ACCEPTED: P. W. C. Prasad, A. Beg, and A. K. Singh, "Effect of Quine-McCluskey Simplification on Boolean Space Complexity,"
in IEEE Conference on Innovative Technologies in Intelligent Systems & Industrial Applications (CITISIA 2009), Bandar Sunway, Malaysia, 2009

Effect of Quine-McCluskey Simplification on Boolean Space Complexity

P. W. Chandana Prasad, Azam Beg, Ashutosh Kumar Singh

Abstract—**The minimization of logic gates is needed to simplify the hardware design area of programmable logic arrays (PLAs) and to speed up the circuits. The VLSI designers can use minimization methods to produce high speed, inexpensive and energy-efficient integrated circuits with increased complexity. Quine-McCluskey (Q-M) is an attractive algorithm for simplifying Boolean expressions because it can handle any number of variables. This paper describes a new model for the estimation of circuit complexity, based on Quine-McCluskey simplification method. The proposed method utilizes data derived from Monte-Carlo simulations for any Boolean function with different count of variables and product term complexities. The model allows design feasibility and performance analysis prior to the circuit realization.**

I. Introduction

LOGIC level simulation is still one of the most frequently used operations in digital circuits during both the design and the test stages [1]. With the rapid increase of the amount of logic to be integrated in a single chip, there is a need for greater efforts in optimization of the design process [2]. One of the key objectives of designing digital logic circuits including the PLA and Programmable Array Logic (PAL) [3], is to keep the number of logic gates as minimum as possible, which will reduce the production cost of these systems. To simplify the complexity of a circuit, the designer must find another circuit that computes the same function as the original but does so with fewer or simpler gates. The logic design complexity is direct related to the complexity of the Boolean functions. Therefore designers have to make the Boolean function the simplest. The basic problem of logic minimization [4] is that the conversion of logic equations in the gate level netlists. Boolean function representations also have direct impact on the computation time and memory requirements in the design of digital circuits. In all these cases, the efficiency of any method depends on the complexity of the Boolean function [5]. Research on the complexity of Boolean functions in non-uniform computation models is now part of one of the most interesting and important areas in the theoretical computer science [6]-[8].

The most known simplification method called the Map Method, first proposed by Veitch [9} and slightly modified by Karnaugh [10], provides a simple straightforward procedure for minimizing Boolean functions. However, the map method is not convenient as long as the number of variables exceeds five or six. In order to minimize the Boolean functions with many variables, the other method called as Tabulation Method; this is a step-by-step procedure and was first formulated by Quine [11] and later improved by McCluskey [4], also known as the Quine-McCluskey method (Q-M).

Like the K-map, the Q-M method seeks to collect product terms by looking for entries that differ only in a single bit. The only difference from a K-map is that we do it by searching, rather than mapping. The beauty of the Q-M method is that it takes over where the K-map begins to fail. The Q-M technique is capable of minimizing logic relationships for any number of inputs. The main advantage of this method is that it can be implemented in the software in an algorithmic fashion. But the disadvantage of this method is that the computational complexity still remains high.

Rapid increase in the design complexity and the need to reduce time-to-market have resulted in a need for computer-aided design (CAD) tools that can help make important design decisions *early* in the design process.

Area complexity is one of the most important criteria that have to be taken into account while working with simplification methods. However, to be able make these decisions early, there is a need for methods to estimate the area complexity and power consumption from a design description of the circuit at a high level of abstraction [12].

Nemani, and Najm, [12] proposed an area and power estimation capability, given only a *functional* view of the design, such as when a circuit is described only by Boolean equations. In this case, no structural information is known—the lower level (gate-level or lower) description of this function is not available. Of course, a given Boolean function can be implemented in many ways, with varying power dissipation levels. They were interested in predicting the minimal area and power dissipation of the function that meets a given delay specification. In this paper [12], they use "gate-count" as a measure of complexity, mainly due to the key fact observed by Muller [13], and also because of the popularity of cell-based (or library based) design.

In an early work, Shannon [14] studied area complexity, measured in terms of the number of relay elements used in

Manuscript received on April 1, 2009.
Dr Chandana Withana is with Study Group, Charles Sturt University, Australia, Sydney, Australia (phone: +61-2-9291-9356; fax: +61-2-9283-3302; e-mail: c.withana@sga.edu.au).
Dr Azam Beg is with the College of Information Technology, United Arab Emirates University, Al-Ain, UAE (phone: +971-3-713-5561; fax: +971-3-713-767-2018; e-mail: abeg@uaeu.ac.ae).
Dr Ashutosh Kumar Singh is with the Department of Electrical/Communication Engineering and Computing, Curtin University of Technology Sarawak Campus, Malaysia. (phone: +60 85 443939; Fax: +60 85 443837 Email: ashutosh.s@curtin.edu.my

building a Boolean function (switch-count). In that paper, Shannon proved that the *asymptotic* complexity of Boolean functions was *exponential* in the number of inputs, and that for large number of inputs, *almost every* Boolean function was exponentially complex. Muller, [13] demonstrated the same result for Boolean functions implemented using logic gates (gate-count measure). A key result of his work is that a measure of complexity based on gate-count is independent of the nature of the library used for implementing the function. Several researchers have also reported results on the relationship between area complexity and entropy of a Boolean function. These include Kellerman, [15]-[18] empirically demonstrated the relation between entropy and area complexity, with area complexity measured as literal count. They showed that *randomly generated* Boolean functions have a complexity *exponential*, and proposed to use that model as an area predictor for logic circuits. However, the circuits tested were very small, typically having less than ten inputs.

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. Researchers in this area are actively involved in developing mathematical models that predict the output size of the logic circuit in order to predict the complexity of the design in terms of the time needed to optimize it and verify its logic. During the last two decades, Binary Decision Diagrams have gained great popularity as successful method for the representation of Boolean functions [19]. Over the years, the number of nodes in a BDD has been used to assess the complexity of the Boolean circuit [20]. 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.

The main objective of this paper propose a mathematical model to estimate the Boolean function complexity derived from randomly generated Boolean functions and simplified use of Q-M method. The remainder of this paper is divided as follows: Background information pertaining to Q-M simplification method is given in Section II. The proposed mathematical model and the discussions on the experimental results and comparisons are explained in the Section III. Finally, we conclude our paper with the advantages of using our model and our future developments.

II. PRELIMINARIES

Quine-McCluskey Method

Following are the important terms which will be used throughout the paper, defining again for the reader convenient and self contained.

Definitions

Literal: It is a variable or its negation $(x \text{ or } x')$

Minterm: A product of the literals where each variable appears exactly once either true or complemented form, i.e., a normal product term consisting of *n* literals for *n* variable function.

Maxterm: A sum of the literals where each variable appears exactly once either true or complemented form i.e a normal sum term consisting of *n* literals.

DNF Form (sum-of-product): The disjunctive normal form is the sum of minterms of the variables.

CNF Form (product-of-sum): Conjunctive normal form is a product-of-maxterm of the variables.

Prime Implicant: A prime implicant of a function is the product which can not be combined with another term to eliminate a variable for further simplification.

Essential Prime Implicant: Prime implicant that is able to cover an output of the function which is not covered by any combination of prime implicant called essential prime implicant.

Quine-McCluskey (Q-M) method [4], [11] minimizes a logical expression realizing a given Boolean function which is more efficient for computer algorithm, makes this more useful now even though it was introduced more than 55 years ago [21]. The method utilizes the following three basic simplification laws:

(i) $x + x' = 1$ (Complement)

(ii) $x + x = x$ (Idempotent)

(iii) $x(y + z) = xy + xz$ (Distributive)

This method is also known as tabulation method because it gives deterministic steps to check the minimum form of function based on selection of essential prime implicants using a table. Steps can be broadly categorized in three steps:

(a) Find the Prime Implicant

In this step, we replace the literals in form of 0 and 1 and generate a table. Initially, the number of rows in table is equal to the total number of minterms of the original un-simplified function. If two terms are only different in one bit like 101 and 111 i.e. one variable is appearing in both form (variable and its negation), then we can use complement law. Iteratively, we compare all term and generate the prime implicant.

(b) Find the Essential Prime Implicant

Using prime implicants from above step, we generate the table to find essential prime implicants. Note that some prime implicants can be redundant and may be omitted, but if they appear only once, they cannot be omitted and provide prime implicant.

(c) Find Other Prime Implicant

It is not necessary that essential prime implicants cover all the minterms. In that case, we consider other prime implicant to make sure that all minterms has been covered. Instead of

using trial and error to consider these prime implicants we can use Petrick Method [22].

In general, Q-M method provides better method for the function simplification than K-map, but still is an NP-hard problem, and it becomes impractical for large input sizes due to exponential complexity.

III. Proposed Method

The complexity of a digital circuit mainly depends on the number of literals represented by the Boolean function. It can be easily observed that if the number of product terms in a Boolean function increases, the complexity of the Boolean function increases, and hence the complexity of its logic circuit increases. If the number of product terms in the Boolean function increases beyond a particular limit, the product terms simplify among themselves and the complexity of the function starts to decrease. Consequently, the circuit complexity decreases [23].

We carried out a large set of experiments on Boolean functions derived from the Monte Carlo data of randomly generated Boolean functions to analyze the exact Boolean function complexity variation, i.e., the relation between the number of product terms and the number of nodes for any number of variables. For each variable count n between 1 and 15 inclusive and for each term count between 1 and $2n$-1 (non-repeating product terms), 100 SOP terms were randomly generated and the average number of literals in the output function were determined after Q-M simplification process. This process was repeated until the average size of the Boolean function complexities (i.e. number of literals) became 1. Then the graphs for Boolean function complexities (Fig. 1) were plotted against the product term (min-term) count for number of variables 1 to 15.

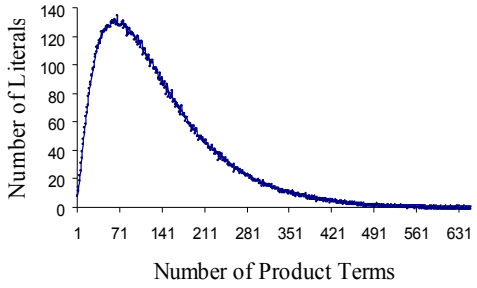

Fig. 1. Complexity distribution for 10 variables

Fig. 1 illustrates the Boolean function complexity relationship, i.e., the relation between the average number of literals and the number of product terms for randomly generated Boolean functions with 10 variables. The graph indicates that the Boolean function complexity (i.e. number of literals) increases as the number of product terms increases. This is clear from the rising edge of the curve shown in Fig.1. At the end of the rising edge, the number of literals reaches a maximum (~130 in this case). This peak indicates the maximum number of literals that any randomly generated Boolean function with 10 variables can produce independently of the number of product terms. The peak also specifies the number of product terms (critical limit) of a Boolean function that leads to the maximum number of literals. For 10 variables, the Boolean function having ~60 product terms leads to a maximum number of literals (130). If the number of product terms increases above the critical limit, as expected, the product terms starts to simplify and the complexity of the Boolean function decreases. The graph shown in Fig. 1 indicates that, as the number of product terms increases further, the complexity of the Boolean function decreases at a slower rate while it ultimately reaches 1. With 10 variables, the Boolean function complexity reduces to 1 when the number of product terms reaches nearly 560.

Mathematical Model

Analysis of the graph shown in Fig. 1 reveals that the Boolean function complexity can be modeled mathematically by the following equation:

$$N = \alpha \cdot t^{\beta} \cdot e^{(-t \cdot \gamma)} + 1 \qquad (1)$$

where,

N is the number of literals,

t is the number of non-repeating product terms in the Boolean function,

α, β and γ are three constants depend on the number of variables.

It can be inferred that the following equation follows the *Weibull* function pattern [24]. For 10 variables the values of the constants are α = 7.7, β =0.904 and γ = 0.0145. Fig. 2 depicts the experimental results obtained by Q-M simplification of randomly generated Boolean functions and the theoretical results obtained using equation (1).

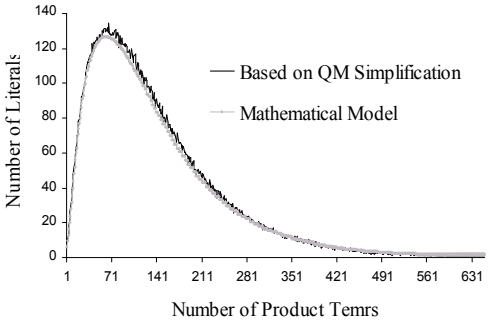

Fig. 2. Results comparison for Boolean space complexity of 10 variable functions

Fig. 2 indicates that the proposed model represented by equation (1) provides a very good approximation of the Boolean space complexity in terms of literals. The same work has been repeated for Boolean functions with 2 to 15 variables. The results for 10 variables are shown in Fig. 2.

Fig. 3 and Fig. 4 illustrate experimental and mathematical model results for 13 and 15 respectively.

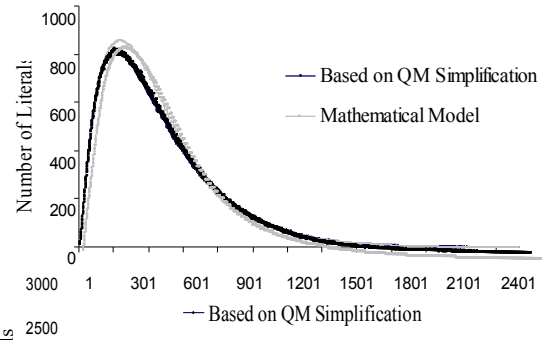

Fig. 2. Complexit

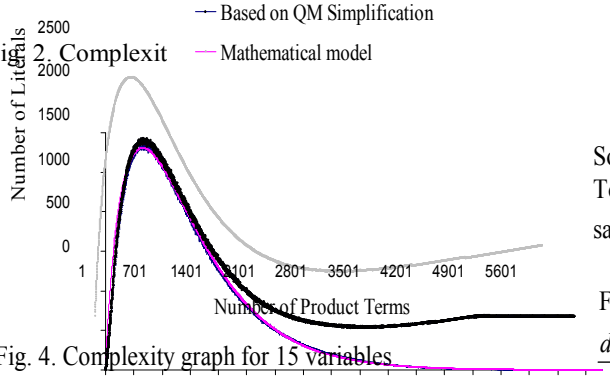

Fig. 4. Complexity graph for 15 variables

Variations of Constants α, β and γ

The values of α, β and γ that were obtained experimentally, for Boolean functions with 2 to 15 variables. Using curve fitting techniques, the variations of α, β and γ were mathematically modeled and are represented by the following equations (2), (3) and (4).

$$\alpha = 0.5 + 0.28 \cdot e^{(v/3)}, \tag{2}$$

$$\beta = 1 + 67.2 \cdot e^{(-1.3 \cdot v)}, \text{ and} \tag{3}$$

$$\gamma = 18 \cdot e^{(-v)}, \tag{4}$$

where, v is the number of variables.

Maximum Complexity

The maximum number of literals (N_{max}) for a given number of variables can be determined by applying first and second order maxima and minima theorems to equation (2). The value of t when N is max ($N = N_{max}$) can be calculated by solving the maxima theorem equations

$$\frac{d(N)}{d(t)} = 0 \quad \text{and} \quad \frac{d^2(N)}{d(t)^2} < 0$$

From (2) we have,

$$\frac{d(N)}{d(t)} = \frac{d[\mu \cdot \alpha \cdot t^{\beta} \cdot e^{(-t \cdot \gamma)} + 1]}{d(t)}$$

$$\frac{d(N)}{d(t)} = \alpha \cdot \mu \frac{d[t^{\beta} \cdot e^{(-t \cdot \gamma)}]}{d(t)} \tag{6}$$

$$\frac{d(N)}{d(t)} = \alpha \cdot \mu \cdot \{t^{\beta} \cdot e^{(-t \cdot \gamma)} \cdot (-\gamma) + e^{(-t \cdot \gamma)} \cdot \beta \cdot t^{(\beta-1)}\} \tag{7}$$

Solving $\frac{d(N)}{d(t)} = 0 \Rightarrow$

$$\alpha \cdot \mu \cdot \{-\gamma \cdot t^{\beta} \cdot e^{(-t \cdot \gamma)} + \beta \cdot e^{(-t \cdot \gamma)} \cdot t^{(\beta-1)}\} = 0 \tag{8}$$

$$\alpha \cdot \mu \cdot e^{(-t \cdot \gamma)} \cdot t^{(\beta-1)} \cdot \{\beta - \gamma \cdot t\} = 0 \tag{9}$$

Since $\alpha \neq 0$ and $\mu \neq 0$

$$e^{(-t \cdot \gamma)} = 0 \text{ or } t^{(\beta-1)} = 0 \text{ or } \beta - \gamma \cdot t = 0$$

$$\Rightarrow t = \infty \text{ or } t = 0 \text{ or } t = \beta / \gamma \tag{10}$$

So the critical points (maxima and minima) are ∞, 0 and β/γ. To be a maxima, these critical points must satisfy $d^2(N)/d(t)^2 < 0$.

From (7) we can write,

$$\frac{d^2(N)}{d(t)^2} = \alpha \cdot \mu \cdot \beta \cdot \frac{d[e^{(-t \cdot \gamma)} \cdot t^{(\beta-1)}]}{d(t)} - \alpha \cdot \mu \cdot \gamma \cdot \frac{d[t^{\beta} \cdot e^{(-t \cdot \gamma)}]}{d(t)} \tag{11}$$

$$\frac{d^2(N)}{d(t)^2} = \alpha \cdot \mu \cdot \beta \cdot \{e^{-t \cdot \gamma} \cdot (\beta - 1) \cdot t^{(\beta-2)} + t^{(\beta-1)} \cdot e^{-t \cdot \gamma}(\gamma)\}$$
$$- \alpha \cdot \mu \cdot \gamma \cdot \{-\gamma \cdot t^{\beta} \cdot e^{-t \cdot \gamma} + e^{-t \cdot \gamma} \cdot \beta \cdot t^{(\beta-1)}\} \tag{12}$$

$$\frac{d^2(N)}{d(t)^2} = \alpha \cdot \mu \cdot \beta \cdot e^{-t \cdot \gamma} \cdot t^{(\beta-2)} \{(\beta - 1) - \gamma \cdot t\}$$
$$- \alpha \cdot \mu \cdot \gamma \cdot e^{-t \cdot \gamma} \cdot t^{(\beta-2)} \{-\gamma \cdot t^2 + \beta \cdot t\} \tag{13}$$

$$\frac{d^2(N)}{d(t)^2} = \alpha \cdot \mu \cdot e^{-t \cdot \gamma} \cdot t^{(\beta-2)} \{\beta^2 - \beta - \beta \cdot \gamma \cdot t$$
$$+ \gamma^2 \cdot t^2 - \beta \cdot \gamma \cdot t\} \tag{14}$$

$$\frac{d^2(N)}{d(t)^2} = \alpha \cdot \mu \cdot e^{-t \cdot \gamma} \cdot t^{(\beta-2)} \{\beta^2 - \beta - 2\beta \cdot \gamma \cdot t + \gamma^2 \cdot t^2\} \tag{15}$$

For $t = \beta/\gamma$ we have

$$\left.\frac{d^2(N)}{d(t)^2}\right|_{(t=\beta/\gamma)} = \alpha \cdot \mu \cdot e^{-(\beta/\gamma) \cdot \gamma} \cdot (\beta/\gamma)^{(\beta-2)}$$
$$\{\beta^2 - \beta - 2\beta \cdot \gamma \cdot (\beta/\gamma) + \gamma^2 \cdot (\beta/\gamma)^2\}$$

$$= \alpha \cdot \mu \cdot e^{-\beta} \cdot (\beta/\gamma)^{(\beta-2)} \{\beta^2 - \beta - 2\beta + \beta^2\}$$

$$= -\alpha \cdot \mu \cdot \beta \cdot e^{-\beta} \cdot \left(\frac{\beta}{\gamma}\right)^{(\beta-2)} \quad (16)$$

Since α, μ, β and γ are positive $= -\alpha \cdot \mu \cdot \beta \cdot e^{-\beta} \cdot \left(\frac{\beta}{\gamma}\right)^{(\beta-2)}$ is negative, hence the critical point $t = \beta/\gamma$ is a *maxima*. Thus the number of nodes will be maximum when the number of product terms is $t = \beta/\gamma$, as given in equation (10).

The maximum number of nodes ($N_{\max}$) can be found by substituting β/γ for Number of Product Terms (NPT) in (2). The maximum number of literals ($NN_{\max}$) is given by:

$$N_{\max} = \mu \cdot \alpha \cdot \left[\frac{\beta}{\gamma}\right]^{\beta} \cdot e^{\left(-\beta/\gamma \cdot \gamma\right)} + 1$$

$$N_{\max} = \frac{\mu \cdot \alpha \cdot \beta^{\beta} \cdot e^{-\beta}}{\gamma^{\beta}} + 1$$

$$N_{\max} = (\mu \cdot \alpha) \cdot \left(\frac{\beta}{\gamma \cdot e}\right)^{\beta} + 1 \quad (17)$$

Thus, the variation of Boolean space complexity is defined by the equation (2), number of product terms for maximum complexity by the equation (10) and the maximum complexity by the equation (17). The values of constants α, β and γ can be obtained from equations (3), (4) and (5).

From equation (10), for 10 variables with symmetric sift reordering method, the maximum number of literlas (N_{max})will occur when the number of product terms ($t_{(N=NNmax)}$) is:

$$t|_{(N=N_{\max})} = \beta/\gamma = 0.904/0.0145 = 62.34 \cong 62$$

From equation (17), for 10 variables, the number of literlas will be maximum when t = 62. The maximum number of literlas ($N_{\max}$) will be

$$N_{\max} = (1 \cdot 7.7) \cdot \left(\frac{0.904}{0.0145 \cdot 2.7182}\right)^{0.904} + 1$$

$$= 131.72 \cong 132 \text{ nodes}$$

The calculated values of $t_{(when\ N=Nmax)}$ and N_{max} (62 and 132 respectively for 10 variables) match with the experimental values in Fig. 2 (62 and 132). This demonstrates the accuracy of our developed model.

Advantages

Using the mathematical fitting approximation represented by equation (1), we can accurately predict the followings:

a) The complexity of the Boolean space (i.e., number of literals) given the number of product terms
b) The number of product terms of a Boolean function for which Boolean space complexity will be maximum (i.e,, maximum number of literals). (eg: V=10, t =62 gives $N_{\max}$)
c) The maximum Boolean space complexity for given number of variables in a Boolean function. (e.g., V=10, $N_{\max}$=132)
d) Boolean functions that will have equal complexity, even though their corresponding Boolean functions have different number of literals, different number of product terms. (e.g.., equal complexity (V, t) = (10,29) = (11,21) = (11,295)

Most of the current VLSI CAD tools use techniques such as BDDs to find the circuit complexity; they use of number of nodes and other parameters. Building a complete BDD and counting its nodes can require a lot more time than predicting it using the proposed model. Estimating the Boolean space complexity using the proposed mathematical model is expected to reduce the time complexity required to develop and execute those applications.

IV. CONCLUSION

In this work, we address the problem of predicting the area complexity of any Boolean function knowing the number of product terms and the number of variables. We analyzed the behavior of Q-M simplification for different number of product terms and introduced a mathematical model to predict the Boolean space complexity.

Our experimental results show good correlation between the theoretical results and the results predicted by the mathematical model. The proposed model can greatly reduce the time complexity for applications that use Boolean functions as the input function. Our future work will be mainly concentrated on having wider range of variables to check that the fitting is correct, to verify the the proposed method with benchmark circuits, to do a comparison with other applications that perform the circuit complexity calculations.

REFERENCES

[1] M. Alexander, "Digital Logic Testing and Simulation," Chapter 2: Combinational Logic Test. Harper and Row, New York, 1996.

[2] M. Thornton, and V.S.S. Nair, "Iterative Combinational Logic Synthesis Techniques using Spectral Data," *Technical report*, Southern Methodist University, 1992.

[3] R. Czrewin, D. Kania, and J. Kulisz, "FSMs state encoding targeting at logic level minimization," Bulletin of the Polish Academy of Sciences, Vol. 54 (4), pp 479- 487, 2006.

[4] E. J. McCluskey, "Minimization of Boolean Functions," *Bell System Technical Journal*, vol. 35, no. 5, pp. 1417-1444, 1956.

[5] 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, 1997..

[6] I. Wegener, "The Complexity of Boolean Functions," *Wiley and Sons*. Inc, 1987.

[7] C. Meinel, and A. Slobodova, "On the Complexity of Constructing Optimal Ordered Binary Decision Diagrams," *Proceedings of the 19th International Symposium on Mathematical Foundation of Computer Science*, pp.515-524, 1994.

[8] S. Tani, K. Hamaguchi, and S. Yajima, "The Complexity of the Optimal Variable Ordering Problems of a Shared Binary Decision Diagram," *IEICE Transaction on Information and Systems*, Vol. 4, pp. 271-281, 1996.

[9] E.W. Veith, 'A Chart Method for Simplifiying Truth Functions.' , Proc. of the ACM, pp. 127-133, 1952.

[10] M. Karnaugh, 'A Map Method for Synthesis of Combinational Logic Circuits', Trans. AIEE, Comm. And Electronics, Vol.72(1), pp. 593-99, 1953.

[11] W.V. Quine,' The Problem of Simplifiying Truth Functions', Am. Math. Montly, Vol.59(8), pp.521-31,1952.

[12] M.,Nemani, and F. N. Najm, "High-Level Area and Power Estimation for VLSI Circuits,".*IEEE Transaction on Computer Aided Design of Integrated Circuits and Systems*, 18 (6), pp. 697-713, 1999.

[13] D. E.Muller, "Complexity in Electronic Switching Circuits,".*IRE Trans. on Electronic Computers*, Vol. 5, pp. 15-19, 1956.

[14] C. E.Shannon, "The Synthesis of Two-Terminal Switching Circuits,". Bell System Technical Journal, 28 (1),. Pp. 59-98, 1949.

[15] E. Kellerman, "A Formula for Logical Network Cost," *IEEE Transaction on Computers*, 17(9), pp. 881-884, 1968.

[16] N. Pippenger, "Information Theory and the Complexity of Boolean Functions," *Mathematical Systems Theory*, Vol. 10, pp. 129-167, 1977.

[17] R. W.Cook, and M. J.Flynn, "Logical network cost and entropy," *IEEE Trans. on Computers*, Vol. 22(9), pp.823-826, 1973.

[18] K. T.Cheng, and V.Agrawal, "An Entropy Measure for the Complexity of Multi-output Boolean Functions, ".*Proceedings of the Design Automation Conference*, pp. 302-305, 1990.

[19] R. E. Bryant, "On the Complexity of VLSI Implementations and Graph Representations of Boolean Functions with Application to Integer Multiplication," *IEEE Transaction on Computers*, Vol. 40, 203–213, 1991.

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

[21] Milos Seda, "Heuristic Set-Covering-Based Postprocessing for Improving the Quine-McClusky Method," *Proceeding of World Academy of Science, Engineering and Technology*, vol. 23, pp. 256-260, Aug. 2007.

[22] S.K. Petrick, "On the Minimization of Boolean Functions," in Proceedings of the International Conference Information Processing, pp. 422-423, 1959.

[23] L. Franco, and M. Anthony, "On a Generalization Complexity Measure for Boolean Functions," *Proceedings of the 2004 IEEE International Joint Conference on Neural Networks*, pp.973-978, 2004.

[24] http://www.weibull.com/LifeDataWeb/characteristics_of_the_weibull_distribution.htm