EVALUATION TIME OF BOOLEAN FUNCTIONS: AN ESTIMATION METHODOLOGY BASED ON THE PATH LENGTH OF THEIR BDD REPRESENTATIONS

P.W.C. Prasad, M. Raseen and B.I. Mills
College of Information Technology, UAE University, P.O. Box17555, UAE.
A. Assi
American University of Technology, Department of Computer Engineering, Lebanon.
Correspondence Email: prasadc@uaeu.ac.ae

ABSTRACT

In digital systems where Boolean functions are frequently manipulated, it is important to know how evaluation time of Boolean functions is consumed during their execution by the processor. Estimation of the evaluation time of Boolean functions plays an important role in function-architecture co-design. The evaluation time complexity of Boolean functions represented by Binary Decision Diagrams (BDDs) is directly related to the path length of the BDD. This paper describes a BDD approach that gives an estimation method for the time evaluation of Boolean functions. The proposed technique is validated using both experimental and mathematical techniques.

Keywords: Estimation, Binary Decision Diagram (BDD), Path Length, Boolean function

1. INTRODUCTION

BDDs are well known structures used in the design verification and optimization of digital circuits [1]. For the last two decades, BDDs have gained great popularity in representing discrete functions. A BDD in general is a direct acyclic graph representation of Boolean functions proposed by Akers and Bryant [2], [3].The success of this technique has attracted researchers in the area of VLSI CAD systems [1], [4].

The efficiency of a BDD depends mainly on the size of its graph representation. Over the years the complexity of BDD implementation kept increasing, and the number of nodes in a BDD became a major concern, since it is directly related to the time and space requirements of the digital circuit represented by this BDD [5], [6]. Today, researchers in this area are actively involved in the search for methods to minimize the size of BDDs and consequently improve the overall performance of the designed systems.

One of the important tasks during the design phase of a digital system is to prove the efficiency by checking whether the design fulfills the requirements. Evaluation time is one of the important factors which use the BDD to evaluate logic functions [7]. The evaluation time is not directly related to the number of nodes in a BDD, but it is proportional to the path length of the BDD. Therefore, minimization of the path length can improve the performance of the circuit implementing a Boolean function, which will eventually increase the quality of the final implementation [8], [9]. In general the minimization of the path length in Decision Diagrams (DD) is important in database structures, pattern recognition, logic simulation and software synthesis [8]. The minimization of path lengths are of great importance in embedded systems, real time operating system applications [10], [11] and also for Pass Transistor Logic (PTL) [12], [13].

In order to find the minimum Shortest Path Length (SPL), 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 term of time required to implement, verify and test the design. It will be useful to have a kind of estimation for the path length size prior to make decisions on the feasibility of the design.

The main objective of this paper is to introduce a novel method for the prediction of the SPL of BDDs based on mathematical model. The proposed model will provide some valuable

information about the path length complexity for any number of variables and for any type of variable ordering method which will enable the system performance to be analyzed without building the BDD. It is also capable of predicting the number of product terms in the Boolean function that leads to the maximum SPL complexity. The remaining of this paper is divided as follows: in the second section, we provide the background information pertaining to the BDD and the path length of a BDD. Section three reviews the previous work done on estimation of BDD complexities. The proposed method with the experimental results followed by the mathematical model are given in section four and five respectively. The advantage of this mathematical model is given in section six. Finally, we conclude our paper with future developments in this research work.

2. PRELIMINARIES

Basic definitions for BDDs and path length are given in [1], [3], [5], [8], [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.

2.1 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 [3] on the size of BDDs for the Boolean function (1):

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

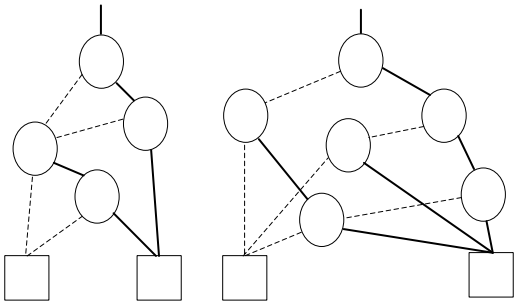

Figure 1: Effect of the variable ordering on the size of BDD

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 5: The *APL* is equal to the sum of the node traversing probabilities of the non-terminal nodes [8], [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 (2):

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

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 root node to terminal node.

3. PREVIOUS WORK

In this section we will briefly describe works done in the area of the estimation of BDD complexity prior to explaining the proposed method.

3.1 Relation between the Size of a Boolean Function and the ROBDD Complexity

The complexity of the ROBDD mainly depends on the number of nodes represented by the ROBDD. An experiment was done in [14], [15] to analyze 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. The experimental graph variation reveals that the complexity of the ROBDD can be modeled mathematically by the equation (3). Figure 2 indicates that the mathematical model represented by equation (3) provides a very good approximation of the ROBDD complexity.

$$NN = \alpha \cdot NPT^{\beta} \cdot e^{(-NPT \cdot \gamma)} + 1 \qquad (3)$$

Where,
NN is the number of nodes (Complexity of ROBDD), NPT is the number of non-repeating product terms in the Boolean function, α, β and γ are three constants. For 10 variables the values of the constants are $\alpha = 7.7$, $\beta = 0.904$ and $\gamma = 0.0145$.

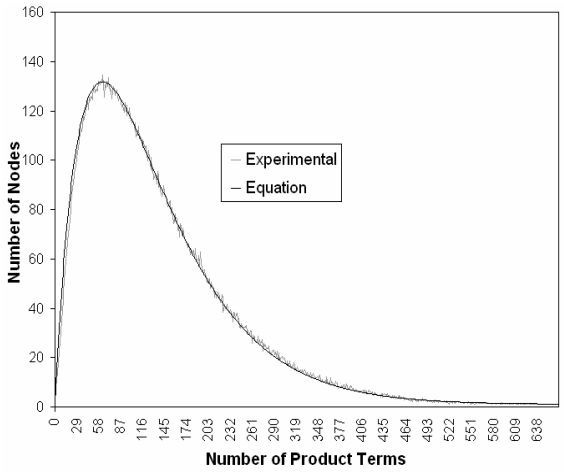

Figure 2: Experimental/Equation BDD Complexity for 10 variables.

3.2 Behavior of XOR/XNOR Min-term Representations

This work analyzed the complexity variation in ROBDD for a specific group of XOR/XNOR min-terms [16]. 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 was modeled mathematically by equation (4): Figure 3 show that the mathematical model represented by equation (4) provides a good approximation of the experimental ROBDD complexity.

$$NN = \alpha \cdot \left[\beta^2 - (NXM - \beta)^2\right]^{0.5} + 1 \qquad (4)$$

where, *NN* is the number of nodes (Complexity of ROBDD), *NXM* is the number of XOR/XNOR min-terms in the Boolean function, and β is 2^{n-2} (*n*-number of inputs) and $\alpha = 0.605234$.

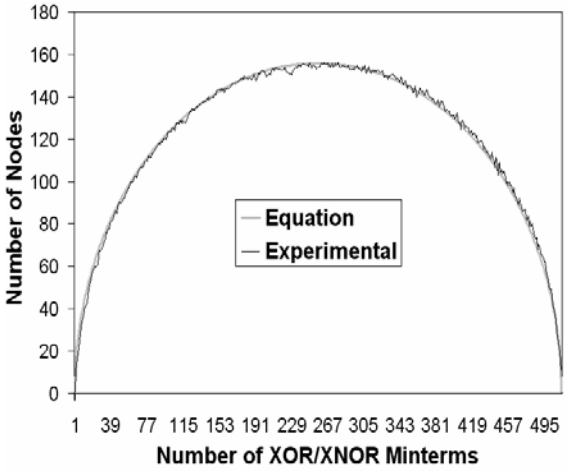

Figure 3: Experimental/Equation ROBDD Complexity for XOR/XNOR Min-terms

4. BDD PATH LENGTH COMPLEXITY ANALYSIS

4.1 Proposed Method

An experiment was carried out using Colorado University Decision Diagram (CUDD) Package [17] 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 SPL average was determined by using CUDD package for specific variable ordering technique and was plotted against term count for each variable count. The following algorithm explains this method in detail.

Step 1: The number of variables (*n*) is fixed (for example *n* =10 variables)
Step 2: The variable ordering technique is selected (Ex: Symmetric Sift is chosen from the CUDD)
Step 3: Number of product terms (*m*) is set to 1
Step 4: A random Boolean function with *m* product terms is generated.
Step 5: Run the CUDD in order to build the BDD for the Boolean function
Step 6: Reorder the BDD using the variable ordering selected in step 2.
Step 7: Record the number of nodes in the ROBDD.
Step 8: Steps 4 to 7 are repeated 100 times to build 100 different BDDs for different random equations with the same number of product terms. For each BDD the APL and SPL are calculated and the mean sizes are recorded.
Step 9: Steps 4 to 8 are repeated for *m* = 2, 3, 4, etc. until the average size of the SPL complexities becomes 1.
Step 10: A graph that represents the SPL complexity in terms of number of nodes with respect to the number product terms of the Boolean function is then plotted.

4.2 SPL variations against the size of the Boolean function

Figure 4 illustrates the SPL complexity relation for Boolean functions with product terms having *n*=10 variables using the Symmetric Sift reordering technique. The Symmetric Sift reordering technique was used as the base variable ordering method, due to its efficiency compared to all the CUDD methods.

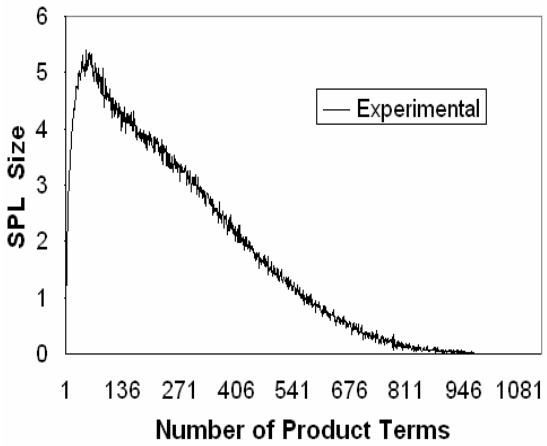

Figure 4: SPL size variation for 10 variables using the Symmetric Sift reordering technique.

The graph indicates that the complexity (size) of the path length in general (SPL) increases as the number of product terms increases. This is clear from the rising edge of the curve shown in Figures 4. At the end of the rising edge in the graph, the size of the SPL reaches a maximum ($SPL \cong 5.4$ in this case). This peak indicates the maximum 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 SPL for any Boolean function with 10 variables. The number of product terms that leads to the maximum SPL is 50. 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 the SPL size.

The SPL graph shown in Figure 4 indicates that as the number of product terms increases the complexity of the SPL decreases in a slower rate and ultimately reaches to 0. Figure 4 illustrates that the falling edge of the SPL graph behaves a bit different than the other complexity graphs shown in Figure 2 and Figure 3, where the decrease is with a roll off, to be independent of the variable count. The location and height of the peak and the slope of the logarithm of the roll off varied. Reduction of the 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.

5. 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. Under suitable conditions, if $f(x)$ has a peak, then, $\left(f(x)/f_{\max}\right)^{\alpha} \cdot f_{\max}$ has a broader or narrow peak for $\alpha \langle 1 \text{ or } \alpha \rangle 1$. Analyzing all the above factors for the behavior of the SPL graph, the complexity behavior was modeled mathematically by the following equations:

$$1 = \sum_{i=1}^{2} \left(\frac{\log(t+1)}{(\log(t+1))^{\beta_i}} \right)^{\alpha_i} \qquad (5)$$

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. 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 (6) was used in order to calculate the constant β_2,

$$\beta_2 = 3 + \left(\frac{1.8}{[e^{(v-11.5)}]^2 + 1} \right) \qquad (6)$$

It can be inferred from the Figure 4 that the curve has two peaks (Figure5), which needs four scaling parameters to define the locations of the peaks: i.e. (x_1, y_1) and (x_2, y_2)

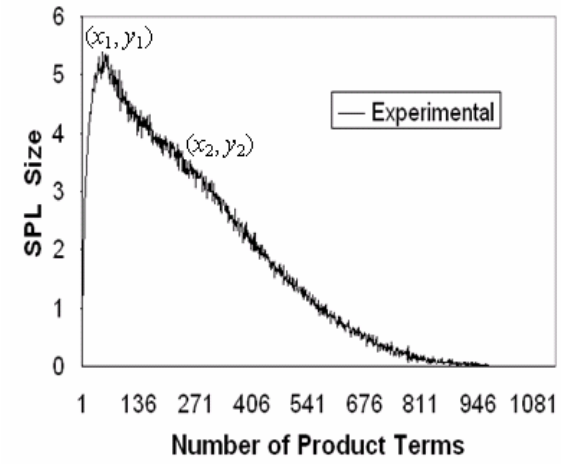

Figure 5: Peaks of the SPL complexity behavior

The final behavior of the SPL curve can be found by the following equation:

$$1 = \sum_{i=1}^{2} \cdot y_i \left(\frac{\log\left(t/x_i + 1 \right)}{\left(\log\left(t/x + 1 \right) \right)^{\beta_i}} \right)^{\alpha_i} \qquad (7)$$

In this mathematical model, the peaks (x_i, y_i) were found by performing an empirical fit for each time. Figure 6 depicts the experimental results obtained by the CUDD package and the theoretical results obtained using equation (7). It shows that the mathematical model represented by equation (7) provides a very good estimation for the SPL complexity behavior, where the experimental and equational results produced a match.

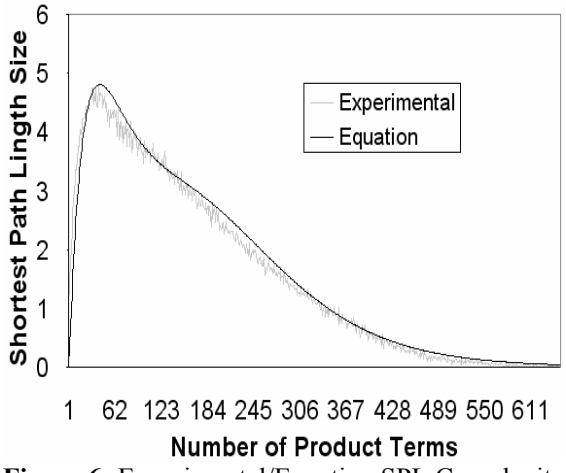

Figure 6: Experimental/Equation SPL Complexity behavior for 10 variables

Further verification of the mathematical model was done by matching the model for Boolean functions with 2 to 15 variables. It can be inferred that the experimental and mathematical curves are following a similar pattern for all the variable levels. Figure 7 and Figure 8 illustrate the experimental and mathematical models for variables 8 and 12 respectively.

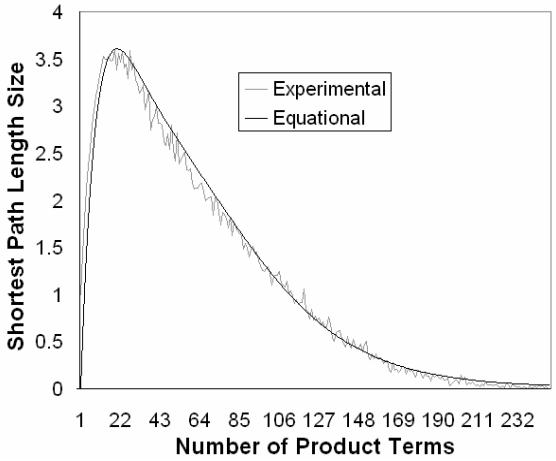

Figure 7: Experimental/Equation SPL Complexity behavior for 8 variables

5.1 Effect of the Reordering Methods on Path length Variations

The experiment done in section 4 using the Symmetric Sift CUDD reordering method was extended to understand the relation of Symmetric Sift 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.

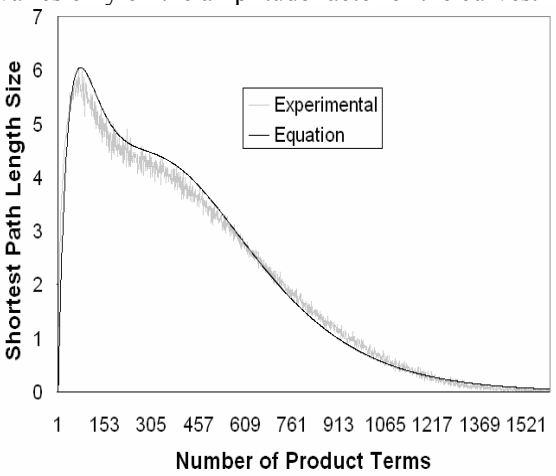

Figure 8: Experimental/Equation SPL Complexity behavior for 12 variables

By analyzing the effect of the reordering methods on the model, equation (7) 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 lesser than 1 for methods with higher efficiency than the Symmetric Sift. Equation (8) represents the mathematical model for the SPL for any reordering method.

$$1 = \mu \cdot \sum_{i=1}^{2} \cdot y_i \left(\frac{\log\left(t/x_i + 1 \right)}{\left(\log\left(t/x + 1 \right) \right)^{\beta_i}} \right)^{\alpha_i} \qquad (8)$$

The amplification factor was calculated and depicted in table 1. Figure 9 shows the comparison graphs of the SPL behavior 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

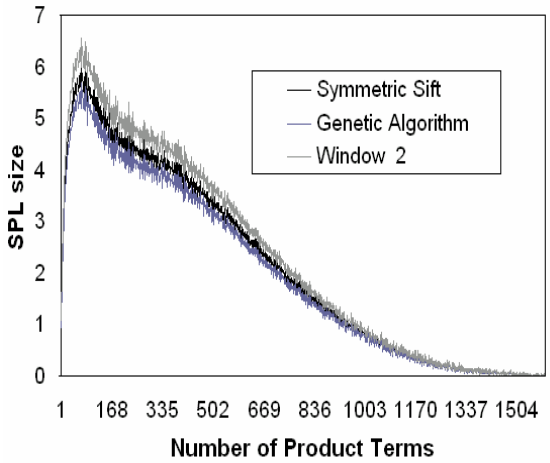

Figure 9: Effect of the reordering methods for SPL variations

6. ADVANTAGES

The developed mathematical model represented by equation (7), provides some useful information on the following, without the need of building the BDD.

1. The complexity behavior of the 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 SPL of Boolean functions for any number of variables.

7. FUTURE WORK

Future work includes minimizing the global error of the match and to develop experiments to much larger number of variables. We are in the process of matching an automated global fit for any SPL curve in order to find the complexity for any number of product terms. Investigation and modeling a mathematical relationship for other BDD characteristics (i.e. average path length, longest path length and number of path) are also considered.

8. CONCLUSION

We have discussed the idea of using BDD to study and model a relationship between the path length and the number of product terms in a Boolean function. We have introduced a mathematical model that can predict valuable information related to the SPL behavior 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 to handle evaluation time minimization by path length optimization problems prior to the design of digital circuits. Our experimental results show good correlation between the experimental results and those given by the mathematical model.

REFERENCES

[1] 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.

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

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

[4] Miczo, Alexander, *Digital Logic Testing and Simulation*, *Chapter 2: "Combinational Logic Test*," Harper and Row, New York, 1986.

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

[6] P.W.C. Prasad, A. Assi and M. Raseen, "*BDD Minimization Using Graph Parameter Permutation*", The 2004 International Conference on VLSI, 491-494, 2004.

[7] 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.

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

[9] 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, 2001.

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

[11] S. Nagayama and T. Sasao, "*Code generation for embedded systems using heterogeneous MDDs,*" the 12th workshop on Synthesis and System Integration of Mixed Information technologies (SASIMI 2003), pp. 258-264, Hiroshima, Japan, April 3-4, 2003.

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

[13] 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.

[14] M. Raseen, P.W.C. Prasad and A. Assi, "*Mathematical Model to Predict the Number of Nodes in an ROBDD*," The 47th IEEE International Midwest Symposium on Circuit and Systems (MWSCAS), Vol. III, pp.431-434, 2004.

[15] Ali Assi, M. Raseen, P.W.C. Prasad and A. Harb, "*An Efficient Mathematical Estimation of the BDDs Complexity*", Accepted for presentation in *16th IASTED International Conference on Modeling and Simulation,* pp 381-386, 2005.

[16] M. Raseen, A.Assi, P.W. C. Prasad and A. Harb, "*Effect of Boolean Min-terms on the Complexity of ROBDDs*", International Conference on Computational Intelligence (ICCI 2004), pp.454-457, 2004.

[17] F. Somenzi, "*CUDD: Decision Diagram Package*". *ftp://vlsi.colorado.edu/ pub/*, 2003.