ACCEPTED: P. W. C. Prasad and A. Beg, "A Methodology for Evaluation Time Approximation,"
in 50th IEEE International Midwest Symposium on Circuits and Systems (MWSCAS/NEWCAS'07), Montreal, Canada, 2007, pp. 776-778.

A Methodology for Evaluation Time Approximation

Prasad P. W. C.
College of Information Technology
United Arab Emirates University
Al-Ain, UAE
prasadc@uaeu.ac.ae

Azam Beg
College of Information Technology
United Arab Emirates University
Al-Ain, UAE
abeg@uaeu.ac.ae

***Abstract*—This paper describes a feed-forward neural network model (FFNNM) for complexity prediction of path related objective functions, mainly average path length (APL) of an arbitrary Boolean function (BF). The proposed model is determined by neural training process of evaluation time derived from the Monte Carlo data of randomly generated BFs. Experimental results show a good correlation between the ISCAS benchmark circuits and those predicted by the FFNNM. This model is capable of providing an estimation of the performance of a circuit prior to its final implementation.**

I. Introduction

One of the most important functions of CAD tools is to provide robust and efficient data structures to represent BFs as well as fast algorithms to manipulate these data structures [1], [2]. BDD was introduced by Akers [3] in 1978 as a data structure for efficient representation and manipulation of BFs. Over the years, the number of nodes in a BDD became a major concern since it is proportional to the complexity of the Boolean circuits [4]. However, the number of nodes is not directly related to the evaluation time for a BF. Rather the evaluation time is directly related to the total expected path length of a BDD [5]-[7]. The minimization of average evaluation time is very useful, in embedded system using *real-time operating system* (RTOS). Building an actual BDD model needs time for implementation, verification and testing [8]. So 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 done on the estimation of combinational and sequential circuit parameters from the exact BF describing the circuit in which BF plays a major role [9]-[11]. In this paper, we focus on the APL of a BDD, which is the sum of the path lengths over all assignments of values to the variables divided by the number of assignments.

Neural networks (NNs) have proven their usefulness in the area of pattern recognition, prediction applications and for their computational properties [12], [13]. The measure of efficiency of the circuits has been addressed in relation with the area of circuit implementation, where the complexity of BFs is analyzed in terms of their implementation using different kind of circuits, from those with simple SOP to FFNNs. The main objective of this paper is to extend the work done by the same authors on the BDD complexity to estimate the complexity of APL. The remaining of this paper is divided as follows: Section two provides the proposed FFNNM for APL complexity prediction. The proposed model validation for the simulation and actual results for IASCAS benchmark circuits are given in section three. Finally, in section four we conclude this research work with a summary of our future developments.

II. Feed-Forward Neural Network Model for APL Prediction

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 CUDD package [14] was used to determine the APL. This process was repeated until the average size of the APL complexities (i.e. number of nodes) became 1. Then the graphs for APL complexities (Figure 1) were plotted against the product term (min-term) count for number of variables 1 to 14.

If the data were presented to the NN for training in this case, only 10 to 14 variable cases may be learnt by the NN and 2 to 9 variable values may be ignored. So in order to provide

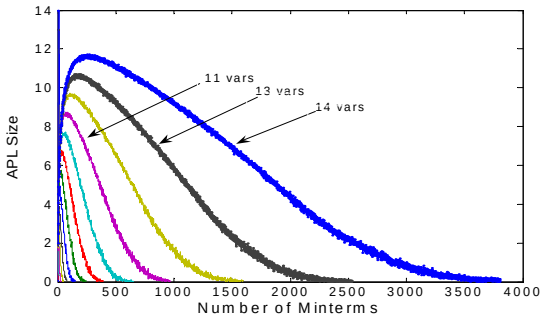

Figure 1. APL complexity variation (from simulations) for different variable values

ACCEPTED: P. W. C. Prasad and A. Beg, "A Methodology for Evaluation Time Approximation,"
in 50th IEEE International Midwest Symposium on Circuits and Systems (MWSCAS/NEWCAS'07), Montreal, Canada, 2007, pp. 776-778.

similar importance to all variable values (2 to 14), we considered a logarithmic transformation of the product terms.

We used an NN modeling software package called Brain-Maker version 3.75 [15] to model the APL behavior. Our experiments involved different number of neurons in the single hidden layer (some of which are listed in Table 1).

TABLE 1. Configuration & training statistics for APL-complexity NNMs.

Neurons in the single hidden layer	Training Epochs	Training time (min:sec)	Training accuracy %	Validation accuracy %
2	293	3:31	56.2	51.5
3	266	3:53	67.6	64.9
4	102	1:33	97.6	96.9
5	54	0:50	97.9	97.9
7	32	0:31	98.0	98.9
10	47	0:49	97.7	98.6
14	18	0:19	97.9	98.4
18	24	0:26	98.0	98.3

* Brain Maker training parameters: training tolerance = 0.075; testing tolerance = 0.075; learning rate adjustment type = heuristic; initial weights set randomly

We had acquired a total of 10,528 data sets by running BF simulations. The simulation results (min-terms) were logarithmically transformed before being utilized for FFNNM training. We used 90% of the data sets as the *training set* and the other 10% as the *validation set*. A total of 72 different configurations of FFNNM were used to collect the data on FFNNM learn-ability. A given FFNNM was considered to be sufficiently trained when it had learnt 97.5% of the training facts, 1000 epochs were done or when further convergence was not observed.

Note that we repeated our experiments at least three times for every FFNNM configuration in order to find the best prediction performance and to alleviate the chances of ending up with local minima. We noticed that just 5 neurons in the hidden layers provided reasonable training and validation accuracies. The training and validation accuracies being close to each other (as shown in Table 1) validate the performance of our FFNNMs. Figures 2 and 3 illustrates the simulation and FFNNM results for APL complexity behavior for 10 and 14 variables respectively.

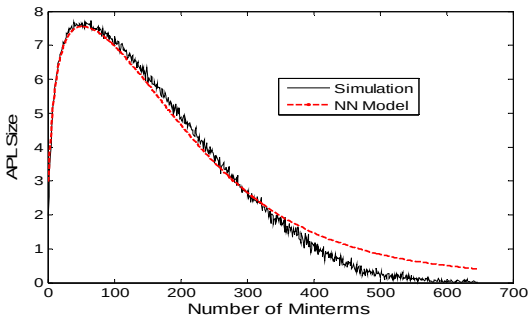

Figure 2. Comparison of APL complexity for 10 variables

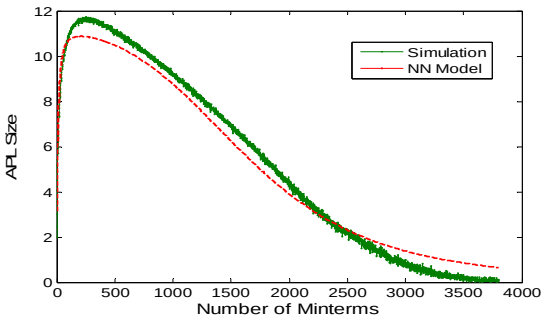

Figure 3. Comparison of APL complexity for 14 variables

III. Neural Network Model Validation

The FFNNM with experimental data has one-time cost of training, after which the model can be run very quickly (few micro-seconds or less) to predict the APL complexity of various functions with different number of variables and min-terms. Table 2 illustrates the ISCAS benchmark circuit [16] validation results for simulation using CUDD package and the proposed FFNNM.

The ISCAS benchmarks are sets of multi-input compound Boolean expressions, because the randomly generated BFs used for the experiments were single output SOP expressions and the benchmark functions were split into

TABLE 2. FFNNM validation with ISCAS benchmark circuits.

Circuit Name	Total Mean Value		Standard Deviation	Correlation
	Actual	NN Model		
5xp1	1.029	1.011	0.160	0.941
alu4	0.884	0.897	0.103	0.811
apex4	0.720	0.680	0.030	0.991
apex7	0.881	0.910	0.064	0.961
b12	0.920	0.903	0.150	0.832
b9	0.747	0.906	0.128	0.920
C17	1.157	1.103	0.098	1.000
c8	0.922	0.959	0.085	0.996
cc	0.761	0.904	0.121	0.723
cht	1.045	1.059	0.020	0.853
clip	0.953	0.920	0.043	0.636
cm138a	0.987	0.928	0.000	1.000
cm162a	0.685	0.813	0.050	0.995
cm163a	0.756	0.802	0.078	0.901
cm82a	1.356	1.175	0.013	0.998
cmb	0.374	0.408	0.004	1.000
con1	0.933	0.893	0.059	1.000
cu	0.644	0.702	0.085	0.962
decod	0.666	0.897	0.131	1.000
i6	1.018	1.037	0.024	0.862
i7	1.011	0.973	0.055	0.141
inc	0.836	0.846	0.055	0.685
misex1	0.950	0.926	0.112	0.749
Misex3	0.680	0.871	0.165	0.976
Pcle	0.602	0.690	0.087	0.889
rd53	1.184	0.785	0.649	-0.644
rd73	0.841	0.491	0.087	0.864
Sao2	0.373	0.397	0.019	0.026
Sct	0.763	0.847	0.089	0.973
sqrt8	1.050	0.957	0.115	0.248
Squar5	0.907	0.773	0.339	-0.280
ttt2	0.741	0.790	0.048	0.987
X2	0.866	0.928	0.100	0.893
x4	0.670	0.731	0.054	0.961
z4ml	0.837	0.847	0.079	0.543
Average				0.754

multiple single-output expressions, and then expanded directly to SOP term. For each of these expressions, the node count was computed using the CUDD package. The curves are almost similar geometrically. To increase the comparability of the data, each APL count was divided by the peak of the APL count for the given variable count.

For some benchmarks, lack of variation made the correlation meaningless. But, for the complete set of 541 circuits, the FFNNM correlation coefficient of about 0.754 is very significant, the chances of getting this assuming the measures are actually uncorrelated is less that 1 in a million: firm evidence of a significant relation between the two measures. Given the correlation coefficient, near 0.754, and that the model is predicting the average value of the node count (which is liable to considerable deviation) the model is judged to have passed a non-trivial test of its relevance. It can be inferred from these results that the FFNNM is a better model on prediction of the APL complexity if the input data range is known. Although the benchmark circuits considered had up to 94 inputs, mostly those benchmarks consisted of product terms of 1-14 variables. The circuits for all outputs were measured. It was observed that the term-variable count combinations were almost all to the left of the peak complexity, and thus still in region of logarithmic complexity. So, empirically the most important part of the model is the logarithmic rise, and it was this part that has been validly tested by the benchmark circuit analysis.

Due to the above factor, one of the important consideration is what is the largest variable should be for which these curves need to be generated in order to justify the FFNNM for benchmarks. It is obvious that curves are going to be more difficult to generate for larger number of variables because of the lack of sample product terms that can be extracted from the benchmarks.

IV. CONCLUSION

We have introduced a FFNNM for deriving the evaluation time complexity of BFs. The FFNNMs were obtained through the training utilizing the experimental data for random BFs. The evaluation time of the benchmark circuits follows similar pattern as the FFNNM, so it is shown to have strong descriptive power for the benchmark data. Therefore, the FFNNMs are promoted as a method of predicting for a given BF, APL complexity measures which provides evaluation time of a VLSI circuit implementation. In light of the results, we also conclude that the FFNNMs proposed in this work could be a valuable tool for exploring the complex computational capabilities of NNs.

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] I. Wegener, “The Complexity of Boolean functions,” *Wiley and Sons. Inc*., 1987.

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

[4] R. E. Bryant, “Graph−Based Algorithm for Boolean Function Manipulation,” *IEEE Transaction of Computers*, Vol. 35, pp. 677-691, 1986.

[5] R. Ebendt, S. Hoehne, W. Guenther, and R. Drechsler, “Minimization of the expected path length in BDDs based on local changes,” *Proceedings of Asia and South Pacific Design Automation Conference*, pp. 866-871, 2004.

[6] R. Ebendt, S. Hoehne, W. Guenther, and R. Drechsler, “Minimization of the expected path length in BDDs based on local changes,” *Proceedings of Asia and South Pacific Design Automation Conference*, pp. 866-871, Yokohama, 2004.

[7] Y. Y. Liu, K. H. Wang, T. T. Hwang, and C. L. Liu, “Binary decision diagrams with minimum expected path length," *Proceedings of DATE 01*, pp. 708–712, 2001.

[8] N. Drechsler, M. Hilgemeier, G. Fey, and R. Drechsler, “Disjoint Sum of Product Minimization by Evolutionary Algorithms,” *Proceedings of Applications of Evolutionary Computing, Evo Workshops*, pp. 198-207, 2004.

[9] N. Ramalingam, S. Bhanja, “Causal Probabilistic Input Dependency Learning for Switching model in VLSI Circuits,” *Proceedings of ACM Great Lakes Symposium on VLSI*, pp. 112-115, 2005.

[10] S. Bhanja, K. Lingasubramanian, and N. Ranganathan, “Estimation of Switching Activity in Sequential Circuits using Dynamic Bayesian Networks,” *Proceedings of VLSI Design 2005*, pp. 586-591, 2005.

[11] M. Raseen, P.W.C. Prasad, and A. Assi, “An Efficient Estimation of the ROBDD's Complexity” in *Integration - the VLSI journal*, Elsevier Publication, Vol. 39(3) pp. 211-228, 2005.

[12] I. Parberry, Circuit Complexity and Neural Networks. *MIT Press*, 1994.

[13] M., Caudill, “AI Expert: Neural Network Primer,” *Miller Freeman Publications*, 1990.

[14] F. Somenzi, “CUDD: CU Decision Diagram Package,” ftp://vlsi.colorado.edu/ pub/., 2003.

[15] BrainMaker User’s Guide and Reference Manual, 7th ed. *California Scientific Software Press*, 1998

[16] M. Hansen, H. Yalcin, and J. P. Hayes, “Unveiling the ISCAS-85 Benchmarks: A Case Study in Reverse Engineering,” *IEEE Transaction on Design and Test*, vol. 16, pp. 72-80, 1999.

[17] F. Brglez and H. Fujiwara, “A neutral netlist of 10 combinational circuits and a target translator in Fortran,” *in Proceedings of International Symposium on Circuit and Systems, Special Sess. On ATPG and Fault Simulation*, 1985, pp. 663-6985.