ROBDD Optimization Using Sub Graph Complexity

Mohamed Raseen and K.Thanushkodi

Coimbatore Institute of Engineering and Information Technology,
Vellimalaipattinam, Narasipuram (Post), Coimbatore, Tamilnadu 641109, India

Summary
This paper describes a novel method of variable reordering for Reduced Ordered Binary Decision Diagrams (ROBDD). The proposed method results in ROBDD with lesser number of nodes and lesser APL. The variable order in ROBDD is important since it affects the number of ROBDD nodes. The problem of constructing an ROBDD with minimum number of nodes has become of growing importance since there is no unique method that can be used to obtain the least number of nodes for all Boolean functions. In this work, the proposed variable ordering methods uses the graph topology to find the optimal variable ordering; therefore the input Boolean function (benchmark circuits) are converted to a unidirectional graph. The variable order is found by substituting the values of logic 1 and logic 0 for all the variables. The variable that produces minimal sub graph is assigned as next variable in variable order. This process of assignment and selection is repeated iteratively until all variables are selected. The efficiency of the proposed method is demonstrated by building the ROBDD for selected benchmark circuits. The number of nodes is then compared for proposed method with existing methods in Colorado University Decision Diagram (CUDD) package. The experimental results using benchmark circuits show that the proposed method is an encouraging approach towards minimizing the evaluation time and number of nodes for a Boolean function.

Key words:
Binary Decision Diagrams, Variable Ordering, Graph Representations, Boolean Functions Representations.

1. Introduction

For the last two decades Binary Decision Diagrams (BDD) has gained great popularity as a method for representing discrete functions. BDD in general is a direct acyclic graph representation of a Boolean function proposed by Akers and Bryant [1], [2]. The success of this technique has attracted many researchers in the area of synthesis and verification of digital VLSI circuits. Since BDD allow efficient representation of many practical functions [3], [4], BDDs have become very popular data structures. The efficiency of BDDs depends mainly on the size of their graph representations.

The size of the BDD dramatically depends on the chosen order of variables [5], [6], [7]. Finding a better variable order is often worth spending considerable computational effort [8]. Some functions, such as adders, lead to BDD sizes that are exponential to the number of input variables. But some other variable orderings lead to linear complexity for BDD sizes. Determining an optimal variable ordering is an NP-hard problem [9]. Another parameter critical during the construction of BDDs is the maximal memory requirement, which is directly proportional to the number of nodes. A good ordering can lead to a smaller BDD and faster runtime, whereas a bad ordering can lead to an exponential growth in the size of BDD and hence can exceed the available memory [10]. Accordingly, much attention has been devoted to techniques for finding a good variable ordering. All these variable ordering techniques fall into two categories: Static Variable Ordering (SVO) algorithms [11], [12] and Dynamic Variable Ordering (DVO) algorithms [6], [13].

The evaluation time is also another important parameter, when BDDs are used to evaluate logic functions. The evaluation time is proportional to the path length in the BDD. Therefore, minimization of the path length can improve the performance of the circuit, which will eventually increase the quality of the final implementation. In general the minimum path length in Decision Diagrams (DD) is important in databases, pattern recognition, logic simulation and software synthesis [14].The minimization of Average Path Length (APL) proposed in [14], [15], [16] reduces the average evaluation time of logic functions. The minimization of the APL leads to circuits with a smaller depth on the paths from Root to Terminal nodes. By this, the circuit is optimized for speed and the number of very long paths is reduced [17]. The APL minimization is very much effective in Real time operating system applications [18], [19], [20]. The minimization of Longest Path Length (LPL) of BDD can reduce the longest evaluation time which is more important for Pass Transistor Logic (PTL) [21], [21], [22]. One of the main problems with the pass transistor network is the presence of long paths: the delay of a chain of n pass transistors is proportional to n^2. The path length can be reduced by inserting buffers, but this increases area. Hence the minimization of longest evaluation time will

Manuscript received August 5, 2008.
Manuscript revised August 20, 2008.

improve the performance of the circuit [21], [22]. We proposed an algorithm for finding the optimal variable ordering for the minimization of BDD with regard to number of nodes and the Path length. The resulted initial variable order will produce the BDD with minimum possible APL and consequently reducing the number of nodes to an affordable size.

This paper is organized as follows. Besides section one being an introduction, we will discuss the necessary terminology and definitions in section two. In Section three, we propose the method to calculate the minimum APL and number of nodes of BDD based on good variable ordering. Section four explains the proposed method using an example. The experimental results are given in section five. Finally in section six we conclude our paper with an outline of our future work.

2. Preliminaries

Basic definitions for BDDs are given in [1], [2], [23], [24]. In the following we review some of these definitions.

Definition 1: A *BDD* is a directed acyclic graph (DAG). The graph has two sink nodes labeled 0 and 1 representing the Boolean functions 0 and 1. Each non-sink node is labeled with a Boolean variable *v* and has two out-edges labeled 1 (if *then*) and 0 (or *else*). Each non-sink node represents the Boolean function corresponding to its 1 edge if *v*=1, or the Boolean function corresponding to its 0 edge if v=0.

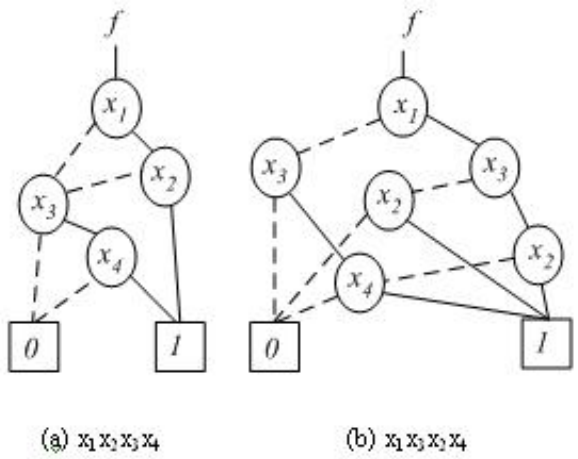
Definition 2: An *Ordered BDD* (OBDD) is a BDD in which each variable is encountered no more than once in any path. The order of variables is same along each path.

Definition 3: A *Reduced OBDD* (ROBDD) is an OBDD that is reduced by two reduction rules: *deletion rule* and *merging rule.* These Reduction rules remove redundancies from the OBDD.

2.1 *Variable Ordering*

The size of a BDD is largely affected (and varies from linear to exponential) by the choice of the variable ordering. Figure 1 illustrates the effect of the variable ordering on the size of BDDs [1] for the following Boolean function (1):

$$f = x_1 \cdot x_2 + x_1 \cdot \overline{x_2} \cdot x_3 \cdot x_4 + \overline{x_1} \cdot x_3 \cdot x_4 \qquad (1)$$



(a) $x_1x_2x_3x_4$ (b) $x_1x_3x_2x_4$

Figure 1- Effect of variable ordering on the size of BDDs

Definition 4: In a BDD, a sequence of edge and nodes leading from the root node to a terminal node is a *Path*. The number of non-terminal nodes on the path is the *Path Length*.

Definition 5: The *APL* is equal to the sum of the node traversing probabilities of the non-terminal nodes [14], [18], which give the following equation (2):

$$APL = \sum_{i=0}^{N-1} P(v_i) \qquad (2)$$

Where, *N* denotes the number of non-terminal nodes.

Definition 6: The edge traversing probability, denoted by $P(e_{i0})$ (or $P(e_{i1})$), is the fraction of all 2^n assignments of values to the variables whose path includes e_{i0} (or e_{i1}), where e_{i0} (or e_{i1}) denotes the 0-edge (or the 1-edge) directed from away node V_i [14]. Since all paths include the root node, this node is traversed with probability 1.00. Since all assignments to values of variables are equally likely, we can use the following equation (3) to calculate the $P(V_i)$ for the rest of the nodes:

$$\frac{P(vi)}{2} = P(e_{i0}) = P(e_{i1}) \qquad (3)$$

Definition 7: *The Longest Path Length (LPL)* of a BDD denoted by LPL (BDD), is the *Length of the Longest Path from the root to terminal node.*

Example 1: Consider the BDD graph given in Figure 2, we will calculate the APL in following order:

The root node $P(V_0)$ is always equal to 1.00. Then we calculate the $P(V_1) = P(e_{0_0}) = 0.50$ and $P(V_2) = P(e_{1_0}) = 0.50$. In a similar manner we calculate

$P(V_3) = P(e_{2_0}) = 0.25$

$P(V_4) = P(e_{21}) = 0.25$

$P(V_5) = P(e_{4_0}) + P(e_{1_1}) = 0.125 + 0.25 = 0.375$

So,

$APL = \sum_{I=0}^{5} P(V_i) = 2.875$

$LPL = LongestPath = x_1 \rightarrow x_3 \rightarrow x_2 \rightarrow x_4 = 4$

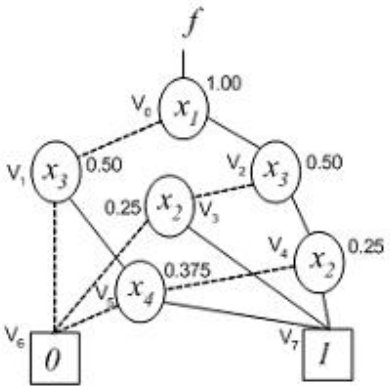


Figure 2- Node Traversing Probability in a BDD

Definition 8: In a Decision Diagram (DD) for logic function f , the *memory size of the DD,* denoted by *Mem (DD),* is the number of words needed to represent the DD in memory [18].

In a memory, each non-terminal node requires an index and pointers to the succeeding nodes. Since each non-terminal node in a BDD has two pointers, the memory size needed to represent a BDD is

$$Mem(BDD) = (2+1) \times nodes(BDD) \qquad (4)$$

3. Proposed Method

The proposed method is a static variable ordering technique [25], [26], [30]. The main focus of the research work is to develop methods to reduce the complexity of Boolean functions by finding optimal variable order for the corresponding ROBDD. The software that will be used for the entire research work is CUDD (Colorado university decision diagram)[27] package with C interface. The following steps will be performed in the proposed method

Step 1: Using the BLIF file, a graph with input, output and intermediate nodes is developed and stored in the RAM.

Step 2: The values of 0 is assigned to first variable and the graph is simplified.

Step 3: The parameters of the new graph (number of nodes) are recorded.

Step 4: The value of 1 is assigned to first variable and the graph is simplified.

Step 5: The parameters of the new graph (number of nodes) are recorded.

Step 6: Steps 2 to 5 are repeated for all the remaining variables.

Step 7: The assignment (1 or 0) that produced minimum graph parameter is selected as the variable in the order.

Step 8: Steps 2 to 7 are repeated for the new simplified graph in a recursive way and all the other variables in the order are found.

Step 9: Using the variable order generated, the ROBDD is built with minimum nodes.

Step 10: The method is repeated for benchmark circuits and results are tabulated to prove the efficiency of the proposed method.

4. Example

In the following we explain the proposed method mentioned in section 3 using an example. Consider the BLIF (Berkley logic interchange format) file shown in the figure 3.

```
.model basic
.inputs A B C D
.outputs Z
.names A B C D Z
0001 1
1111 1
1000 1
.end
```

Figure 3- Example BLIF file

The circuit in figure 3 has four inputs (A,B,C,D) and one output (Z). The circuit is converted into a graph shown in figure 4.

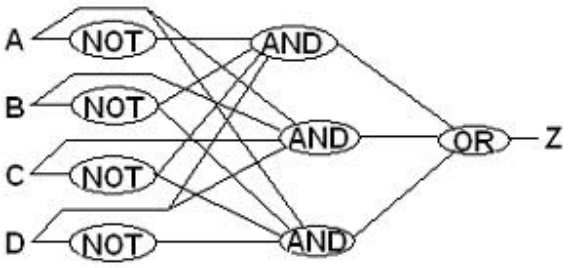


Figure 4- Graph for the example BLIF file

From figure 4 it can be seen that the complete BLIF function requires 8 nodes in the graph. To find the variable order, the variable A is substituted the value of logic zero and simplified. After substitution and reduction (of A=0) a new sub graph in figure 5 is obtained.

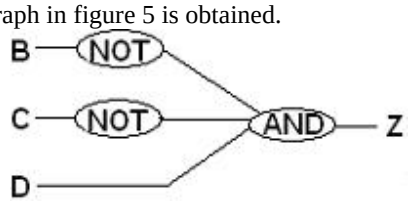


Figure 5-Sub graph for variable A = 0

From the sub graph (figure 5) for A=0 it can be seen that the complete graph reduces from 8 nodes to 3 nodes. Next the variable A is assigned the value of logic 1 (in figure 4) and simplified. After assignment (A=1) and simplification the figure 6 is obtained.

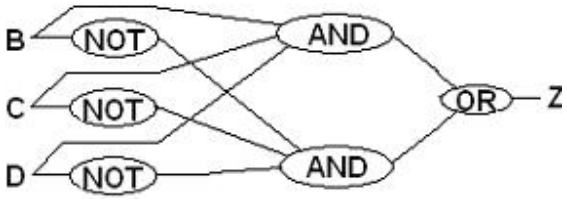


Figure 6- Sub graph for variable A = 1

For the substitution of variable A to the value of logic 1 the graph (figure 6) reduces to 6 nodes. After assigning values of logic one and zero for variable A, the variable B is assigned the values. From the graph of full circuit the variable B is assigned a value of logic 0. After assignment of B=0 the graph in figure 7 is obtained.

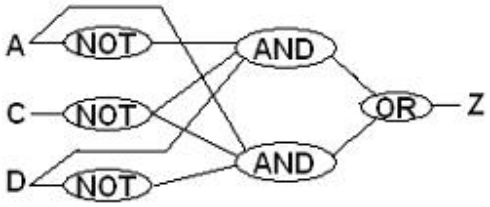


Figure 7- Sub graph for variable B = 0

From figure 7 it can be inferred that the substitution of logic zero to variable B gives a sub graph of 6 nodes. Next variable B is assigned the value of logic 1 and simplified. Thus figure 8 is obtained for B = 1.

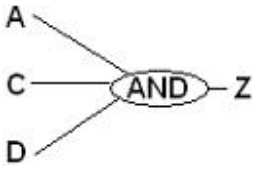


Figure 8- Sub graph for variable B = 1

By substituting logic 1 to variable B we get the sub graph (figure 8) with 1 node. Similarly values of logic 0 and logic 1 is assigned to the remaining variables (C,D) and sub graphs (figure 9 to figure 12) are obtained.

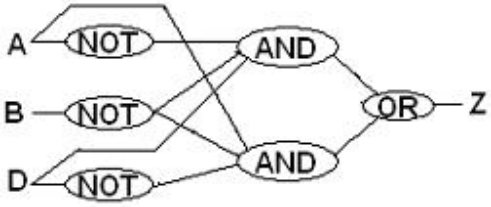


Figure 9- Sub graph for variable C = 0

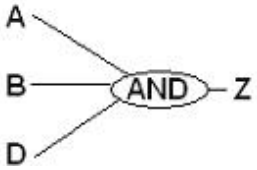


Figure 10- Sub graph for variable C = 1

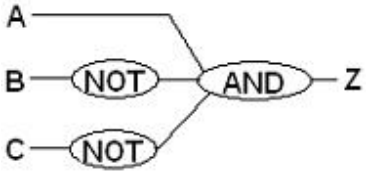


Figure 11- Sub graph for variable D = 0

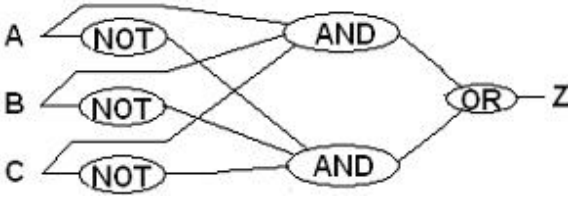


Figure 12- Sub graph for variable D = 1

The number of nodes in the sub graphs (after substitution and simplification) is summarized in table 1.

Table 1: Number of nodes in sub graphs

Variable	A		B		C		D	
Assigned Value	0	1	0	1	0	1	0	1
Number of nodes in sub graph	3	6	6	1	6	1	3	6

From the table we can infer that simplification occurs maximum to 1 node. The sub graph gets one node when B

is substituted with logic 1 or C is substituted with logic 1. The variable that produces minimum nodes is selected as next variable in the ROBDD variable order. In this case B is selected as first variable in order since it produces minimum nodes for assignment of logic 1. Variable C (C=1) also produced 1 node but is not selected since B precedes C in the substitution order.

To generate other variables in order we start wit the sub graph that produced minimum nodes (with B=1). Figure 8 is taken as parent graph and sub graphs are built with the remaining variables(A,C,D). This process is repeated iteratively until all the variables in the order are found.

The example shown in this section is a simple BLIF file with 4 variables and single output. The example is also a simple single level PLA structure. The proposed method explained in this section was applied to much complex ISCAS benchmarks with multi level circuits and with many intermediate nodes.

5. Experimental Results

In this section we present experimental results obtained for selected ISCAS benchmark circuits [28], [29] using the Colorado University Decision Diagram (CUDD) package. The CUDD provided interface for C programming. The proposed model was implemented with approximately 2000 lines of C program. Table 2 illustrates the results of the proposed method and eleven different CUDD methods. The results indicate the superiority of the proposed method, in terms of number of nodes. It is difficult to conduct a head-to-head comparison of different variables ordering techniques due to the fact that other parameters such as memory used while reordering, time taken to reorder, algorithm complexity etc. need to be considered when comparing the results. Based on the experimental results it can be inferred that the proposed method performs better than most of the CUDD methods for most of the circuits. Circuits alu2, b1, b12, c8, cc, cht, cm138a, cm162a, cmb, cordic, decod, lld, misex2, sqrt8, s1rt8ml, sqar5 and tcon produced lesser nodes than all the eleven methods in CUDD. The remaining circuits produced lesser nodes than most of the CUDD methods. Apart from the number of nodes the experiment was conducted for APL. Table 3 illustrates the APL generated for the benchmarks with the proposed method and eleven other CUDD methods. From the results in table 3, it can be inferred that the proposed method performs better in APL than most of the existing methods in CUDD.

6. Conclusion

A new algorithm for minimizing the Evaluation time in BDD has been developed. The algorithm has been implemented using ISCAS benchmark circuits and the results have been compared with the eleven CUDD reordering methods. Experimental results indicates that this algorithm is promising, yielding better results than more mature reordering techniques for most of the benchmarks. It is also quite clear that the minimization of the Evaluation time of a BDD can improve the performance of the circuit, and have a strong influence on the quality of the final implementation. Our future work and developments will concentrate on investigating LPL(Longest Path Length) and SPL(Shortest Path Length) minimization for larger scale benchmark circuits.

Table 2- Experimental Results (number of nodes)

Benchmark	Number of NODES (Existing CUDD methods)											Proposed Method
	Random	Random pivot	Sift	Sift converge	Symm sift	Symm sift converge	Group sift	Group sift converge	Window2	Annealing	Genetic	
5xp1	107	107	90	94	90	94	94	94	94	90	90	107
alu2	220	270	322	319	320	319	320	319	319	318	318	211
alu4	853	900	849	976	981	976	981	980	980	975	954	892
apex4	1410	1406	1458	1457	1458	1457	1458	1457	1457	1586	1586	1422
apex5	2051	1790	1879	1767	1767	1767	1762	1793	1793	3522	3536	1946
b1	12	12	12	12	12	12	12	12	12	12	12	11
b12	102	102	90	99	103	93	89	97	97	89	93	93
c8	154	154	146	146	146	146	146	146	146	139	139	129
cc	111	115	112	125	125	125	125	125	125	125	125	103
cht	219	218	218	196	218	196	214	197	197	213	240	182
clip	175	185	185	185	185	185	164	137	137	128	128	183
cm138a	56	56	56	56	56	56	56	56	56	56	56	53
cm150a	45	45	33	33	33	33	33	33	33	33	33	35
cm162a	96	101	114	114	114	114	114	114	114	114	114	88
cm163a	54	54	40	40	40	40	40	40	40	40	40	50
cmb	52	52	52	52	52	52	52	52	52	52	52	49
con1	19	19	19	18	19	18	19	18	18	19	19	19
cordic	107	110	91	93	93	93	91	93	93	90	93	84
count	225	225	216	216	216	216	216	216	216	216	216	230
cps	4142	4358	2865	2824	2790	2824	2856	2862	2862	2828	2828	3069
cu	92	90	91	91	92	93	91	92	92	90	91	91
decod	96	96	96	96	96	96	96	96	96	96	96	91
ex4p	979	883	789	767	773	767	761	728	728	726	730	773
example2	910	926	850	784	797	783	800	796	796	784	793	855
lal	213	212	194	191	191	191	191	191	191	191	191	192
ldd	163	163	160	151	185	172	187	172	172	185	185	142
misex1	76	74	71	79	71	79	76	74	74	71	71	74
misex2	183	185	179	179	181	179	179	179	179	181	180	171
sqrt8	48	48	48	48	48	48	62	44	44	45	45	40
sqrt8ml	45	45	50	46	45	46	45	44	44	45	45	42
squar5	61	60	55	60	55	60	55	61	61	60	60	55
table5	2230	2216	1997	1827	1831	1827	1831	1827	1827	1831	1831	1862
tcon	48	48	48	48	48	48	48	48	48	48	48	45
ttt2	228	324	337	299	323	299	309	299	299	323	323	283
x2	59	57	51	51	51	51	52	51	51	51	51	56
x4	1152	1100	972	885	892	884	1296	1317	1317	1326	1319	1126

Table 3- Experimental Results (APL)

Benchmark	APL (Existing CUDD methods)											Proposed Method
	Random	Random pivot	Sift	Sift converge	Symm sift	Symm sift converge	Group sift	Group sift converge	Window2	Annealing	Genetic	
alu2	24.380859	28.904297	30.337891	30.267578	30.24141	30.267578	30.267578	30.181641	30.181641	30.205078	30.205078	24.505859
alu4	55.039063	55.345703	55.679199	60.150391	60.421387	60.150391	60.421387	59.921387	59.921387	59.650391	64.948242	54.922852
apex4	118.03125	119.0625	119.6875	119.835938	119.6874	119.835938	119.6875	119.835938	119.835938	130.0625	130.648438	114.515625
apex5	335.929566	313.786252	364.862363	360.467083	360.467083	360.467083	356.217358	361.13539	361.13539	396.722882	397.849935	234.719415
apex6	247.590464	247.465464	227.714518	276.464945	276.714945	276.714945	276.674662	277.299662	277.299662	276.799662	276.174662	273.100565
apex7	110.966087	149.954407	95.52388	104.173309	103.923309	104.173309	103.763153	104.001434	104.001434	116.033634	114.85656	103.835907
b1	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5
c8	58.755859	58.755859	57.755859	57.755859	57.755859	57.755859	57.755859	57.755859	57.755859	46.736328	46.736328	48.251953
C17	5.25	5.25	5.25	5.25	5.25	5.25	5.25	5.25	5.25	5.25	5.25	5
cc	36.031205	38.03125	40.375	43.3125	43.3125	43.3125	43.3125	43.3125	43.3125	43.3125	43.3125	40.90625
cht	94.5	95	94.5	94.5	94.5	94.5	94.5	94.5	94.5	94.5	97.75	74.75
clip	31.609375	32.609375	32.609375	32.609375	32.609375	32.609375	32.5625	32.25	32.25	29.703125	29.703125	32.1875
cm42a	18.75	18.75	18.75	18.75	18.75	18.75	18.75	18.75	18.75	18.75	18.75	18.75
cm85a	10.3125	15.65625	17.5	17.5	17.5	17.5	17.5	17.5	17.5	17.5	17.5	14.34375
cm162a	20.632813	21.484375	21.226563	21.226563	21.226563	21.226563	21.226563	21.226563	21.226563	21.226563	21.226563	20.632813
comp	26.79572	26.462158	24.890564	25.593689	25.593689	25.593689	26.437439	26.749939	26.749939	49.999939	49.999939	26.972382
cordic	18.18042	18.203857	19.234131	19.234131	19.234131	19.234131	19.234131	19.234131	19.234131	18.366943	19.866943	13.306885
cps	313.431288	323.399575	323.699575	321.699575	321.699575	321.699575	322.512075	322.762075	322.762075	322.762075	322.762075	251.119352
cu	25.357422	25.351563	25.351563	25.375	25.375	25.375	25.375	25.375	25.375	25.835938	27.085938	25.398438
ex4p	70.445648	91.034271	94.523712	95.430817	95.430817	95.430817	95.83316	101.973785	101.973785	104.10733	104.10733	50.881683
example2	183.663208	185.110962	182.619995	174.590698	172.715698	174.090698	174.215698	172.215698	172.215698	174.590698	181.090698	160.762573
frg1	10.266755	10.250275	11.156635	13.09553	12.773172	13.148172	12.855691	13.105691	13.105691	12.670532	12.353897	12.796021
i1	23.867188	24.92188	24.492188	24.492188	24.492188	24.492188	24.492188	24.492188	24.492188	24.492188	24.492188	24.054688
ldd	48.59375	48.59375	48.59375	50.257813	56.921875	56.921875	56.921875	56.921875	56.921875	56.921875	56.921875	46.515625
majority	2.5625	3.8125	3.8125	3.8125	3.8125	3.8125	3.8125	3.8125	3.8125	3.8125	3.8125	2.5625
misex1	23.84375	23.96875	23.96875	23.96875	23.96875	23.96875	23.96875	23.96875	23.96875	23.96875	23.96875	23.15625
misex2	40.248047	43.966797	42.98047	42.998047	42.998047	42.998047	42.998047	42.998047	42.998047	42.998047	42.873047	40.123047
parity	16	16	16	16	16	16	16	16	16	16	16	16
pcle	27.006836	27.006836	27.006836	27.006836	27.006836	27.006836	27.006836	27.006836	27.006836	27.006836	27.006836	27.006836
sao2	22.876953	22.806641	22.681641	22.681641	22.681641	22.681641	22.681641	22.681641	22.681641	22.681641	22.681641	14.638672
sct	49.516311	46.151123	42.77832	42.27832	43.15332	42.27832	43.15332	42.27832	42.27832	43.15332	43.15332	42.313965
sqrt8	13.046875	13.046875	13.03125	13.046875	13.046875	13.046875	14.203125	15.984375	15.984375	16.125	16.125	13.03125
sqrt8ml	16.125	16.125	15.84375	15.640625	15.640625	15.640625	15.640625	14.890625	14.890625	14.890625	14.890625	16.125
squar5	22.3125	23.3125	23.6875	23.3125	23.6875	23.3125	23.6875	23.3125	23.3125	23.3125	23.3125	21.5625
table5	85.639114	85.879593	87.383499	86.367874	86.367874	86.367874	86.367874	86.367874	86.367874	86.367874	86.367874	81.002335
ttt2	55.531494	70.438838	73.375244	70.679932	71.672119	70.679932	70.679932	70.679932	70.679932	71.672119	71.672119	60.69165
vda	183.389893	183.249756	183.979249	184.072998	184.135498	184.072998	184.072998	184.072998	184.072998	177.800537	176.300537	175.692627
vg2	44.245258	46.409443	44.58205	44.58205	44.58205	44.58205	44.58205	44.58205	44.58205	44.58205	36.913597	44.371967
x1	131.186701	130.931841	130.872787	131.286029	131.286029	131.286029	128.076099	128.076099	128.076099	142.651331	142.276331	114.274374
x4	264.226746	261.613708	263.698547	241.59137	241.59137	241.59137	243.17926	239.611877	239.611877	240.111877	240.611877	179.789001
z4ml	19	19	19	19	19	19	17.5	16.375	16.375	16.375	16.375	19

References

[1] R. E. Bryant, Graph–Based Algorithm for Boolean Function Manipulation, *IEEE Trans. Computers, Vol. 35*, pp. 677-691, 1986.

[2] S. B. Akers, Binary Decision Diagram, *IEEE Trans. Computers, Vol. 27*, pp. 509-516, 1978.

[3] K. Priyank, VLSI Logic Test, Validation and Verification, Properties & Applications of Binary Decision Diagrams, *Lecture Notes*, Department of Electrical and Computer Engineering University of Utah, Salt Lake City, UT 84112, 1997.

[4] Ingo W.: "Complexity of Boolean function", *John Wiley & Sons Ltd, and B. G. Teubner, Stuttgart*, 1987.

[5] P. W. C. Prasad and A. K. Singh, An Efficient Method for Minimization of Binary Decision Diagrams, *Proceedings of 3rd Int. Conf. on Advances in Strategic Technologies*, pp. 683-688, 2003.

[6] R. Rudell, "Dynamic Variable Ordering for Ordered Binary Decision Diagrams," *Proceedings of the International Conference on Computer Aided Design (ICCAD)*, pp. 42-47, 1993.

[7] R. Ebendt, Reducing the number of variable movements in exact BDD minimization, *Proceedings of 2003 Int. Symp. on Circuits and Systems*, pp. 605-608, 2003.

[8] Fadi A. Aloul, Igor L. Markov, Karem A. Sakallah, "Improving the Efficiency of Circuit-to-BDD Conversion by Gate and Input Ordering" *20th International Conference on Computer Design (ICCD 2002)*, pp. 64-69, 2002.

[9] Justin E. Harlow and Franc Brglez, "Design of Experiments and evaluation of BDD ordering Heuristics", *Inter. Journal on Software Tools for Technology Transfer*, Vol. 3 , No.2 , pp. 193-206, 2001.

[10] F. Aloul, I. Markov, K. Sakallah, "MINCE: A Static Global Variable-Ordering Heuristic for SAT Search and BDD Manipulation", *Journal of Universal Computer Science* (JUCS), Vol 10, No 4, pp 1-6, 2004.

[11] M. Fujita, H. Fujisawa, and N. Kawato, "Evaluation and Improvements of Boolean Comparison Method Based on Binary Decision Diagrams," *Proceedings of the InternationalConference on Computer Aided Design (ICCAD)*, pp. 2-5, 1988.

[12] S. Malik, A. Wang, R. Brayton, and A. Sangiovanni-Vincentelli, "Logic Verification Using Binary Decision Diagrams in a Logic Synthesis Environment," *Proceedings of the International Conference on Computer Aided Design (ICCAD)*, pp. 6-9, 1988.

[13] F. Somenzi, "Efficient Manipulation of Decision Diagrams," *in International Journal on Software Tools for Technology Transfer (STTT)*, 3(2), pp. 171-181, 2001.

[14] S. Nagayama, A. Mishchenko, T. Sasao, and J. T. Butler, "Minimization of average path length in BDDs by variable reordering," *International Workshop on Logic and Synthesis*, pp. 207-213, Laguna Beach, California, U.S.A., May 28-30, 2003.

[15] R. Ebendt, S. Hoehne, W. Guenther, and R. Drechsler, "Minimization of the expected path length in BDDs based on local changes," *Proceedings of Asia and South Pacific Design Automation Conference (ASP-DAC'2004)*, pp. 866-871, Yokohama, Japan, Jan. 2004.

[16] Y. Liu, K. H. Wang, T. T. Hwang, C. L. Liu, "Binary decision Diagrams with minimum expected path length," *Proceedings of DATE 01*, pp. 708–712, Mar. 13-16, 2001.

[17] Görschwin Fey, Junhao Shi and Rolf Drechsler, "BDD Circuit Optimization for Path Delay Fault-Testability", *Proceedings of EUROMICRO Symposium on Digital System Design*, pp. 168-172, 2004.

[18] S. Nagayama and T. Sasao, "On the minimization of longest path length for decision diagrams," *International Workshop on Logic and Synthesis (IWLS)*, June 2-4, Temecula, California, U.S.A., pp. 28-35, 2004.

[19] F. Balarin, M. Chiodo, P. Giusto, H. Hsieh, A. Jurecska, L. Lavagno, A. Sangiovanni-Vincentelli, E. M. Sentovich, and K. Suzuki, "Synthesis of software programs for embedded control applications," *IEEE Trans. CAD*, Vol. 18, No. 6, pp. 834-849, June 1999.

[20] M. Lindgren, H. Hansson, and H. Thane, "Using measurements to derive the worst-case execution time," *7th International Conference on Real-Time Systems and Applications (RTCSA'00)*, pp. 15-22, 2000.

[21] R. S. Shelar and S. S. Sapatnekar, Recursive Bipartitioning of BDD's for Performance Driven Synthesis of Pass Transistor Logic, *Proceedings of IEEE/ACM ICCAD*, pp. 449 – 452, 2001

[22] V. Bertacco, S. Minato, P. Verplaetse, L. Benini, and G. De Micheli: "Decision Diagrams and Pass Transistor Logic Synthesis", *Stanford University CSL Technical Report*, No. CSL-TR-97-748, Dec. 1997.

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

[24] R. Drechsler and D. Sieling, Binary Decision Diagrams in Theory and Practice, *Springer-Verlag Trans.*, pp. 112-136, 2001.

[25] P.W.C. Prasad, M. Raseen, A. Assi and S.M.N.A. Senanayake, "BDD Path Length Minimization based on Initial Variable Ordering", Journal of Computer

Science, Vol. 1, Issue 4, pp.521-529, Science Publications, USA, 2005

[26] P. W. C. Prasad, M. Raseen and S. Sasikumaran, "Delay Minimization in Pass Transistor Logic use of Binary Decision Diagram", *2nd International Conference on Information Technology (ICIT 2005)*, pp 66-70, Jordan, May 2005.

[27] F. Somenzi, CUDD: Colorado University Decision Diagram Package. *ftp://vlsi.colorado.edu/ pub/.*, 2003.

[28] S. yang. Logic synthesis and optimization benchmarks user guide version 3.0. *Technical report*, Microelectronic Centre of North Caroline, Research Triangle Park, NC, January 1991.

[29] M. Hansen, H. Yalcin, and J. P. Hayes, "Unveiling the ISCAS-85 Benchmarks: A Case Study in Reverse Engineering," *IEEE International Journal on Design and Test*, vol. 16, no. 3, pp. 72-80, July-Sept. 1999.

[30] P.W. C. Prasad, A. Assi, A. Harb and V.C. Prasad, "Binary Decision Diagrams: An Improved Variable Ordering using Graph Representation of Boolean Functions", *International Journal of Computer Science*, vol. 1, no. 1, pp 1-7, 2006.

Mohamed Raseen is currently a research scholar in Coimbatore Institute of Engineering and information technology. He has obtained his MS degree in university of Houston-Clearlake and his bachelor's degree from Government college of Technology, Coimbatore India. He has published 17 papers in international conferences and journals. His research area is binary decision diagrams and algorithms.

Dr.K.Thanushkodi has got 30 yrs of teaching experience in Government Engineering Colleges. He has published 45 papers in international journals and conferences. He has guided 1 PhD and 1 MS (by research) students. He is currently guiding 15 research scholars in the area of power system engineering, power electronics and computer networks. He has been principal in-charge and Dean in Government College of engineering Bargur. Heserved as senate member in periyar university salem. He served as member of the research board, Anna University at channai. He Served as Member, Academic Council, Anna University, Chennai. He is Serving as Member Board of Studies in Electrical Engineering, Anna University, Chennai. He is serving as Member, Board of Studies in Electrical and Electronics & Electronics and Communication Engineering, Amritha Viswa Vidya Peetham, Deemed University, Coimbatore. He is sServing as Governing Council Member SACS MAVMM Engineering College, Madurai. He served as Professor and Head of E&I, EEE, CSE & IT Departments at Government College of Technology, Coimbatore.