International Journal of Soft Computing 3 (6): 451-460, 2008
ISSN: 1816-9503

ROBDD's Parameter Estimation for Simplified Boolean Functions

Mohamed Raseen and K. Thanushkodi
Coimbatore Institute of Engineering and Information Technology,
Vellimalaipattinam, Narasipuram (Post), Coimbatore, 641109, India

Abstract: For decades, the storage of Boolean functions using Binary Decision Diagrams (BDD) has been popular. Efficiency of representing a Boolean function by BDD depends upon the size of the BDD (Number of nodes). Evaluating time on Boolean function using BDD depends upon the path lengths of the BDD. The efficiency of manipulating Boolean function depends upon the BDD parameters such as Longest Path Length (LPL), Average Path Length (APL) and Shortest Path Length (SPL). Lesser the value of the parameters, lesser will be the computational time. This study describes a new mathematical model for the estimation of the complexity of Binary Decision Diagrams (BDDs) for simplified Boolean functions. This study provides a mathematical model for estimation of Number of Nodes, APL, LPL and SPL for a given set of values such as number of nodes/number of product terms. The mathematical model precisely matches the experimental values obtained from CUDD (Colorado university decision diagram package). This mathematical model works for any type of variable ordering method. The model enables the system to find the number of BDD nodes and path lengths (APL, LPL, SPL) without building the BDD. Hence, a great reduction in time complexity for VLSI and CAD designs can be achieved. It can also offer very useful clues to tackle BDD optimization problems in the design of digital circuits.

Key words: Binary decision diagrams, design complexity, mathematical models, boolean functions representations, simplified boolean functions

INTRODUCTION

For the last 2 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 (1978) and Bryant (1986). 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 (Priyank, 1997; Ingo, 1987), 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 (Prasad and Singh, 2003; Rudell, 1993; Ebendt, 2003). Finding a better variable order is often worth spending considerable computational effort (Aloul *et al*., 2000). 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 (Harlow and Brglez, 2001). 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 (Aloul *et al*., 2004). Accordingly, much attention has been devoted to techniques for finding a good variable ordering. All these variable ordering techniques fall into 2 categories: Static Variable Ordering (SVO) algorithms (Fujita *et al*., 1988; Malik *et al*., 1988) and Dynamic Variable Ordering (DVO) algorithms (Rudell, 1993; Somenzi, 2001).

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

Corresponding Author: Mohamed Raseen, Coimbatore Institute of Engineering and Information Technology, Vellimalaipattinam, Narasipuram (Post), Coimbatore, India 641109

implementation. In general, the minimum path length in Decision Diagrams (DD) is important in databases, pattern recognition, logic simulation and software synthesis (Nagayama *et al*., 2003). The minimization of Average Path Length (APL) proposed in (Nagayama *et al*., 2003; Ebendt *et al*., 2004; Liu *et al*., 2001) 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 (Fey *et al*., 2004). The APL minimization is very much effective in Real time operating system applications (Nagayama and Sasao, 2004a; Balarin *et al*., 1999; Lindgren *et al*., 2000). The minimization of Longest Path Length (LPL) of BDD can reduce the longest evaluation time which is more important for Pass Transistor Logic (PTL) (Nagayama and Sasao, 2004b; Shelar and Sapatnekar, 2001; Bertacco *et al*., 1997). 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 (Prasad *et al*., 2005). The path length can be reduced by inserting buffers, but this increases the circuit area. The minimization of longest evaluation time will improve the performance of the circuit (Nagayama and Sasao, 2004b; Bertacco *et al*., 1997). An algorithm for finding the optimal variable ordering for the minimization of BDD path length was investigated by the authors (Prasad *et al*., 2005). The resulted initial variable order will produce the BDD with minimum possible APL and consequently reducing the number of nodes to an affordable size.

The size of the BDD is very vital Since, it is directly proportional to the memory needed for storing the BDD. The sizes of the BDDs for un-simplified Boolean function is given by Raseen *et al*. (2006, 2005a). Figure 1 shows the variation of BDD sizes for un-simplified Boolean functions with 9 variables.

From Raseen *et al*. (2006, 2005a) it can be inferred that the size of the BDD for un-simplified Boolean function is an exponential decay. The sizes of BDDs for XOR/XNOR functions are given in (Raseen *et al*., 2005b). The variation of the sizes of BDD for XOR/XNOR functions (Raseen *et al*., 2005b) is given in Fig. 2.

From Raseen *et al*. (2005b) the sizes of the BDD for XOR/XNOR functions vary in a semi-circular pattern. This study describes the sizes of BDDs for simplified Boolean functions which are not XOR/XNOR functions.

PRELIMINARIES

Basic definitions for BDDs are given in Bryant (1686), Akers (1978), Drechsler and Becker (1998) and Drechsler and Sieling (2001). In the following we review some of these definitions.

Definition 1: A BDD is a Directed Acyclic Graph (DAG). The graph has 2 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 2 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.

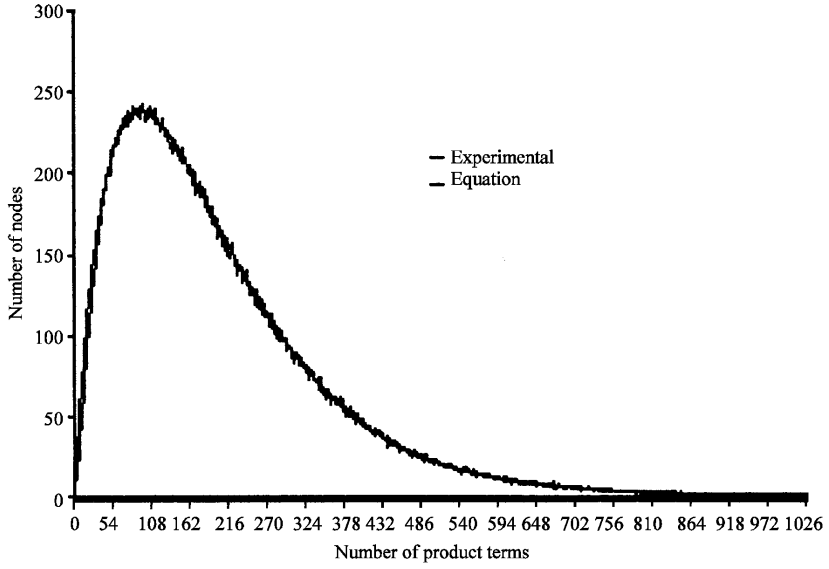

Fig. 1: BDD Size variation for un-simplified Boolean Functions with 9 variables

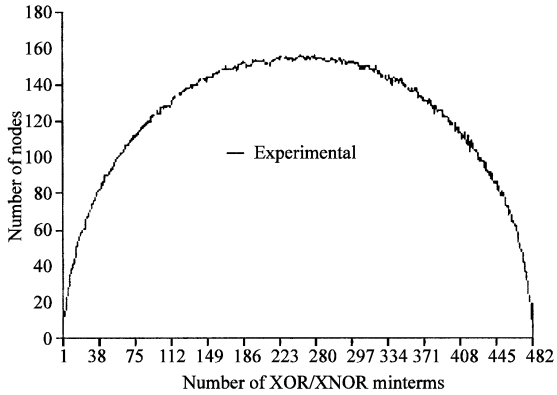

Fig. 2: BDD Size variation for XOR/XNOR minterms

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 2 reduction rules: deletion rule and merging rule. These Reduction rules remove redundancies from the OBDD.

Variable ordering: The size of a BDD is largely affected (and varies from linear to exponential) by the choice of the variable ordering (Prasad *et al*., 2006). Figure 3 illustrates the effect of the variable ordering on the size of BDDs (Bryant, 1986) for the following Boolean function (1):

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

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 (Nagayama *et al*., 2003; Nagayama and Sasao, 2004), which give the following Eq. (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}),

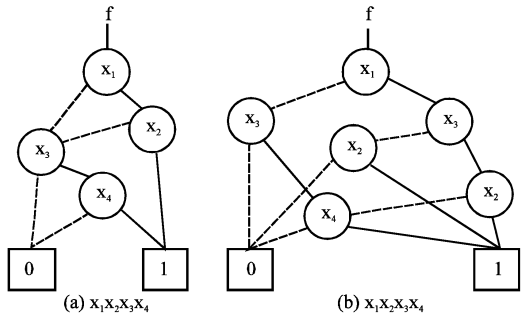

Fig. 3: Effect of variable ordering on the size of BDDs

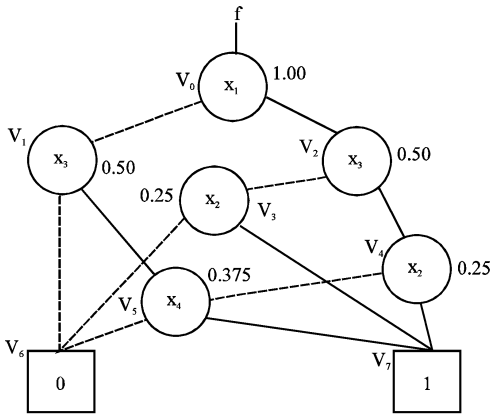

Fig. 4: Node traversing probability in a BDD

where, e_{i0} (or e_{i1}) denotes the 0-edge (or the 1-edge) directed from away node V (Nagayama *et al*., 2003). 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 Eq. (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 Fig. 4, 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_{00}) = 0.50 and P (V_2) = P (e_{10}) = 0.50. In a similar manner we calculate:

- P (V_3) = P (e_{20}) = 0.25.
- P (V_4) = P (e_{21}) = 0.25.
- P (V_5) = P (e_{40}) = P (e_{11}) = 0.125+0.25 = 0.375.

So,

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

$$LPL = \text{Longest Path} = x_1 \rightarrow x_3 \rightarrow x_2 \rightarrow x_4 = 4$$

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 (Nagayama and Sasao, 2004).

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 2 pointers, the memory size needed to represent a BDD is:

Mem (BDD = (2+1) × nodes (BDD) (Ingo, 1987)

PROPOSED MATHEMATICAL MODEL

The complexity of the BDD depends on the number of nodes in the BDD and the corresponding path lengths. The number of nodes and path lengths for a BDD depends upon Boolean function represented by the BDD. An experiment was conducted to find the actual relation between the number of nodes, APL, LPL and SPL in the BDD and number of product terms (in the simplified function). The CUDD (Colorado University Decision Diagram) (Somenzi, 2003), was used to conduct the experiment. The experiment included the following steps:

Step 1: The number of variables is read from the user.

Step 2: The variable ordering is fixed (for example genetic algorithm).

Step 3: The number of product terms (n) is set to 1.

Step 4: A random Boolean function with n non repeating product term is generated.

Step 5: Simplify the generated function using Quine Mc-Cluskey method.

Step 6: Run the CUDD to build the BDD for the simplified function.

Step 7: Reorder the BDD using the method selected in step 2.

Step 8: Record the number of nodes, APL, LPL and SPL in BDD.

Step 9: Repeat steps 4-7 for n=1 to n= (number of inputs)^2

Step 10: Record the number of nodes, APL, LPL and SPL for all variables and all product terms.

Step 11: Using the collected data the mathematical models are obtained.

Mathematical model for number of nodes: Figure 5 illustrates the relation between the number of product terms and number of BDD nodes for 8 variables using genetic algorithm reduction technique.

For 8 variables the number of nodes increases as the number of product terms increase. The number of nodes increases up to a certain limit beyond which the numbers of nodes reduce to 0.

This variation is due to the fact that beyond a certain limit the input Boolean function simplifies and reduces to 0 nodes. The Fig. 6 shows the number of nodes for 9 variables. From Fig. 6 it can be inferred that the variation of number of nodes follows a specific pattern of rising and falling. This pattern can be modeled using an

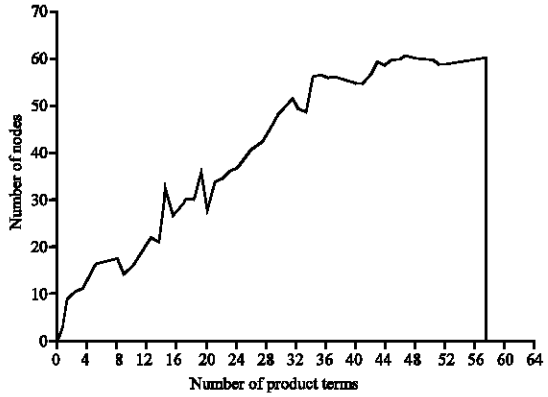

Fig. 5: Number of nodes for 8 variables

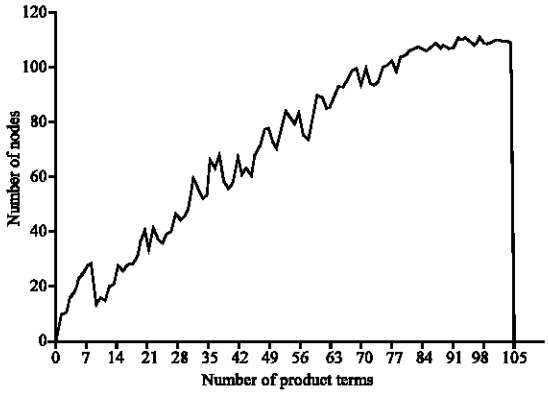

Fig. 6: Number of nodes for 9 variables

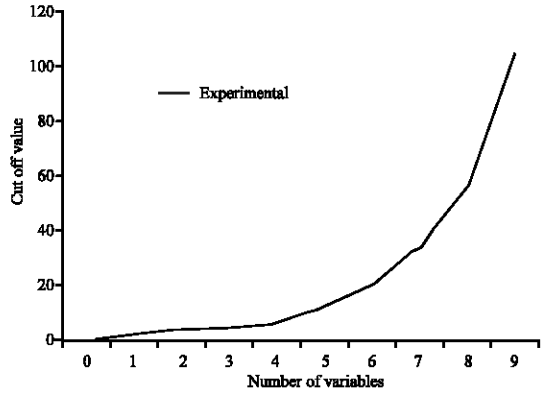

Fig. 7: Cut off values (experimental)

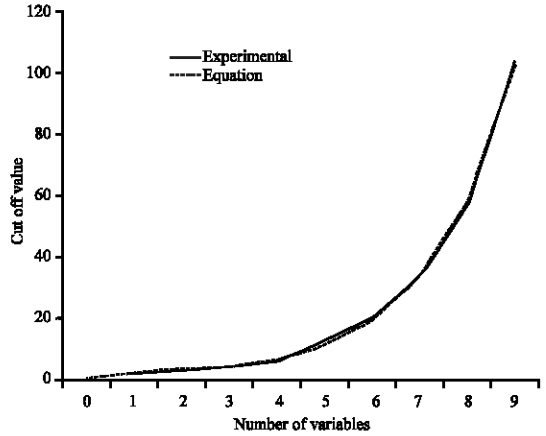

Fig. 8: Cut off values (experimental and equation)

equation. From the experimental results it can be inferred that the number of nodes starts from 0 and rises to a particular limit and falls to zero beyond the limit. The relation between the number of variables and the number of product terms (for which the nodes reduces to 0) is shown in Fig. 7.

The curve in Fig. 7 can be represented using the Eq. (4) shown:

$$COP = 0.6378*e^{(0.5591*NV)} \quad (4)$$

where,

COP : Cutoff point.
NV : Number of variables.

Figure 8 shows the experimental value of the cutoff points and the estimation using the equation. From Fig. 8 it can be inferred that the equation matches the experimental values precisely.

The variation of number of nodes in Fig. 6 (for 9 variables) can be modeled using the following Eq. (2):

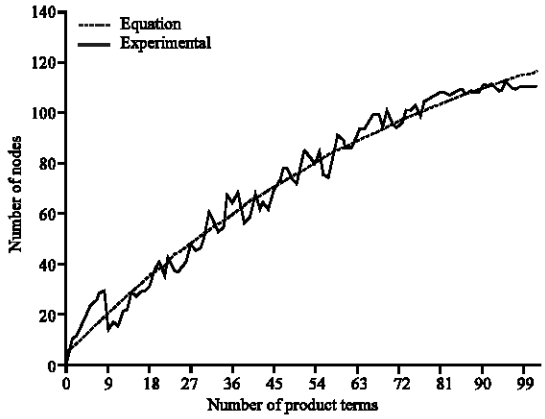

Fig. 9: Number of nodes (experimental and equation)

Table 1: Values pf parameters a, b, c and d

Number of variables	a	b	c	d
3	3.867	0.02509	-3.867	-27.94
4	4.716	0.05466	-4.716	-40.63
5	8.79	0.02206	-8.38	-1.203
6	-0.3187	0.2131	7.875	0.0771
7	-65.710	0.09625	75.02	0.09281
8	51.530	0.05834	-38.62	0.06276
9	323.500	0.002399	-319.00	0.008221

$$\begin{aligned} NN &= 0 \text{ for } NPT <= 0 \\ &= 0 \text{ for } NPT > COP \\ &= 323.5*e^{(-0.002399*NPT)} - 319*e^{(-0.008221*NPT)} \end{aligned} \quad (5)$$

where, COP is Eq. (4). The Fig. 9 shows the number of nodes from experiment and from Eq. (5). From Fig. 9 it can be inferred that the equation is a good representation of the system.

The Eq. (5) can be generalized for any number of variables as follows:

$$\begin{aligned} NN &= 0 \text{ for } NPT <= 0 \\ &= 0 \text{ for } NPT > COP \\ &= a*e^{b} + c*e^{d} \end{aligned} \quad (6)$$

The values of the parameters a, b, c and d from experiment are shown in Table 1.

From the Table 1, using curve fitting techniques the equations of the variables a, b, c and d are determined as:

$$a = 0.0001624*e^{1.641*NV} - 0.6142*e^{0.5608*NV}$$
$$b = 0.0213*e^{0.2728*NV} - 0.000002493*e^{1.029*NV}$$
$$c = 0.0000002389*e^{2.098*NV} + 1.267*e^{0.423*NV}$$
$$d = 1.211*e^{-3.095*NV} + 1.24*e^{-3.103*NV}$$

where, NV is the number of variables. Thus when the number of variables (NV) and the number of simplified

SOP terms (NPT) is given the number of nodes in the BDD can be found using the equations. This experiment was conducted using Genetic algorithm as the variable ordering method. In order to find the nodes for other methods the Eq. (6) has to be modified with an amplification factor derived from (Raseen *et al*., 2006). The new equation supporting all variable ordering method is given by Eq. (7):

$$\begin{aligned} NN &= 0 \text{ for } NPT <= 0 \\ &= 0 \text{ for } NPT > COP \\ &= A.F.\ (a*e^{(b*NPT)} + c*e^{(d*NPT)}) \end{aligned} \quad (7)$$

where, A.F. is the amplification factor given by a table in Raseen *et al*. (2006). Hence, Eq. (7) can be used to find the number of nodes for all the existing variable reordering methods.

Mathematical Model for APL: Figure 10 illustrates the relation between the number of product terms and APL of BDD for 8 variables using genetic algorithm reduction technique.

For 8 variables the number of nodes increases as the number of product terms increase. The number of nodes increases up to a certain limit beyond which the APL reduces to 0.

This variation is due to the fact that beyond a certain limit the input Boolean function simplifies and reduces to 0 nodes. The Fig. 11 shows the APL for 9 variables. From Fig. 11 it can be inferred that the variation of number of nodes follows a specific pattern of rising and falling. This pattern can be modeled using an equation.

From the experimental results it can be inferred that the number of nodes starts from 0 and rises to a particular limit and falls to zero beyond the limit. The relation between the number of variables and the number of product terms (for which the nodes reduces to zero) is shown in Fig. 12.

The curve in Fig. 12 can be represented using the Eq. (8):

$$COP = 0.5691*e^{(0.5771*NV)} \quad (8)$$

where, COP is cutoff point and NV is the number of variables.

Figure 13 shows the experimental value of the cutoff points and the estimation using the equation. From Fig. 13 it can be inferred that the equation matches the experimental values precisely.

The variation of number of nodes in Fig. 11 (for 9 variables) can be modeled using the following Eq. (6):

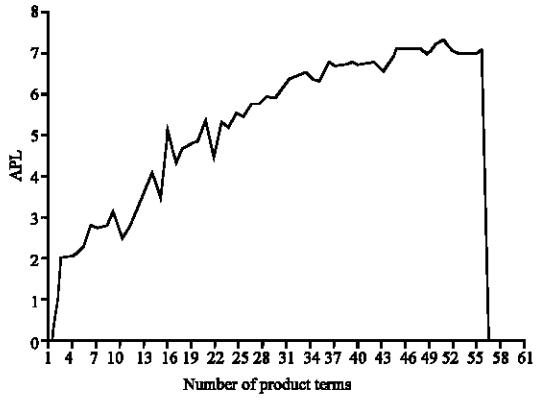

Fig. 10: APL for 8 variables

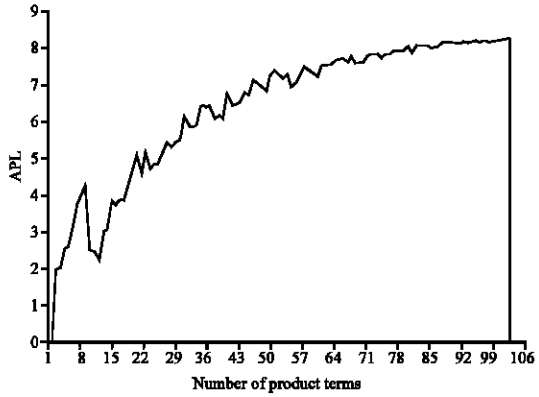

Fig. 11: APL for 9 variables

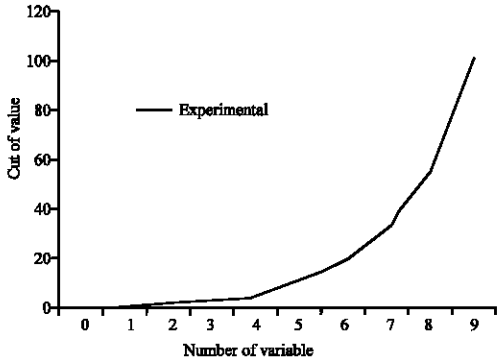

Fig. 12: Cut off values (experimental)

$$\begin{aligned} APL &= 0 \text{ for } NPT <= 0 \\ &= 0 \text{ for } NPT > COP \\ &= 8.598*e^{(-0.00000491*NPT)} - 7.081*e^{(-0.02893*NPT)} \end{aligned} \quad (9)$$

where, COP is Eq. (8). The Fig. 14 shows the APL from experiment and from Eq. (9). From Fig. 14 it can be inferred that the equation is a good representation of the system.

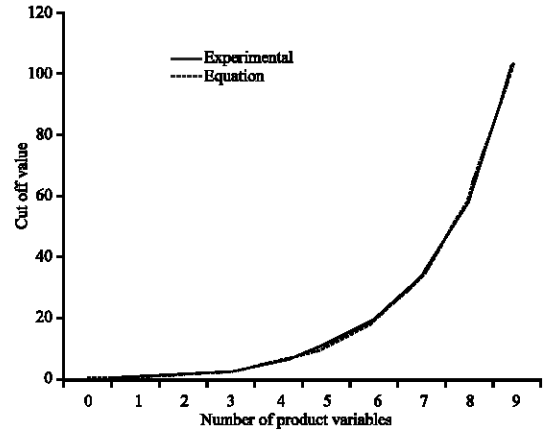

Fig. 13: Cut off values (experimental and equation)

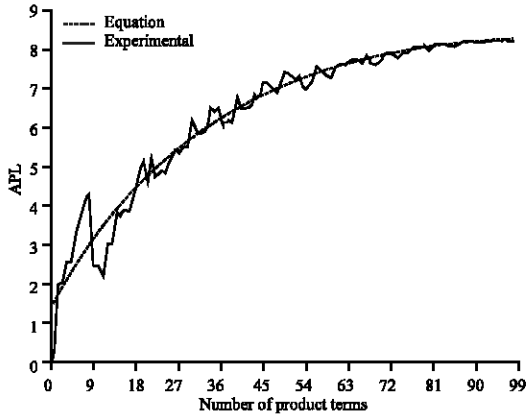

Fig. 14: APL (experimental and equation)

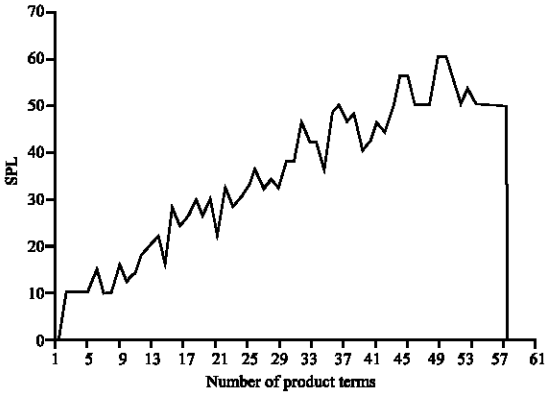

Fig. 15: SPL for 8 variables

The Eq. (9) can be generalized for any number of variables as follows:

$$\begin{aligned} APL &= 0 \text{ for } NPT <= 0 \\ &= 0 \text{ for } NPT > COP \\ &= a*e^{b} + c*e^{d} \end{aligned} \quad (10)$$

Table 2: Values of parameters a, b, c and d for APL

Number of variables	a	b	c	d
3	1.579	0.09313	-1.579	-28.45
4	1.613	0.1238	-1.613	-38.27
5	3.089	0.02641	-3.086	-0.9286
6	-0.1995	0.1783	2.289	-0.07951
7	17.28	0.07927	-15.1	0.08261
8	17.48	-0.00805	-16.43	-0.02548
9	8.589	-4.91e-06	-7.081	0.02893

The values of the parameters a, b, c and d from experiment are shown in Table 2.

From the Table 2, using curve fitting techniques the equations of the variables a, b, c and d are determined as

$$a = -1.215e\text{-}7*e^{2.175*NV} + 0.09628*e^{0.6876*NV}$$
$$b = 0.109*e^{-0.01264*NV} - 0.0001234*e^{0.7595*NV}$$
$$c = 1.367e\text{-}9*e^{2.663*NV} - 0.05769*e^{0.7325*NV}$$
$$d = -1.386e9*e^{-3.121*NV} + 1.415e9*e^{-3.128*NV}$$

where, NV is the number of variables. Thus when the number of variables (NV) and the number of simplified SOP terms (NPT) is given the APL of the BDD can be found using the equations. This experiment was conducted using Genetic algorithm as the variable ordering method. In order to find the APL for other methods the Eq. (10) has to be modified with an amplification factor derived from Raseen *et al.* (2006). The new equation supporting all variable ordering method is given by Eq. (11):

$$\begin{aligned} APL &= 0 \text{ for } NPT <= 0 \\ &= 0 \text{ for } NPT > COP \\ &= A.F.(a*e^{(b*NPT)} + c*e^{(d*NPT)}) \end{aligned} \quad (11)$$

where, A.F. is the amplification factor given by a table in Raseen *et al.* (2006). Hence, Eq. (11) can be used to find the number of nodes for all the existing variable reordering methods.

Mathematical Model for SPL: Figure 15 illustrates the relation between the number of product terms and SPL of BDD for 8 variables using genetic algorithm reduction technique.

For 8 variables the number of nodes increases as the number of product terms increase. The number of nodes increases up to a certain limit beyond which the APL reduces to 0.

This variation is due to the fact that beyond a certain limit the input Boolean function simplifies and reduces to 0 nodes. The Fig. 16 shows the SPL for 9 variables. From Fig. 16 it can be inferred that the variation of SPL follows a specific pattern of rising and falling. This pattern can be modeled using an equation.

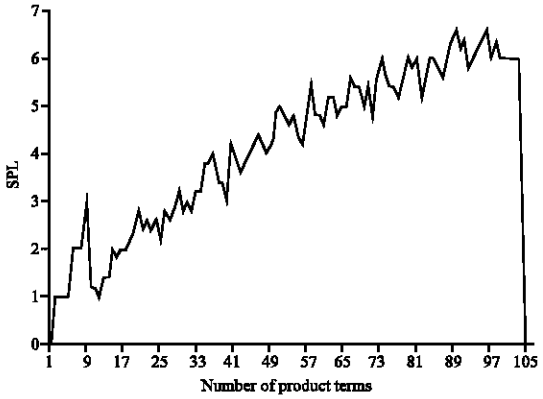

Fig. 16: SPL for 9 variables

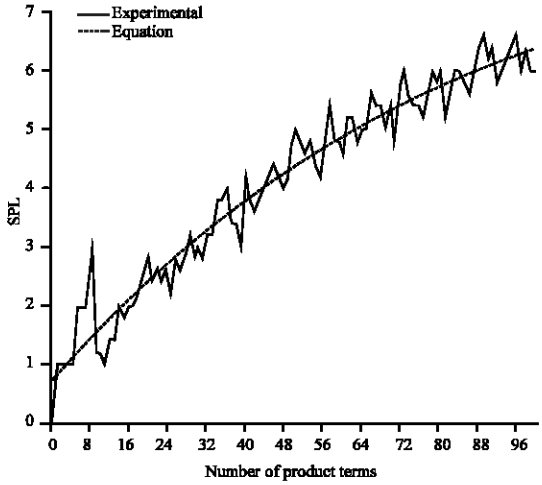

Fig. 17: SPL (experimental and equation)

The cutoff points for LPL (beyond which the LPL goes to 0) is same as the cutoff points for APL and is given by Eq. (8). The variation of number of nodes in Fig. 16 (for 9 variables) can be modeled using the following Eq. (12):

$$\begin{aligned} SPL &= 0 \text{ for NPT} <= 0 \\ &= 0 \text{ for NPT} > COP \\ &= 19.25*e^{(-0.002422*NPT)} - 18.5*e^{(-0.00741*NPT)} \end{aligned} \quad (12)$$

where, COP is Eq. (4). The Fig. 17 shows the SPL from experiment and from Eq. (12). From Fig. 17 it can be inferred that the equation is a good representation of the system. The Eq. (9) can be generalized for any number of variables as follows:

$$\begin{aligned} SPL &= 0 \text{ for NPT} <= 0 \\ &= 0 \text{ for NPT} > COP \\ &= a*e^{b} + c*e^{d} \end{aligned} \quad (13)$$

Table 3: Values of parameters a, b, c and d for SPL

Number of variables	a	b	c	d
3	1.911	-0.02311	-1.917	-0.8247
4	0.624	0.2471	-0.624	-31.96
5	6.463	0.2142	-5.45	0.2258
6	-0.01412	0.2704	1.191	0.08485
7	-0.00568	0.2157	-1.062	0.06953
8	13.22	-0.00204	-12.8	-0.01299
9	19.25	-0.00242	-18.5	-0.00741

The values of the parameters a, b, c and d from experiment are shown in Table 3.

From the Table 3, using curve fitting techniques the equations of the variables a, b, c and d are determined as:

$$\begin{aligned} a &= 0.01681*e^{0.7863*NV} + 7.113*e^{-0.4266*NV} \\ b &= 1.171*e^{-0.3141*NV} - 294.5*e^{-2.138*NV} \\ c &= -0.02052*e^{0.7607*NV} - 13.16*e^{-0.6588*NV} \\ d &= -1.201e8*e^{-3.126*NV} + 1.499e8*e^{-3.2*NV} \end{aligned}$$

where, NV is the number of variables. The new equation supporting all variable ordering method is given by Eq. (14):

$$\begin{aligned} SPL &= 0 \text{ for NPT} <= 0 \\ &= 0 \text{ for NPT} > COP \\ &= A.F.\ (a*e^{(b*NPT)} + c*e^{(d*NPT)}) \end{aligned} \quad (14)$$

where, A.F. is the amplification factor given by a table in Raseen *et al.* (2006). Hence, Eq. (14) can be used to find the number of nodes for all the existing variable reordering methods.

Mathematical Model for LPL: The variation of Longest Path Length is very simple and is equal to the number of variables. Experimental results indicate that the LPL is equal to the number of variables for any number of product terms. Thus

LPL = Number of variables.

CONCLUSION

In this research, we address the problem of finding the number of nodes, APL, LPL and SPL in a BDD for simplified Boolean functions without actually building the BDD. We introduce a mathematical model for the estimation of the BDD sizes. We also introduce a mathematical model for the estimation of path lengths. We have shown that the mathematical model accurately represents the experimental results. The model will help the VLSI/CAD implementations to exactly estimate the ROBDD nodes and path lengths without actually building

the ROBDD. The introduced method will decrease the time complexity for systems that use BDD for representing Boolean functions. The method also provides the user with an estimate of the maximum complexity (cut off point) for a given number of variables. By finding the maximum complexity the user can determine the limit of the memory needed for BDD manipulations. The experimental and the equation graphs match very well thereby demonstrating the efficiency of the proposed method. Future works and developments will be the application of the proposed method for the universal Benchmark circuits (Yang, 1991; Hansen *et al*., 1999) and to validate the proposed method.

REFERENCES

Akers, S.B., 1978. Binary decision diagram. IEEE. Trans. Comput., 27: 509-516.

Aloul, F., I. Markov and K. Sakallah, 2004. MINCE: A static global variable-ordering heuristic for SAT Search and BDD Manipulation. J. Univ. Comput. Sci. (JUCS), 10 (4): 1-6.

Aloul, F.A., I.L. Markov and K.A. Sakallah, 2000. Improving the Efficiency of Circuit-to-BDD Conversion by Gate and Input Ordering 20th. Int. Conf. Comput. Design (ICCD 2002), pp: 64-69.

Balarin, F., M. Chiodo, P. Giusto, H. Hsieh, A. Jurecska, L. Lavagno, A.S. Vincentelli, E.M. Sentovich and K. Suzuki, 1999. Synthesis of software programs for embedded control applications. IEEE. Trans. CAD, 18 (6): 834-849.

Bertacco, V., S. Minato, P. Verplaetse, L. Benini and G. De Micheli, 1997. Decision Diagrams and Pass Transistor Logic Synthesis. July 18, 2008 Stanford University CSL Technical Report, No. CSL-TR-97-748.

Bryant, R.E., 1986. Graph-based algorithm for boolean function manipulation. IEEE. Trans. Comput., 35: 677-691.

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

Drechsler, R. and B. Becker, 1998. Binary decision diagrams theory and implementation. Kluwer Academic Publishers.

Ebendt, R., 2003. Reducing the number of variable movements in exact BDD minimization. In: Proc. 2003 Int. Symp. Circuits Syst., pp: 605-608.

Ebendt, R., S. Hoehne, W. Guenther and R. Drechsler, 2004. Minimization of the expected path length in BDDs based on local changes. In: Proc. Asia South Pacific Design Automation Conference (ASP-DAC), Yokohama, Japan, pp: 866-871.

Fey, G., J. Shi and R. Drechsler, 2004. BDD circuit optimization for path delay fault-testability. In: Proceeding of the EUROMICRO Symposium Digital System Design, pp: 168-172.

Fujita, M., H. Fujisawa and N. Kawato, 1988. Evaluation and improvements of boolean comparison method based on binary decision diagrams. IN: Proc. Int. Conf. Comput. Aided Design (ICCAD), pp: 2-5.

Hansen, M., H. Yalcin and J.P. Hayes, 1999. Unveiling the ISCAS-85 Benchmarks: A case study in reverse engineering. IEEE. Int. J. Design Test, 16 (3): 72-80.

Harlow, J.E. and F. Brglez, 2001. Design of Experiments and evaluation of BDD ordering Heuristics. Int. J. Software Tools Technol. Trans., 3 (2): 193-206.

Ingo, W., 1987. Complexity of Boolean Function. John Wiley and Sons Ltd and Teubner, B.G. Stuttgart.

Lindgren, M., H. Hansson and H. Thane, 2000. Using measurements to derive the worst-case execution time. In: Proc. 7th Int. Conf. Real-Time Syst. Applic. (RTCSA), pp: 15-22.

Liu, Y., K.H. Wang, T.T. Hwang and C.L. Liu, 2001. Binary decision Diagrams with minimum expected path length. In: Proc. DATE, pp: 708-712.

Malik, S.A., Wang, R. Brayton and A. Sangiovanni-Vincentelli, 1988. Logic verification using binary decision diagrams in a logic synthesis environment. In: Proc. Int. Conf. Comput. Aided Design (ICCAD), pp: 6-9.

Nagayama, S. and T. Sasao, 2004. On the minimization of longest path length for decision diagrams. International Workshop on Logic and Synthesis (IWLS), Temecula, California, U.S.A., pp: 28-35.

Nagayama, S. and T. Sasao, 2004. On the minimization of longest path length for decision diagrams. International Workshop on Logic and Synthesis (IWLS), pp: 28-35.

Nagayama, S.A., Mishchenko, T. Sasao and J.T. Butler, 2003. Minimization of average path length in BDDs by variable reordering. International Workshop on Logic and Synthesis, Laguna Beach, California, U.S.A., pp: 207-213.

Prasad, P.W.C. and A.K. Singh, 2003.An efficient method for minimization of binary decision diagrams. In: Proc. 3rd Int. Conf. Advances Strategic Technologies, pp: 683-688.

Prasad, P.W.C., A. Assi, A. Harb and V.C. Prasad, 2006. Binary decision diagrams: An improved variable ordering using graph representation of boolean functions. Int. J. Comput. Sci., 1 (1): 1-7.

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

Prasad, P.W.C., M. Raseen, A. Assi and S.M.N.A. Senanayake, 2005. BDD Path Length Minimization based on Initial Variable Ordering. J. Comput. Sci. Publications, USA, 1 (4): 521-529.

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

Raseen, M. P.W.C. Prasad and S.M.N.A. Senanayake, 2005. XOR/XNOR functional behavior on ROBDD Representation. 14th IASTED Int. Conf. Applied Simulation Modeling, Spain, pp: 115-119.

Raseen, M., A. Assi, P.W.C. Prasad and A. Harb, 2005. An efficient mathematical estimation of the BDDs complexity. 16th IASTED Int. Conf. Modeling Simulation, Mexico, pp: 381-386.

Raseen, M., P.W.C. Prasad and A. Assi, 2006. An efficient estimation of the ROBDD's complexity. Integra. VLSI J. Elsevier Publications, UK, 39 (3): 211-228.

Rudell, R., 1993. Dynamic variable ordering for ordered binary decision diagrams. In: Proc. Int. Conf. Comput. Aided Design (ICCAD), pp: 42-47.

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

Somenzi, F., 2001. Efficient manipulation of decision diagrams. Int. J. Software Tools Technol. Trans. (STTT), 3 (2): 171-181.

Shelar, R.S. and S.S. Sapatnekar, 2001. Recursive Bipartitioning of BDD's for Performance Driven Synthesis of Pass Transistor Logic. In: Proc. IEEE/ACM ICCAD, pp: 449-452.

Yang, S. 1991. Logic synthesis and optimization benchmarks user guide version 3.0. Technical report, Microelectronic Centre of North Caroline. Research Triangle Park, NC.