Journal of Computer Science 2 (3): 236-244, 2006
ISSN 1549-3636
© 2006 Science Publications

Empirical Analysis and Mathematical Representation of the Path Length Complexity in Binary Decision Diagrams

[1]A. Assi, [2]P.W.C. Prasad, [2]B. Mills and [2]A. Elchouemi
[1]Department of Electrical Engineering, United Arab Emirates University, Al Ain, UAE
[2]College of Information Technology, United Arab Emirates University, Al Ain, UAE

Abstract: Information about the distribution of path-lengths in a Binary Decision Diagrams (BDDs) representing Boolean functions is useful in determining the speed of hardware and software implementations of the circuit represented by these Boolean functions. This study presents expressions produced from an empirical analysis of a representative collection of Boolean functions. The Average Path Length (APL) and the Shortest Path Length (SPL) have simple behavior as function of the number of variables and the number of terms used in the construction of the Sum of Products (SOPs) in Boolean expressions. We present a generic expression that is uniformly adaptable to each curve of path-length versus number of terms over all the empirical data. This expression makes it possible to estimate the performance characteristics of a circuit without building its BDD. This approach applies to any number of variables, number of terms, or variable ordering method.

Key words: Binary decision diagram, Boolean function, average path length, shortest path length, evaluation time

INTRODUCTION

The use of logic verification and optimization algorithms in VLSI CAD systems requires efficient representation and manipulation of Boolean functions[1]. During the last two decades, BDDs have gained great popularity as successful method for the representation of Boolean functions[2,3]. The ever-increasing complexity of circuit designs is directly related to the complexity of parameters that describe the Boolean function. Over the years, the number of nodes in a BDD became a major concern since it is proportional to the complexity of the Boolean circuit[4]. Over the past two decades most of the problems in the synthesis, design and testing of combinational circuits, have been solved using various mathematical methods[5,6]. Researchers in this area are actively involved in developing mathematical models that predict the number of nodes in a BDD in order to predict the complexity of the design in terms of the time needed to optimize it and verify its logic.

Evaluation time is another crucial parameter of the circuit complexity and it is proportional to the path length of a BDD and one can use BDD structures to estimate the evaluation time of the logic function that represents a circuit[7,8]. Therefore, minimization of the path length can improve the complexity of the circuit implementing a Boolean function, which will eventually enhance the performance of the final implementation. In general the minimization of the path length in Decision Diagrams (DDs) is important in database structures, pattern recognition, logic simulation and software synthesis[7]. The methods proposed for the minimization of APL[7-10] reduces the average evaluation time of logic functions. Most of these methods are based on either Static variable ordering[11,12] or dynamic variable ordering techniques[13]. The minimization of APLs leads to circuits with smaller depth of paths from the root to the terminal node of the BDD. The resulting circuit will be optimized for speed on one hand and on the other hand the number of very long paths in the BDD will be reduced[14]. The minimization of APL is of great importance in real time operating system applications[10,15,16]. The minimization of the LPL (Longest Path Length) and SPL of a BDD can also reduce the evaluation time, which is very important for Pass Transistor Logic (PTL)[7,17,18]. One of the main problems with pass transistor networks is the presence of long paths: the delay of a chain of n pass transistors is proportional to n^2. Inserting buffers can reduce the path length, but this increases the silicon area. So the minimization of the longest evaluation time will improve the performance of the circuit[7,18-20]

Analysis of the BDD methods revealed that the variable ordering in a given Boolean function plays an important role in minimizing the size of the BDD graph as well as minimizing the path length[19,10]. One must go through a number of simulations to find the suitable

Corresponding Author: A. Assi, Department of electrical Engineering, United Arab Emirates University, P.O. Box 17555, Al Ain, U.A.E., Tel: 971-3- 7133609, Fax: 971-3-7626309

variable ordering that leads to the minimum size of the BDD and minimum Path Length. In this approach we need to create the whole BDD representing the Boolean function with the best possible variable ordering. Building the whole BDD may lead to some complexity in the design process in terms of the time required to implement, verify and test the design. It will be useful to have a kind of estimation of the BDD complexity prior to make decisions on the feasibility of the design[20]. For any combinational circuit the only available initial information is the Boolean function that represents this circuit and the number of its variables. This information is usually considered to design and verify circuits using well known mathematical methods.
There has been a lot of research[21-24] done on the estimation of combinational and sequential circuit parameters from the exact Boolean function describing the circuit. What distinguishes this study and prior work[20,25-27] by the some of the current authors is the use of stochastic technique and estimation of parameters from only partial information about the Boolean function.

It is very hard to perform a comparison without having an idea about the path length size for a given number of variables. Therefore, it is important to develop a mathematical model that predicts the path length, knowing the number of variables and the number of product terms of the Boolean function represented by this BDD.

The main objective of this study is to enhance the methodologies proposed in[20,25] to estimate the path length complexity for the Boolean functions represented by the BDD. First, we present experiments that show the behaviors of the APL and SPL and then we extract a unique mathematical model for produced experimental graphs. This study is organized as follows: First is an introduction, followed by the necessary terminologies and definitions of the BDD and path length. . Later we review the previous work done on the estimation of the BDD complexity. The proposed method with the experimental results followed by the mathematical model is given next. Finally the advantages of this mathematical model followed by an outline of our future developments in this research work and conclusion.

PRELIMINARIES

Basic definitions for BDDs and path length are given in[1,3,4,7,10]. 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 (or *then*) and 0 (or *else*). Each non-sink node represents the Boolean function corresponding to its edge "1" if $v = 1$, or the Boolean function corresponding to its edge "0" 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 and always in the same order along each path.

Definition 3: A *Reduced OBDD* (ROBDD) is an OBDD which no nodes have equivalent behavior.

Variable ordering: The size of a BDD is largely affected and its variation can be linear or exponential depending on the choice of the variable ordering in building the BDD. Figure 1 illustrates the effect of the variable ordering [R.E. Bryant, 1986] on the size of BDDs for the Boolean function (1):

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

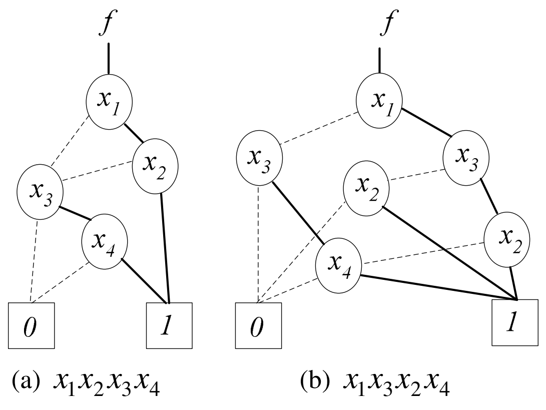

Fig. 1: Effect of the variable ordering on the size of BDD

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

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 variables whose path includes e_{i0} (or e_{i1}), where e_{i0} (or e_{i1}) denotes edge "0" (or the edge "1") directed from away node V_i [7,8]. 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 (2) to calculate the $P(V_i)$ for the rest of the nodes:

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

Definition 5: The *APL* is equal to the sum of the node traversing probabilities of the non-terminal nodes[7,10]. Node traversing probability denoted by $P(v_i)$ is the fraction of all 2^n assignments of values to the variables whose path includes node v_i. The APL can be expressed by the following equation (3):

$$APL = \sum_{i=0}^{N-1} P(v_i) \qquad (3)$$

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

Definition 6: *The Shortest Path Length (SPL)* of a BDD denoted by SPL (BDD), is the *Length of the Shortest Path* from the root node to the terminal node.

Example: Consider the BDD graph shown in Fig. 2. In this example we will compute the APL and the SPL:

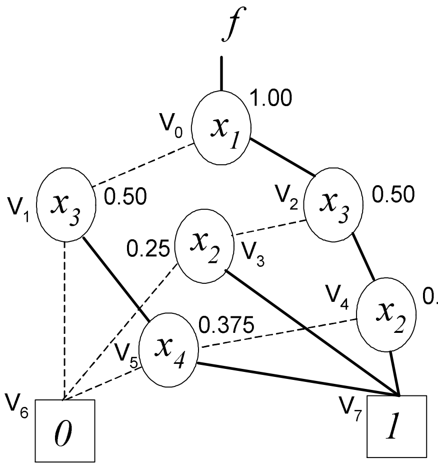

Fig. 2: Node Traversing Probability in a BDD

The root node $P(V_0)$ is always equal to 1.00.

$$P(V_1) = P(e_{0_0}) = 0.50$$

$$P(V_2) = P(e_{1_0}) = 0.50\ .$$

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

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

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

Finally

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

$$LPL = Shortest\ Path\ Length = x_1 \to x_2 = 2$$

Previous work: Here, we provide a brief description of the works done in the area of the estimation of BDD complexity prior to explaining the proposed method.

Relation between the size of a boolean function and the ROBDD complexity[20]: The complexity of the ROBDD mainly depends on the number of nodes represented by the ROBDD. Analysis of the complexity variation in ROBDDs i.e. the relation between the number of product terms and the number of nodes for any number of variables is discussed in these works, the experimental graph variation reveals that the complexity of the ROBDD can be modeled mathematically by equation (4). Figure 3 indicates that the mathematical model represented by equation (4) provides a very good approximation of the ROBDD complexity.

$$NN = \alpha \cdot NPT^{\beta} \cdot e^{(-NPT \cdot \gamma)} + 1 \qquad (4)$$

Where, *NN* is the number of nodes that represents the complexity of ROBDD, *NPT* is the number of non-repeating product terms in the Boolean function, α, β and γ are three constants. Using curve fitting techniques, the variations of α, β and γ were mathematically modeled and represented by the following equations (5), (6) and (7).

$$\alpha = 0.9855 \cdot e^{(0.063 \cdot NV^{1.51})} \qquad (5)$$

$$\beta = 1.031149 \cdot e^{(-0.01551933 \cdot NV)} + 67.2072 \cdot e^{(-1.2985 \cdot NV)} \qquad (6)$$

$$\gamma = 0.962297281 \cdot e^{(-0.4187691 \cdot NV)} + 41.9723 \cdot e^{(-1.5072 \cdot NV)} \qquad (7)$$

Where, *NV* is the Number of Variables.

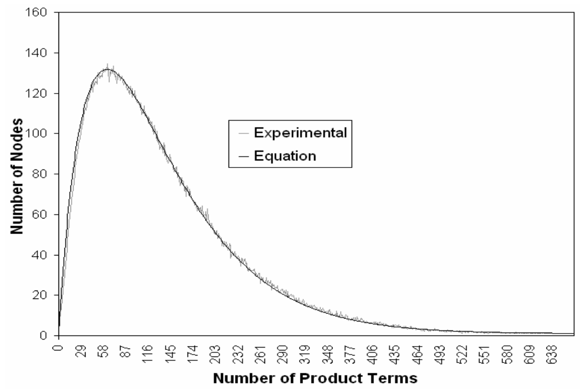

Fig. 3: Experimental/Equation BDD Complexity for 10 variables

Behavior of XOR/XNOR Min-term Representations[26]: In this work, the complexity variation in ROBDD for a specific group of XOR/XNOR min-terms is analyzed. A graph that represents the ROBDD complexity in terms of number of nodes with respect to the number XOR/XNOR min-terms of the Boolean function is then plotted and the behavior of XOR/XNOR is modeled mathematically by equation (8): Figure 4 show that the mathematical model represented by equation (8) provides a good approximation of the experimental ROBDD complexity.

$$NN = \alpha \cdot \left[\beta^2 - (NXM - \beta)^2 \right]^{0.5} + 1 \qquad (8)$$

Where, *NN* is the number of nodes that represents the complexity of ROBDD, *NXM* is the number of XOR/XNOR min-terms in the Boolean function, β is 2n-2 with *n* the number of input variables and α = 0.605234.

ANALYSIS OF THE COMPLEXITY OF PATH LENGTH IN BDDS

Proposed method: An experiment was carried out using Colorado University Decision Diagram (CUDD) Package[28] to analyze the complexity variation of SPL with the number of product terms for any number of variables. For each variable count *n* between 1 and 14 inclusive and for each term count between 1 and 2^n-1, 100 Boolean functions were randomly generated and the APL and SPL average was determined by using CUDD package for specific variable ordering technique. This process was repeated until the average size of the APL and SPL complexities became 1. Then the experimental graphs for APL and SPL complexities were plotted against the product term count for each variable count.

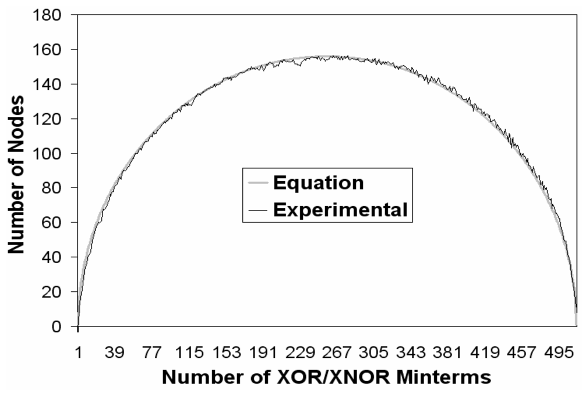

Fig. 4: Experimental/Equation ROBDD Complexity for XOR/XNOR Min-terms

Experimental analysis for APL and SPL complexity variations: Figure 5 and 6 illustrates the APL and SPL complexity relation for Boolean functions with product terms having *n*=10 variables using the Symmetric Sift reordering technique of the CUDD tool.

The graph indicates that the complexity (i.e. size) of the path length in general (APL and SPL) increases as the number of product terms increases. This is clear from the rising edge of the curve shown in Fig. 5 and 6. At the end of the rising edge in the graph, the size of the APL and SPL reaches a maximum ($APL \cong 7.73$, $SPL \cong 5.4$ in this case).

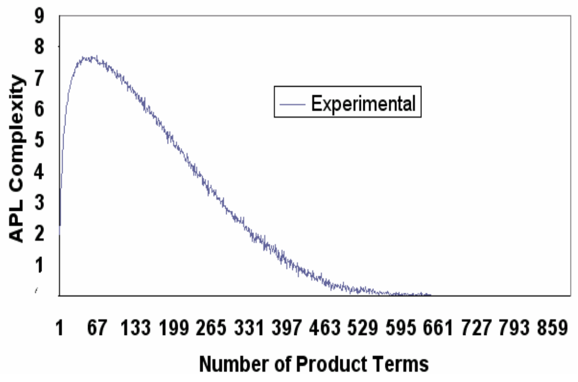

Fig. 5: APL size variation for 10 variables using the symmetric Sift reordering technique

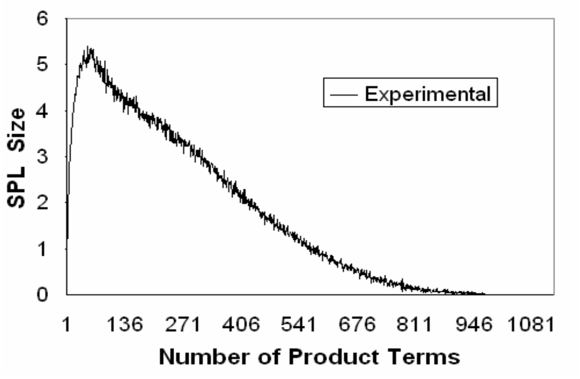

Fig. 6: SPL size variation for 10 variables using the Symmetric Sift reordering technique

This peak indicates the maximum APL and SPL that any Boolean function with 10 variables can produce independently of the number of product terms.

Apart of that the peak also specifies the number of product terms (critical limit) of a Boolean function that leads to the maximum number of APL and SPL for any Boolean function with 10 variables. The number of product terms that leads to the maximum APL and SPL is 66 and 50 respectively. If the number of product terms increases above the critical limit, as expected, the product terms starts to simplify and the BDD will reduce, which will reduce both the path lengths (APL and SPL) size.

The APL and SPL complexity graphs shown in Fig. 5 and 6 indicate that as the number of product terms increases the complexity of the APL and SPL decreases in a slower rate and ultimately reaches 0. Figure 6 illustrates that the falling edge of the SPL graph behaves a bit different than the other complexity graphs shown in Fig 2 and 3, where the decrease is with a roll off, to be independent of the variable count. The

APL complexity variation graph is fairly similar to the Fig. 2, but the roll of is not steeper as the Fig. 2.
The location and height of the peak and the slope of the logarithm of the roll off varied. Reduction of the APL and SPL complexity to 0 implies that all the product terms simplify to logic 1. A simple algebraic expression for these curves was developed, unifying all the cases.

MATHEMATICAL MODEL FOR THE PATH LENGTH BEHAVIOR

Exponentials of rational polynomials fitted the data well; but, a theoretical precedent was not apparent. On the other hand, $\log(t+1)/(t+1)$ not only has the same basic behavior, but is also implicated in other complexity measures, such as Kolmogorov, Tichner, Shannon and Lempel-Zif complexity, as well as the density of the prime numbers. The generic SPL graph has an initial rise, two peaks and roll off to zero, suggesting the sum of two formulas, but with horizontal and vertical scaling and a little peak shaping. We note here that the second peak is not always a peak of the curve, but it is a peak of the difference between the curve and the best affine approximation in that region. The generic APL graph has an initial rise which is similar to SPL rise, but with only one peak and a roll off to 0. Analyzing all the above factors for the behavior of the APL and SPL graphs, the complexity behavior was modeled mathematically by the following equations:

$$1 = \sum_{i=1}^{2} \left(\frac{\log(t+1)}{(\log(t+1))^{\beta_i}} \right)^{\alpha_i} \qquad (9)$$

Where, t is the number of product terms in the Boolean function. The (mostly) constants α and β parameters affect the shape of the peak.

For the SPL, the values $\alpha_1 = 7$, $\beta_1 = 1$ and $\alpha_2 = 10$ gave a close fit, but β_2 taking on two distinct values. $\beta_2 = 3$ for $v \le 11$ and $\beta_2 = 5$ for $v \ge 12$. Eventually the following equation (10) was used in order to calculate the constant β_2,

$$\beta_2 = 3 + \left(\frac{1.8}{[e^{(v-11.5)}]^2 + 1} \right) \qquad (10)$$

It can be inferred from Fig. 6 that the curve has two peaks, which needs four scaling parameters to define the locations of the peaks (Fig. 7): i.e. (x_1, y_1) and (x_2, y_2).
For the APL, the alues $\alpha_1 = 7$, $\beta_1 = 0.7$ and $\alpha_2 = 10$ gave a close fit, but β_2 also taking on two distinct values. $\beta_2 = 2.1$ for $v \le 11$ and $\beta_2 = 3.5$ for $v \ge 12$.
Eventually the following equation (11) was used in order to calculate the constant β_2 for APL.

$$\beta_2 = 0.7\left(3 + \left(\frac{1.8}{[e^{(v-11.5)}]^2 + 1} \right) \right) \qquad (11)$$

The final behavior of the APL and SPL curve can be found by the following single equation (12):

$$1 = \sum_{i=1}^{2} \cdot y_i \left(\frac{\log\left(t/x_i + 1 \right)}{\left(\log\left(t/x + 1 \right) \right)^{\beta_i}} \right)^{\alpha_i} \qquad (12)$$

In this mathematical model, the peaks (x_i, y_i) for both the APL and SPL curves were found by performing an empirical fit for each time. Figure 8 and 9 depict the experimental results obtained for APL and SPL using the CUDD package and the theoretical results obtained using equation (12). The mathematical model represented by equation (12) provides a very good estimation for the APL and SPL complexity behavior, where the experimental and equational results produced a match. Further verification of the mathematical model was done for Boolean functions with 2 to 15 variables. It can be inferred that the experimental and mathematical curves are following a similar pattern for any number of variables. Figure 10-13 illustrates the experimental and mathematical models for APL and SPL for variables 8 and 12 respectively.

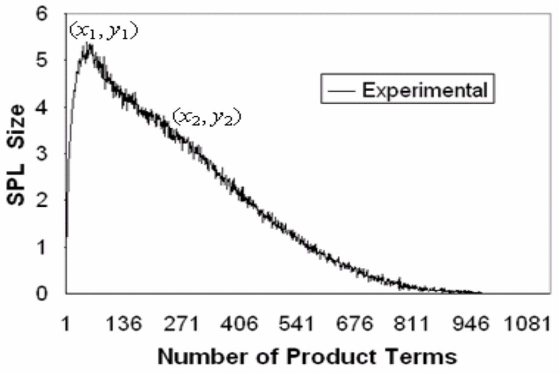

Fig. 7: Peaks of the SPL complexity behavior

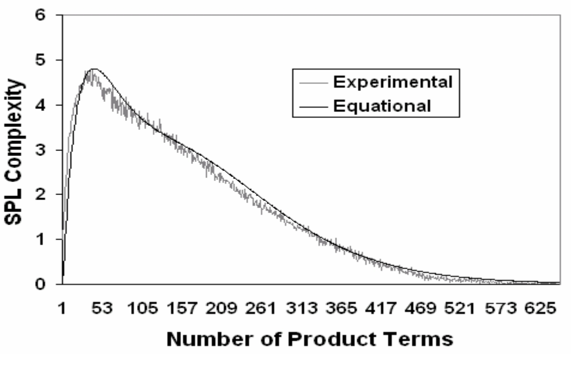

Fig. 8: Experimental/Equation SPL complexity behavior for 10 variables

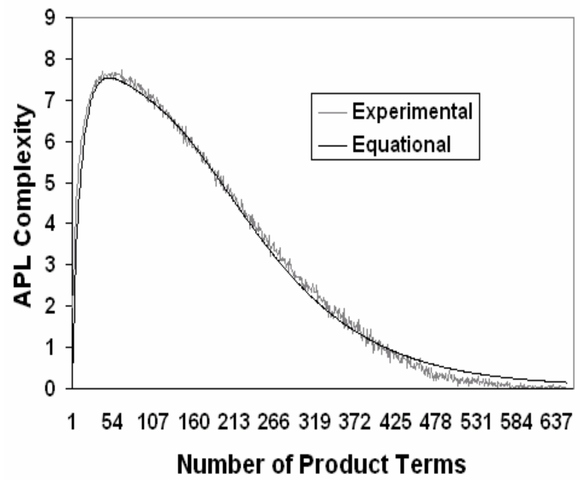

Fig. 9: Experimental/Equation APL Complexity behavior for 10 variables

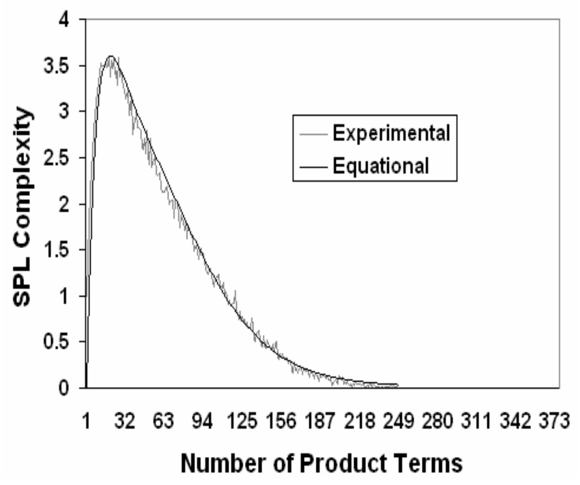

Fig. 12: Experimental/Equation SPL Complexity behavior for 8 variables

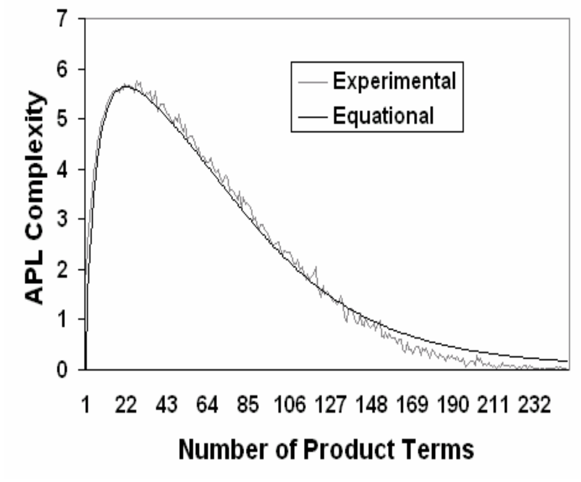

Fig. 10: Experimental/Equation APL Complexity behavior for 8 variables

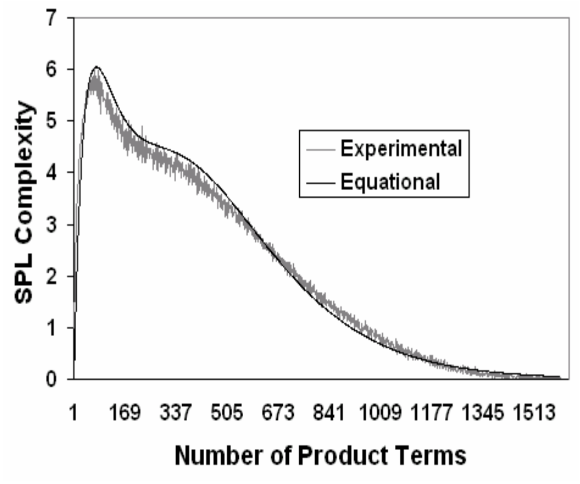

Fig. 13: Experimental/Equation SPL Complexity behavior for 12 variables

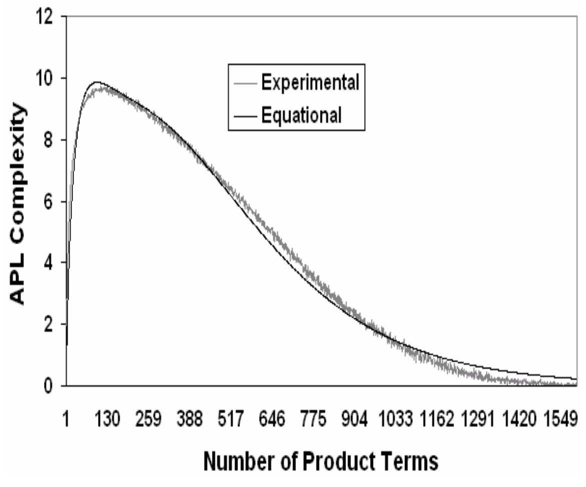

Fig. 11: Experimental/Equation APL Complexity behavior for 12 variables

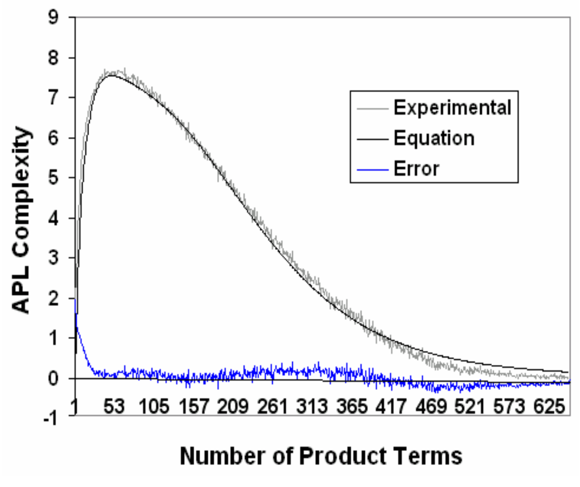

Fig. 14: APL Complexity Estimation Error for 10 variables

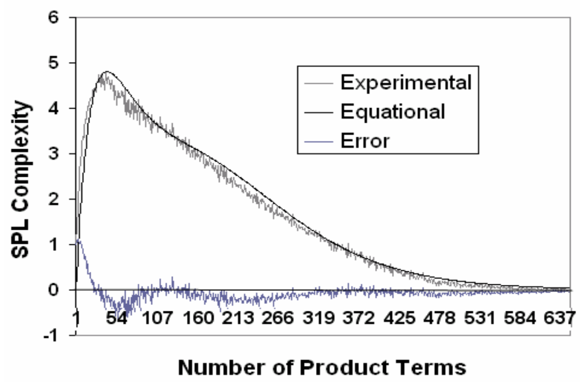

Fig. 15: SPL Complexity Estimation Error for 10 variables

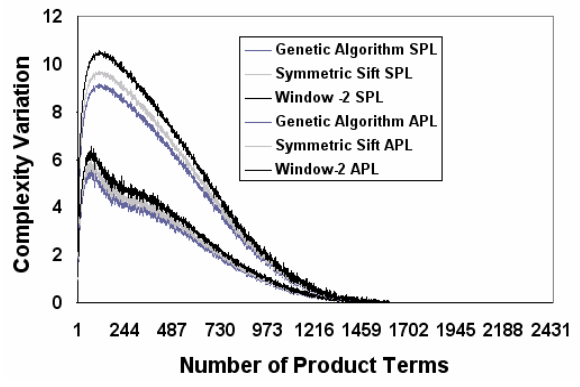

Fig. 16: Effect of the reordering methods for SPL and APL Complexity variations

Figure 14-15 shows the efficiency of the proposed mathematical model, which produces complexity estimation error for APL and SPL**.** It can be inferred that the mathematical expression was able to match the experimental curve with minimum error, which is less then ± 0.01 for most of the Product terms.

Effect of the reordering methods on path length complexity variations: The experiment done earlier using the Symmetric Sift CUDD reordering method was extended here to understand the relation of Symmetric Sift APL and SPL graphs with other reordering techniques. It was observed that the relation between the graphs follows the same pattern and it varies only on the amplitude factor of the curves.

By analyzing the effect of the reordering methods on the model, equation (12) can be modified with an additional amplification factor (μ). The amplification factor is 1 for the Symmetric Sift, greater than 1 for methods with lower efficiency and less than 1 for methods with higher efficiency than the Symmetric Sift. Equation (13) represents the mathematical model for the APL and SPL for any reordering method.

$$1 = \mu \cdot \sum_{i=1}^{2} \cdot y_i \left(\frac{\log\left(t/_{x_i} + 1 \right)}{\left(\log\left(t/_{x} + 1 \right) \right)^{\beta_i}} \right)^{\alpha_i} \qquad (13)$$

The amplification factor was calculated and depicted in Table 1. Figure 16 shows the comparison graphs of the APL and SPL behaviors for Symmetric Sift with two of the other CUDD variable ordering techniques mainly the Genetic Algorithm and Window2. These two graphs show that the efficiency of the reordering method has a definite impact on the path length complexity; an efficient variables ordering leads to a reduced number of nodes, which leads to reduced path lengths.

Table 1: Amplification factor (μ)

Variable Reordering Method	Amplification Factor (μ)
Random	1.024
Random Pivot	0.998
Sift	1.001
Symmetric Sift	1.000
Symmetric Sift Converge	0.971
Group Sift	1.006
Group Sift Converge	0.963
Window 2	1.085
Window 3	1.045
Window 4	1.018
Window Converge 2	1.058
Window Converge 3	1.025
Window Converge 4	0.989
Annealing	0.945
Genetic Algorithm	0.942
Exact	0.942

Advantages: The developed mathematical model represented by equation (10), provides some useful information on the following, without the need of building the BDD.

1. The complexity behavior of the APL and SPL, given the number of product terms of the Boolean function
2. The number of product terms for which the maximum possible depth will occur.
3. The maximum complexity of the APL and SPL of Boolean functions for any number of variables.

CONCLUSION AND FUTURE WORK

Future work includes minimizing the Complexity estimation error of the match and to develop experiments to include larger number of variables. We are in the process of investigating an automated global fit for any SPL and APL curves in order to find the complexity for any number of product terms. Investigating a mathematical model for other BDD characteristics (i.e. longest path length and number of paths) is also considered.

We have discussed the idea of using BDD to study and model a relationship between the path lengths and the number of product terms in a Boolean function. Analyzing the Experimental results, we have introduced a single and unique mathematical model, which is based on an empirical fit that can predict valuable information related to the APL and SPL behaviors without building the BDD. A great reduction in time complexity for digital circuits' designs can be achieved and the model can also offer useful information on the design to handle the minimization of its evaluation time prior to its implementation. Our experimental results show good correlation between the experimental results and those given by the mathematical model.

REFERENCES

1. Priyank, K., 2000. 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.
2. Akers, S.B., 1978. Binary decision diagram. IEEE Trans. Computers, 27: 509-516.
3. Bryant, R.E., 1986. Grap̃h based algorithm for boolean function manipulation. IEEE Trans. Computers, 35: 677-691.
4. Drechsler, R. and D. Sieling, 2001. Binary Decision Diagrams in Theory and Practice. Springer-Verlag Trans., pp: 112-136.
5. Jain, A., M. Narayan and A.S. Vincentelli, 1997. Formal verification of combinational circuits. Proc. Intl. Conf. on VLSI Design, pp: 218-225.
6. Van Eijk, C.A.J., 1997. Formal methods for the verification of digital circuits. Ph.D. Thesis, Eindhoven University of Technology, Netherlands.
7. Nagayama, S. and T. Sasao, 2004. On the minimization of longest path length for decision diagrams. Proc. of Intl. Workshop on Logic and Synthesis (IWLS-2004), pp: 28-35.
8. Liu, Y., K.H. Wang, T.T. Hwang, and C.L. Liu, 2001. Binary decision diagrams with minimum expected path length. Proceedings of DATE 01, pp: 708-712.
9. Prasad, P.W.C., M. Raseen, S.M.N.A. Senanayake and A. Assi, 2005. BDD path length minimization based on initial variable ordering. accepted for Publication in Journal of Computer Science, Science Publications, May 2005.
10. Ebendt, R., S. Hoehne, W. Guenther and R. Drechsler, 2004. Minimization of the expected path length in BDDs based on local changes. Proc. of Asia and South Pacific Design Automation Conf. (ASP-DAC'2004), pp: 866-871.
11. Fujita, M., H. Fujisawa and N. Kawato, 1988. Evaluation and improvements of boolean comparison method based on binary decision diagrams. Proc. Intl. Conf. on Computer Aided Design (ICCAD)., pp: 2-5.
12. Malik, S., A. Wang, R. Brayton and A.S. Vincentelli, 1988. Logic verification using binary decision diagrams in a logic synthesis environment. Proc. Intl. Conf. on Computer Aided Design (ICCAD)., pp: 6-9.
13. Rudell, R., 1993. Dynamic variable ordering for ordered binary decision diagrams. Proc. Intl. Conf. on Computer Aided Design (ICCAD)., pp: 42-47.
14. Fey, G., J. Shi and R. Drechsler, 2004. BDD circuit optimization for path delay fault-testability. Proc. of EUROMICRO Symp. on Digital System Design, pp: 168-172.
15. 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: 834-849.
16. Lindgren, M., H. Hansson and H. Thane, 2000. Using measurements to derive the worst-case execution time. Proc. 7th Intl. Conf. on Real-Time Systems and Applications (RTCSA'00)., pp: 5-22.
17. Shelar, R.S. and S.S. Sapatnekar, 2001. Recursive bipartitioning of BDD's for performance driven synthesis of pass transistor logic. Proc. of IEEE/ACM ICCAD., pp: 449 - 452.
18. Bertacco, V., S. Minato, P. Verplaetse, L. Benini and G.D. Micheli, 1997. Decision diagrams and pass transistor logic synthesis. Stanford University CSL Technical Report, No. CSL-TR-97-748.
19. Görschwin, F., S. Junhao and R. Drechsler, 2004. BDD circuit optimization for path delay fault-testability. Proc. EUROMICRO Symp. on Digital System Design, pp: 168-172.
20. Raseen, M., P.W.C. Prasad and A. Assi, 2005. An efficient estimation of the ROBDD's complexity. accepted for Publication in Integration - the VLSI Journal, Elsevier Publication.
21. Nemani, M. and F.N. Najm, 1996. High-level power estimation and the area complexity of boolean functions. Proc. of IEEE Intl. Symp. on Low Power Electronics and Design, pp: 329-334.
22. Dunne, P.E. and W. van der Hoeke, 2004. Representation and complexity in boolean games. Proc. 9th European Conf. on Logics in Artificial Intelligence, LNCS 3229, Springer-Verlag, pp: 347-35.

23. Ramalingam, N. and S. Bhanja, 2005. Causal probabilistic input dependency learning for switching model in VLSI circuits. Proc. of ACM Great Lakes Symp. on VLSI, pp: 112-115.
24. Bhanja, S., K. Lingasubramanian and N. Ranganathan, 2005. Estimation of switching activity in sequential circuits using dynamic bayesian networks. Proc. of VLSI Design, pp: 586-591.
25. Raseen, M., P.W.C. Prasad and A. Assi, 2004. Mathematical model to predict the number of nodes in an ROBDD. Proc. 47th IEEE Intl. Midwest Symp. on Circuit and Systems (MWSCAS), 3: 431-434.
26. Prasad, P.W.C., M. Raseen and S.M.N.A. Senanayake, 2005. XOR/xnor functional behavior on ROBDD representation. Proc. 14th IASTED Intl. Conf. on Applied Simulation and Modelling (ASM 2005), pp. 115-119, Spain.
27. Raseen, M., A. Assi, P.W.C. Prasad and A. Harb, 2004. Effect of boolean min-terms on the complexity of ROBDDs. Proc. Intl. Conf. on Computational Intelligence (ICCI 2004), pp: 454-457.
28. Somenzi, F., 2003. CUDD: CU Decision Diagram Package. <ftp://vlsi.colorado.edu/> pub/.