ACCEPTED: A. Beg, P. Chandanna, and A. Assi, "Modeling the Behavior of BDD Complexity Using Neural Networks," in 4th IASTED International Conference on Circuits, Signals, and Systems, San Francisco, CA, USA, 2006.

Modeling the Behavior of BDD Complexity Using Neural Networks

Author 1, 2, Dept A, XYZ University
Author 3, Dept B, XYZ University

Abstract - **This research work proposes a new method to estimate the Binary Decision Diagram (BDD) complexity. A feed-forward back-propagation neural network (NN) is used and a large number of randomly generated single output Boolean functions have been considered. Experimental results show good correlation between the theoretical results and those predicted by the NN model (NNM), which will give insights to the complexity of Very Large Scale Integration (VLSI)/Computer Aided Design (CAD) designs. This model demonstrates the ability of NNs to provide reliable classification of BDD complexity.**

I. INTRODUCTION

The efficient representation and manipulation of Boolean functions is important for many algorithms in a wide variety of applications in digital circuits. In particular, many problems in Very Large Scale Integration (VLSI) and Computer Aided Design (CAD) for digital circuits can be expressed as a sequence of operations performed over a set of Boolean functions [1].

With the rapid increase of the amount of circuitries on a single chip, there is a need for greater optimization and efficiency in the design process [2]. Boolean function representation has direct influence on the computation time and space complexity of digital circuits.

During the last two decades, BDDs have gained great popularity as efficient method for representing Boolean functions [3]. BDD in general is a direct acyclic graph representation of Boolean functions proposed by Akers [4] and further generalized by Bryant [5]. In many applications, the efficiency of BDD representations is determined by the size of the BDD defined as the number of nodes in the BDD for a given Boolean function. The success of this technique has attracted researchers in the area of VLSI CAD systems [3]. Evaluation of the space complexity of Boolean functions can be performed by determining the area complexity of the BDD. It will be useful to have an estimation of the BDD complexity prior to making decisions on the feasibility of the design. There have been a lot of research works [6], [7], [8] done on the estimation of combinational and sequential circuit parameters from the exact Boolean function describing the circuit. Mathematical models to predict the complexity of Boolean functions and XOR/XNOR min-terms were introduced in papers [9], [10], [11].

NNs have proven their usefulness in the area of pattern recognition and prediction applications [12]. Apart from this lot of research has been done on the computational properties of NNs [13], [14]. The measure of efficiency of the circuit have been addressed in relation with the area of circuit implementation [15], where the complexity of Boolean functions is analyzed in terms of their implementation using different kind of circuits, from those with simple SOP, to feed forward neural network with threshold functions. In recent times some research has covered the topic of Boolean function complexity using NN learning process. For example, generalization ability of NNs was presented in [18], [19] but the authors only considered Boolean functions of 4 and 8 variables.

The main objective of this paper is to introduce a BDD complexity estimation methodology based on NNs for a wide range of variables. The resulting model will enable the design feasibility and performance to be analyzed without building its complete BDD. This model has produced competitive results against the mathematical prediction of the BDD complexity. In the second section, we review the previous works done by the same authors on the estimation of BDD complexity. The proposed NNM for the estimation of BDD complexity is explained in the third, fourth and fifth sections. Finally in section six, we conclude our paper.

II. PREVIOUS WORK

In this section we will briefly describe the results achieved by the authors for the estimation of BDD complexity.

A. *Relation between the Size of a Boolean function and the BDD Complexity*

The complexity of the ROBDD mainly depends on the number of nodes represented by the BDD. Experiments were done in [9], [10] to analyze the complexity variation in BDDs. The experimental and equation graph (Fig. 1) shows that the complexity of the BDD can be modeled mathematically by (1).

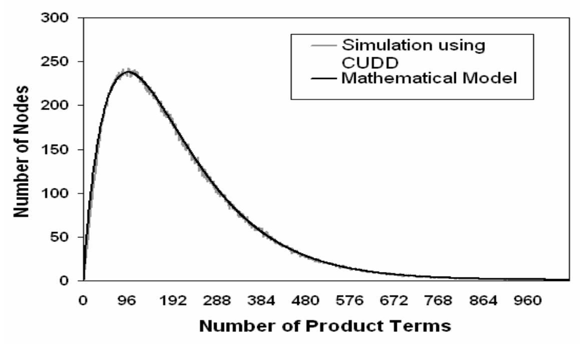

Fig. 1. Simulation / Mathematical BDD Complexity for 11 Variables

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

where NN is the number of nodes that represents the complexity of the BDD, NPT is the number of non-repeating product terms in the Boolean function, α, β and γ are three constants.

III. Analysis of BDD Complexity

For each variable count *n* between 1 and 14 inclusive and for each term count between 1 and *2n-1*, 100 SOP terms were randomly generated and the Colorado University Decision Diagram (CUDD) package [18] was used to determine the BDD complexity. This process was repeated until the BDD complexities (i.e. number of nodes) became 1. Then the experimental graphs for BDD complexity were plotted against the product term count for each number of variables (Fig. 2).

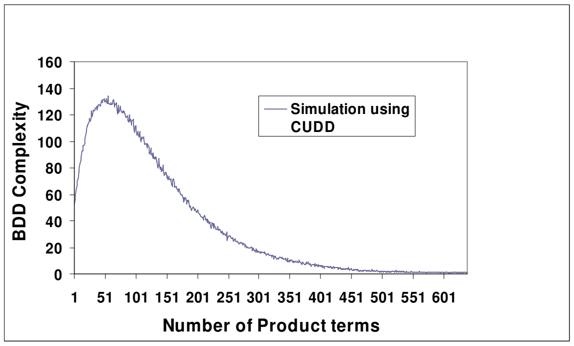

Fig. 2. Simulation results for Boolean complexity for 10 variables

The above graphs indicate that the BDD complexity in general increases as the number of product terms increases. This is clear from the rising edge of the curve. At the end of the rising edge, the graph reaches a maximum complexity. Apart from that the peak also specifies the number of product terms (critical limit) of a Boolean function that leads to the maximum BDD complexity for any Boolean function with 10 variables. If the number of product terms increases above the critical limit, as expected, the product terms starts to simplify and the BDD complexity will reduce. The BDD Complexity graph shown in Fig. 2 indicates that as the number of product terms increases the complexity of the BDD decreases at a slower rate and ultimately reaches 1 node.

IV. Neural Network Modeling Methodology

This section covers the definition and implementation of the Neural Network Model (NNM) for modeling the BDD complexity.

A. *Model Definition and Data Collection*

The purpose of the NNM in this research was to model the complexity of BDDs. Inputs to the NNMs were the number of variables, and the NPT (min-terms) (Fig. 3). For the NNM in this paper, the training and validation data sets were obtained by the experiment done in section III.

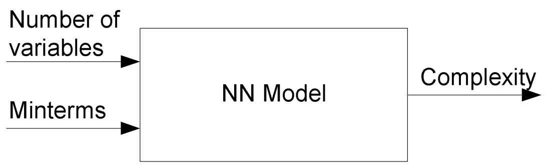

Fig. 3. Inputs and output of the NNM

B. *Data Pre-Processing*

Pre-processing the training and validations sets takes a considerable amount of resources for a practical and reliably functioning NNM. In our research, the first data pre-processing step was to transform the data set in such a way that inputs have equitable distribution of importance. In other words, the larger absolute values of an input should not have more influence than the inputs with smaller magnitudes. The need of such equitable distribution can be explained with the set of figures shown below. Fig. 4 shows the raw (original) data for 2 to 14 variables.

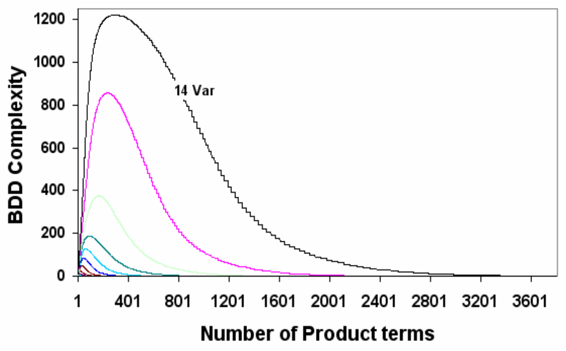

Fig. 4. Raw (untransformed) data

Notice that the plots for 2- to 9-variables are hardly visible when all variables are plotted on the same scale. If the data were presented to the NN for training in this case, only 10- to 14- variable cases could be learnt by the NN and 2- to 9-

variables values could be ignored. So in order to provide similar importance to all variable values (2 to 14), we performed a logarithmic transformation of the product terms (min-terms) and complexity (number of nodes) inputs. The resulting data is plotted in Fig. 5. As we can see now all different plots (from 2 to 14 variables) are in similar ranges and make it easier for NN to learn them.

In order to 'use' or 'run' a trained NN, de-normalization and de-transformation has to be done to restore the predicted outputs to the original ranges. Steps employed in 'training' and 'running' the network are summarized here:

1. Steps for Training the NNM:

a) Take logarithm of actual values of the inputs and output

b) Train the NN with values from step (a)

2. Steps for Using/Running the NNM:

a) Take logarithm of the actual values of the input

b) Present the values from step (a) to the NNM

c) Apply anti-logarithm to the output of the NNM to get the actual result

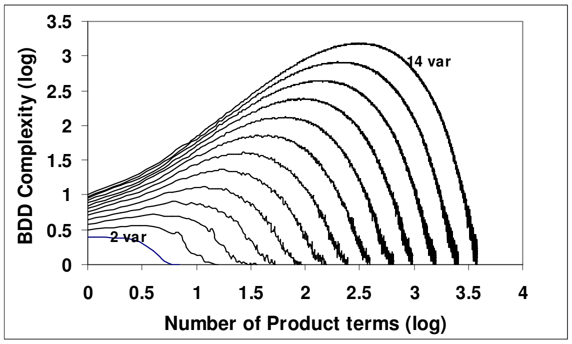

Fig. 5. Log-scaled (transformed) data

D. NN Training and Testing for BDD Complexity

We used an NN-modeling software package called Brain Maker (version 3.75 for MS-Windows [19] to create and test our NNMs. Brain Maker's back-propagation NNs were fully connected, meaning all inputs were connected to all hidden neurons, and all hidden neurons were connected to the outputs. The activation function for the hidden and output layers was a sigmoid function. The difference between the network's actual output and the desired output was treated as the error to be minimized.

We had acquired a total of 19044 data sets during our simulations of Boolean functions (already described in Section 3). From the complete data set, we used 17313 sets (~90% of 19044) for training the NNM and the remaining 1731 sets (~10% of 19044) for validating the NNM. The validation data set allowed us to verify that the NNM was able to make predictions accurately with the data not 'seen' during the training phase. We per-formed the training cycles (also called epochs) until 95% of the training data sets were learnt with less than 5% Mean Squared Error (MSE) compared to 25% error in [16], [17].

A general rule is that as the number of hidden layers increases, the prediction performance goes up, but only up to a certain point, after which the NNM performance starts to deteriorate [19]. To find the optimum topologies for our NNMs, we experimented with up to 3 hidden layers; each layer consisted on a different number of neurons. The details of some of our NNMs experiments are listed in Table 1. We chose a 5-layer NNM (#8 in the table) with 5 neurons in each of its hidden layers. This configuration provided nearly the same training accuracy as its much larger 3-layer counterparts (#9 and #10).

TABLE 1: CONFIGURATION & TRAINING STATISTICS *

CONFIGURATION						TRAINING STATISTICS			
No.	Input Layer Neurons	Hidden Layer 1 Neurons	Hidden Layer 2 Neurons	Hidden Layer 3 Neurons	Output Layer Neurons	Facts Learnt	Facts Not Learnt	% Facts Learnt	Epochs
1	2	10			1	12524	4789	72.3%	1047
2	2	20			1	16208	1105	93.6%	623
3	2	25			1	16059	1254	92.8%	745
4	2	30			1	15844	1469	91.5%	630
5	2	5	5		1	16889	424	97.6%	681
6	2	7	7		1	15261	2052	88.1%	2133
7	2	20	20		1	16987	326	98.1%	100
8	**2**	**5**	**5**	**5**	**1**	**17028**	**285**	**98.4%**	**98**
9	2	5	10	5	1	17049	264	98.5%	24
10	2	20	20	20	1	17079	234	98.6%	17

• Brain Maker training parameters: Training tolerance = 0.05; testing tolerance = 0.05; learning rate adjustment type = heuristic. The training accuracy is a function of initial weights which are automatically assigned by Brain Maker. 'Facts Learnt' means the number of training facts that were learnt by the NNM with <5% error; and 'Facts Not Learnt' means the training facts that did not fulfill the <5% error criteria [17].

E. NN Modeling Results and Analysis

Although NNMs are quite interpolative in nature, they tend to be less prone to noisy data than analytical or statistical models [15]. We used an arbitrary set of values for number-of-variables and NPT and used the NNM to predict the number of nodes (BDD complexity).

Fig. 6 indicates the comparison for experimental results and NNM predictions of BDD complexity for 10 variables. It can be inferred that the NNM result provides a very good approximation of the BDD complexity.

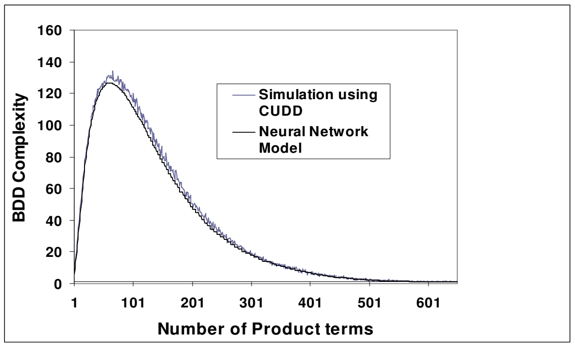

Fig. 6. Complexity analyses of Simulation / Neural network models for 10 variables

The same work has been repeated for Boolean functions with 2 to 15 variables. Fig. 7, 8 and 9 illustrate experimental and predicted NNM results for 7, 12 and 14 variables, respectively.

Fig. 10 shows the efficiency of the proposed NNM, which produces very close fit as the mathematical model [10] for the prediction of BDD complexity. It can be inferred that the NNM was able to match the experimental curve with minimum error for most of the product terms.

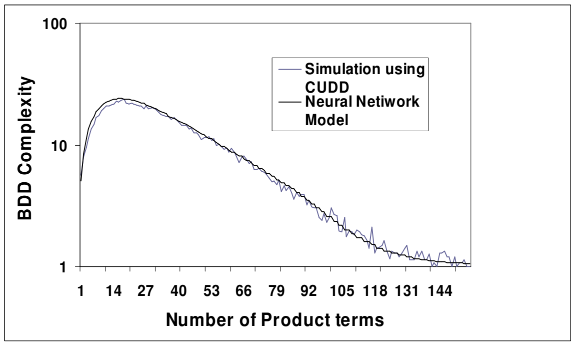

Fig. 7. Complexity analyses of Simulation / Neural network models for 7 variables

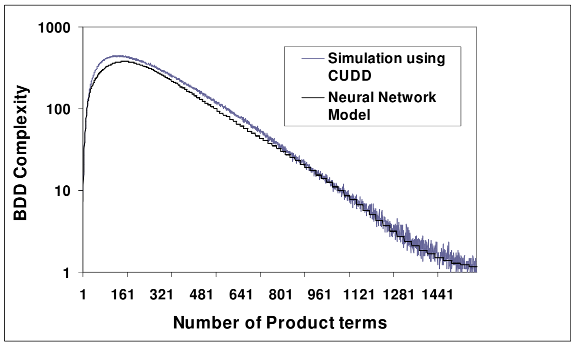

Fig. 8. Complexity analyses of Simulation/ Neural network models for 12 variables

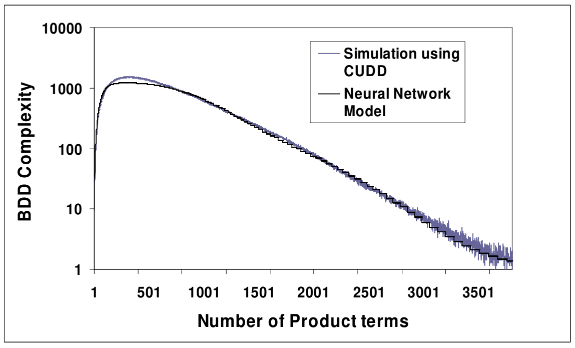

Fig. 9. Complexity analyses of Simulation / Neural network models for 14 variables

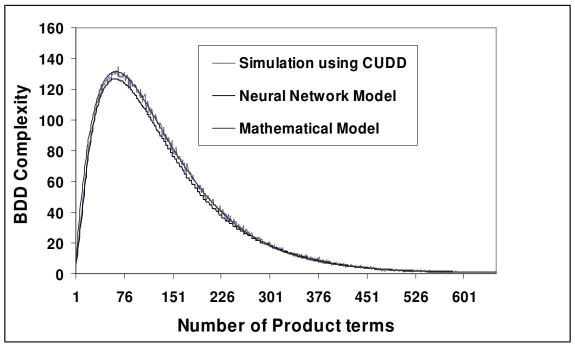

Fig. 10. Comparison with mathematical model

V. CONCLUSION

In this work, we have proposed a new BDD complexity prediction methodology based on NN as another alternative to the CUDD simulation and the mathematical models presented by the same authors. The advantages of the proposed approach are: (1) NNM is relatively more robust and fast as compared to other methods and (2) NNM is a single integrated model for different number of variables and number of product terms. The results show the applicability of NNs to the area of BDD complexity. We see a close match for the CUDD simulation with minimum errors for the calculation of the BDD complexity. The NNM was capable of providing useful clues about the complexity of the final design, which will leads to a great reduction in time complexity for digital circuit's designs. Future work will extend to the NNM for wider range of variables to verify the proposed method with real benchmark circuits.

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

[3] S. Minato. "Binary Decision diagrams and Applications for VLSICAD," Kluwer Academic Publishers, Dordrecht, 1995.

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

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

[6] P. E. Dunne, and W. van der Hoeke "Representation and Complexity in Boolean Games," proc. 9th European Conference on Logics in Artificial Intelligence, LNCS 3229, Springer-Verlag, 2004,pp. 347-35.

[7] N. Ramalingam, S. Bhanja "Causal Probabilistic Input Dependency Learning for Switching model in VLSI Circuits," proceedings of ACM Great Lakes Symposium on VLSI, 2005, pp. 112-115.

[8] S. Bhanja, K. Lingasubramanian and N. Ranganathan "Estimation of Switching Activity in Sequential Circuits using Dynamic Bayesian Networks," proceedings of VLSI Design 2005, 2005, pp. 586-591.

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

[10] M. Raseen, P.W.C. Prasad, and A. Assi "An Efficient Estimation of the ROBDD's Complexity," accepted for Publication in Integration - the VLSI journal, Elsevier Publication, May 2005.

[11] P.W. C. Prasad , M. Raseen and S. M. N. A. Senanayake "XOR/XNOR Functional Behavior on ROBDD Representation," Proc. of 14th IASTED Inter. Conf. on Applied Simulation and Modeling, 2005, pp. 115-119.

[12] M., Caudill "AI Expert: Neural Network Primer,". Miller Freeman Publications, 1990.

[13] K. Y. Siu, V. P. Roychowdhury and T. Kailath "Discrete Neural Computation – A theoretical Foundation," Prentice Hall, 1995.

[14] I. Wegener "The Complexity of Boolean functions," Wiley and Sons. Inc., 1987.

[15] I. Parberry "Circuit Complexity and Neural Networks," MIT Press (1994).

[16] L. Franco, "Role of function complexity and network size in the generalization ability of feed-forward networks", Lecture Notes in Computer Science, v 3512, Computational Intelligence and Bioinspired Systems: 8th International Workshop on Artificial Neural Networks, IWANN 2005, Proceedings, 2005. p 1-8.

[17] L. Franco, M. Anthony, "On a generalization complexity measure for Boolean functions", IEEE Conference on Neural Networks, Proceedings, v 2, 2004 IEEE International Joint Conference on Neural Networks – Proceedings, 2004, p 973-978.

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

[19] "Brain Maker – User's Guide and Reference Manual," 7th ed., California Scientific Software Press, Jun. 1998.