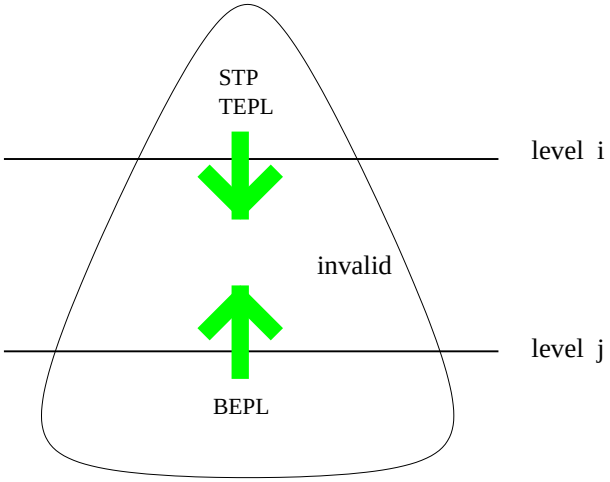

Figure 6. Computation for parts lying between the two swapped variables.

SIS [8]. *LGSynth93* benchmark suite is used in our experiment.

Experiment is conducted to find a minimum expected path length ROBDD. First, the sifting algorithm [4] is used to find a minimum-sized ROBDD which can be used as our initial ROBDD. The reason for this initial ordering heuristic is that a small size ROBDD may lead to a small expected length ROBDD. In subsequent steps, window permutation is used to generate a new input ordering and the expected path length for the new ROBDD is computed. Finally, the probabilities of input variables are set to be equal.

Table 1 shows the results. Column min_node gives the results of ROBDD with initial ordering whose objective is to find an ROBDD with minimum size. Column min_exp_path gives the results of ROBDD with expected path length as objective. Columns $path$ and $size$ give the expected path length and size of ROBDDs, respectively. In column *ratio*, we compute the ratio of the results of minimum nodes to the results of minimum expected path length.

It is clear from Table 1 that in most cases, the expected path length can be reduced at the cost of small increase of the number of nodes except *cordic* and *f51m*. In some cases (*con1*, *vg2*, *cm150a*, *etc.*), we found that both the size and the expected path length of ROBDD are reduced. On the average, the expected path length is reduced by 25% while the the number of nodes is increased by 10%.

Table 1. Results of expected path length with equal input probability

circuit	min_node		min_exp_path		ratio (%)	
	path	size	path	size	path	size
5xp1	37.22	82	31.31	91	61.94	110.98
alu4	62.84	736	47.54	899	75.65	122.15
b12	29.52	77	22.22	81	75.28	105.19
con1	6.44	17	6.06	16	94.17	94.12
cordic	17.08	98	11.82	259	69.19	264.29
ex1010	82.07	1478	80.46	1544	98.03	104.47
inc	31.80	104	27.61	104	86.83	100.00
sao2	21.93	107	10.71	128	48.85	119.63
vg2	46.46	237	30.37	230	65.36	97.05
misex1	23.59	64	22.16	68	93.91	106.25
cm150a	6.25	39	3.50	33	56.00	84.62
cm151a	9.00	34	6.50	36	72.22	105.88
cm162a	19.19	57	11.70	59	60.99	103.51
cm163a	18.09	46	11.70	42	64.68	91.30
cm85a	15.97	41	8.28	47	51.86	114.63
mux	3.73	38	3.50	33	93.72	86.84
z4ml	17.88	35	17.13	32	95.80	91.43
f51m	28.92	60	27.45	76	94.92	126.67
pcle	35.51	101	22.50	89	63.36	88.12
traffic	20.88	41	14.63	38	70.06	92.68
Average					74.64	110.49

6. Conclusions

In this work, we present methods to generate a ROBDD with minimized expected path length. With smaller EPL of a ROBDD, the evaluation time of a Boolean function will be reduced. Three methods are proposed to calculate the expected path length of a ROBDD efficiently. The experimental results indicate that the expected path length of an ROBDD can be reduced at the cost of a small increase of OBDD size.

References

[1] S. B. Akers, "Binary decision diagrams", *IEEE Trans. Comput.*, Vol. C-27, pp. 509-516, June 1978.

[2] R. E. Bryant, "Graph-based algorithms for boolean function manipulation", *IEEE Trans. on Comput.*, Vol. C-35, No. 8, August 1986.

[3] K. S. Brace, R. L. Rudell, and R. E. Bryant, "Efficient Implementation of a BDD Package", *DAC*, 1990.

[4] R. Rudell, "Dynamic variables ordering for ordered binary decision diagrams", *ICCAD*, 1993.

[5] M. Fujita, Y. Matsunaga, and T. Kakuda, "On Variable Ordering of Binary Decision Diagrams for the Application of Multi-level Logic Synthesis", *EDAC*, pp 50-54, Mar. 1991.

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

[7] K. H. Wang, T. T. Hwang, and C. Chen, "Restructuring Binary Decision Diagrams Based on Functional Equivalence", *EDAC*, pp 261-265, Feb. 1993.

[8] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, A. Sangiovanni-Vincentelli, "SIS: A System for Sequential Circuit Synthesis", *Electronics Research Laboratory* Memorandum No. UCB/ERL M92/41, 4 May 1992.