ACCEPTED: A. Beg, P. W. C. Prasad, M. Arshad, and S. K. Hasnain, "Using Recurrent Neural Networks for Circuit Complexity Modeling," in 10th IEEE International Multitopic Conference (ITMIC'06), Islamabad, Pakistan, 2006, pp. 194-197.

Using Recurrent Neural Networks for Circuit Complexity Modeling

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

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

Mirza M. Arshad
Department of Electrical Engineering,
University of Engineering & Technology, Lahore, Pakistan
arshad@scholaris.com

Khursheed Hasnain
PN College of Engineering, Karachi, Pakistan
skhpnec@hotmail.com

***Abstract*-Being able to model the complexity of Boolean functions in terms of number of nodes in a Binary Decision Diagram can be quite useful in VLSI/CAD applications. Our investigation showed that it is possible to use the recurrent neural network (RNN) models for the prediction of circuit complexity. The modeling results matched closely with simulations with an average error of less than 1%. The correlation coefficient between RNN's predictions and actual results for ISCAS benchmark circuits was 0.629.**

I. Introduction

A. Boolean Functions

VLSI for the past several decades has benefited by following the Moore's Law [1] that had predicted doubling of transistor on a chip every year. Constant miniaturization of circuit has increased the complexity of VLSI design many folds. These VLSI designs can directly benefit from optimal representation of a circuits as sets of Boolean functions (BFs) [2]. BFs find applications in areas such as combinational logic verification [3] sequential-machine equivalence [4], logic optimization of combinational circuits [5], test pattern generation [6], timing verification in the presence of false paths [7], and symbolic simulation [8].

Researchers have in the past tried to classify Boolean functions on the basis of different complexity measures, for example, the minimum size to implement a computing entity [10]. The way a Boolean function is implemented directly affects the computation and memory resources. Being able to estimate the circuit complexity based on Boolean functions is useful for conducting design feasibility studies [11]. Mathematical and feed forward neural networks (NN) models have been used previously to model the BF complexity [12][13][14].

B. Neural Networks

NNs are based on the principle of biological neurons. An NN may have one or more input and output neurons as well as one or more (hidden) layers of neurons interconnecting the input and output neurons. In the well-known feed-forward NNs (FFNNs), the outputs of one layer of neurons send data (only) to the next layer (Fig. 1.) Recurrent NNs not only make use of current inputs (time *t*) but also the internal state (time *t-1*, *t-2*,…) so on each time-step, new inputs are fed back into the network [16] (Fig. 2.) In the past, both FFNNs and RNNs have been successfully used in developing prediction models [14][17].

Back-propagation (BP) is a common scheme for creating (*training*) the FFNNs. During the process of NN-creation, internal *weights* of the neurons are iteratively adjusted so that the outputs are produced within desired accuracy [15]. The RNNs are trained using a variation of BP called *back-propagation through time* (BPTT) in which the weights of the hidden layers are adjusted using not only the present inputs but also the previous states [16][17]. The training process of the NNs requires that the *training set* (input-output data-sets) be chosen carefully. The selected dataset usually needs to be pre-processed prior to being fed to a NN [18].

The objective of this paper is to investigate the feasibility of using RNNs to predict the complexity of BFs. Section II.A this paper discusses the data acquisition for RNN model development and Section II.B shows how the data was processed before training the RNNs. Section II.C discusses the RNN configuration and training and Section II.D presents the results and analysis. Lastly, Section **Error! Reference source not found.** presents the conclusions of this research.

II. RNN Model for Boolean Function Complexity Prediction

A. Data Source

We used Colorado University Decision Diagram (CUDD) package [19] to determine the complexity of a variety of BFs (in terms of number of nodes). 100 SOP terms were generated for variable count $n \{n \mid n = 1, 2 \ldots 14\}$. This process was

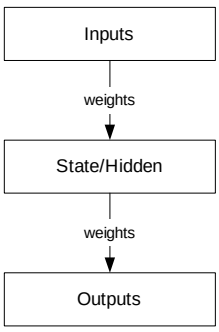

Fig. 1. Topology of a simple feed-forward neural network

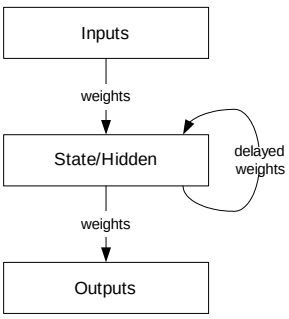

Fig. 2. Topology of a simple recurrent neural network

repeated until BFC became 1. These results are plotted in Fig. 3.

We observed that BFC increases as the number of minterms (product terms) increases until it reaches a peak corresponding to the maximum complexity. Beyond the peak, the minterms tend to simplify and BFC declines and reaches a final value of zero.

B. Data Processing

A purpose of data pre-processing is to ensure that the inputs with larger absolute values should be given the same importance as the inputs that have smaller magnitudes [18][21].

In our case, we can see the need for data transformation in Fig. 3 that shows BFC curves for 2- to 14- variables. (The plotted data was acquired from Boolean function simulations as explained in Section A [14].) The number of *'minterms'* in a function is shown on the horizontal axis; on the vertical axis, *'nodes'* represents the complexity of a BF. The curves for 2-6 variables, in their original form, are not only visually hard to see but also hard for a NN to *learn*. The minimum and maximum values on both axes of these curves vary widely and non-linearly. So the smaller variable curves could be ignored altogether during the NN-training process; data processing alleviates this issue by transforming the curves that have a similar set of minimum and maximum ranges.

In this paper, we first transform the data by taking logarithm of minterms and number of nodes. We did not apply any transformation to variable values due to their existing linearity and their limited range of 2 to 14. Effect of data transformation on the 'visibility' of data is evident in Fig. 4.

C. RNN Configurations and Training Method

In our research, BFC is a function of number of variables and number of minterms; so the RNN can be represented by a block diagram of Fig. 5. This means that in our RNN models, the input neuron count is fixed at 2 (one for 'minterms' and the other for 'variables') and output neuron count at one (for 'node' prediction).

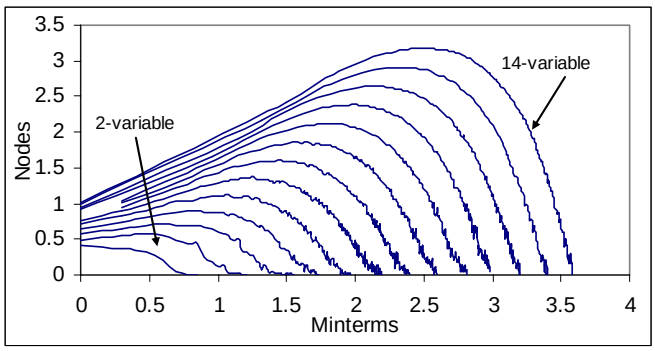

Fig. 4. Effect of logarithmic transformation on the original data. The general shape of the curves changes while making all of them more 'visible'.

For the RNN training purposes, we started with the RNN learning rate of 0.1 and then adjusted it down in increments of 0.01 as the training progressed. The initial weights were randomly set. Although, we experimented with a large number of RNN configurations, we only used 30 of these configurations to collect the data on RNNs' learnability. One recurrent copy of inputs and one recurrent copy of output was used in all the RNNs. The RNN configurations were varied by changing the number of neurons (i.e., 2, 4, 6 … 18, 20) in the (single) hidden layer. Each configuration was used three times to train a completely new NN to take into consideration the effect of local minima. The averages of the three training runs were used to calculate the training and validation accuracy for a given configuration.

The input-output dataset was divided into two sets: (1) *training set:* 90% of the facts were used to train the RNN; (2) *validation set:* 10% facts were used for validating a trained RNN. During NN-training, only the training set is presented to the NN, and not the validation set. The properly converging RNNs provided maximum accuracy well before they reached 1000 epochs, so we stopped NN training when 1000 epochs were completed.

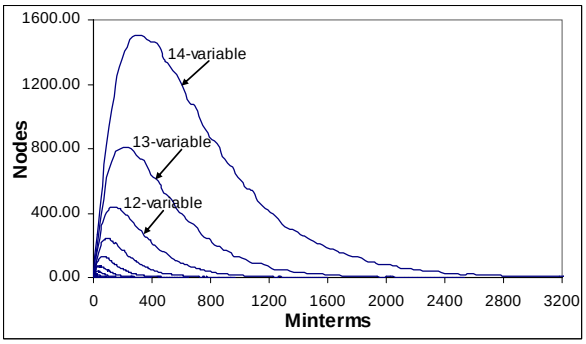

Fig. 3. Boolean function complexity data for 2- to 14-variables in original (raw) format. The smaller variable curves (lower left corner) are not as visible as the large ones and have the potential of not being correctly learnt by the NN.

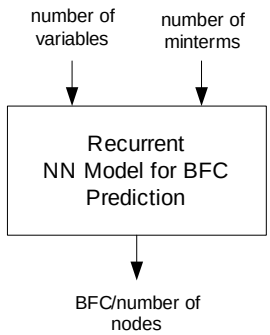

Fig. 5. Inputs and outputs of BFC prediction RNN. Two inputs are the *number of variables*, and *number of minterms*, and the output is BFC in terms of *number of nodes*.

ACCEPTED: A. Beg, P. W. C. Prasad, M. Arshad, and S. K. Hasnain, "Using Recurrent Neural Networks for Circuit Complexity Modeling," in 10th IEEE International Multitopic Conference (ITMIC'06), Islamabad, Pakistan, 2006, pp. 194-197.

D. RNN Training Results and Analysis

The RNN training performance for each of the 2 to 14 variables for all samples is shown in Fig. 6; the average error tends to fluctuate. For example, number of nodes (BFC) for 4 variables is predicted with an average error of 9.8% and for 14 variables with just 0.03% error.

We used combinations of number of variables and minterms to predict the BFC from our RNNs. Two comparative plots (actual BFC values from simulations vs. the predicted BFC values) are shown in Fig. 7 and Fig. 8 for 10 and 12 variables, respectively. We observed that the RNNs were able to learn the overall behavior of the original input-output data sets.

Complete RNN models' average error came to be less than 1%. Additionally, we used ISCAS benchmark circuits [22] to compare the performance of RNN with actual benchmark results derived using CUDD package. (ISCAS benchmarks are sets of multi-input compound Boolean expressions). This comparison is shown in Table 1. For the given circuits, the RNN correlation coefficient of was found to be 0.629, which is an evidence of a significant relationship between the two measures.

III. Conclusions

In this paper, we investigated the use of RNNs for BFC prediction. The RNNs were able to learn from the BFC behavior for 2 to 14 variables with an overall error of less than 1%; the correlation between the actual and predicted values for ISCAS benchmarks was 0.629. An advantage of the proposed RNN is that a single NN can be used to predict the BFC for a wide range of variables. Use of RNNs for benchmark circuits is a subject of our ongoing research. Use of different data pre-processing techniques to improve the RNN model performance is also being looked into.

TABLE 1
COMPARISON OF RNN'S PREDICTIONS WITH ISCAS BENCHMARK CIRCUITS. THE OUTPUT OF RNN IS CLOSELY CORRELATED WITH THE ACTUAL VALUES.

Circuit Name	Number of Circuits	Mean (Nodes)		Mean Difference with Actual	Standard Deviation	Correlation
		Actual	RNN			
5xp1	7	0.845	0.696	0.181	0.917	0.449
Apex7	43	0.699	0.651	0.198	0.938	0.500
b1	1	0.893	0.540	-	-	-
b12	9	0.846	0.822	0.178	0.878	-0.206
C17	2	0.980	0.540	0.000	-	-
cc	11	0.650	0.608	0.097	0.744	0.567
cht	36	0.954	0.651	0.033	0.956	0.708
clip	5	0.910	0.795	0.195	0.991	0.838
cm138a	8	0.858	0.792	0.000	0.000	1.000
cm162a	4	0.413	0.447	0.195	0.910	0.550
cm163a	5	0.593	0.614	0.152	0.950	0.572
cm82a	2	0.892	0.631	0.103	0.975	0.600
Con1	2	0.817	0.584	0.077	1.000	1.000
Cu	10	0.487	0.331	0.236	0.994	0.790
decod	16	0.544	0.388	0.000	1.000	1.000
inc	9	0.862	0.768	0.115	0.986	0.772
majority	1	0.925	0.177	-	-	-
misex1	7	0.781	0.665	0.160	0.954	0.328
pm1	9	0.546	0.541	0.186	0.959	0.689
rd53	3	0.855	0.542	0.179	0.857	0.788
rd73	3	0.564	0.386	0.113	0.999	0.969
Sao2	4	0.567	0.620	0.023	1.000	0.995
Sqrt8	3	0.967	0.917	0.045	-0.946	-0.821
Squar5	8	0.887	0.529	0.035	0.787	0.826
Xor5	1	0.709	0.401	-	-	-
Z4ml	4	0.813	0.765	0.159	0.996	0.923
Total	**213**			**0.121**	**0.811**	**0.629**

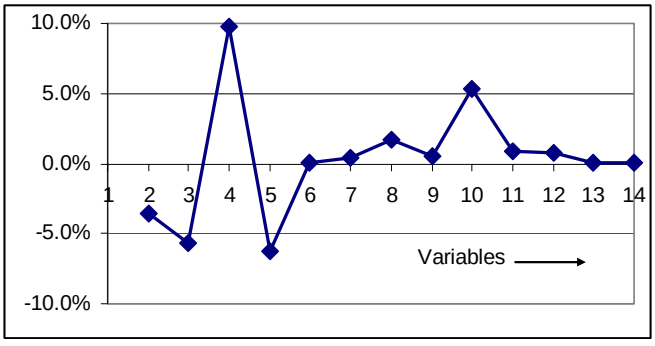

Fig. 6. Prediction accuracy for 2 to 14 variables. Predictions made for larger variables (e.g. 12, 13 and 14) have smaller error than the lower variables (2, 3 and 4).

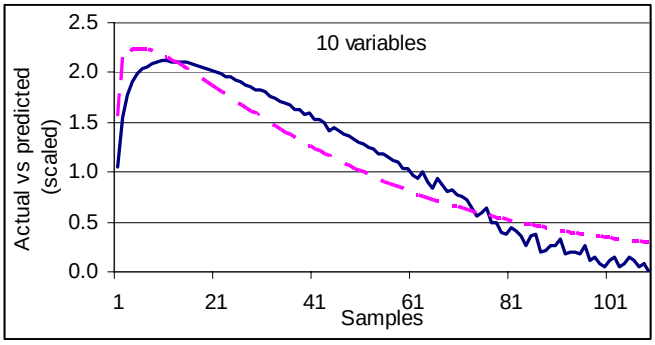

Fig. 7. Actual (solid line) vs. RNN predictions (dashed line) for 10 variables.

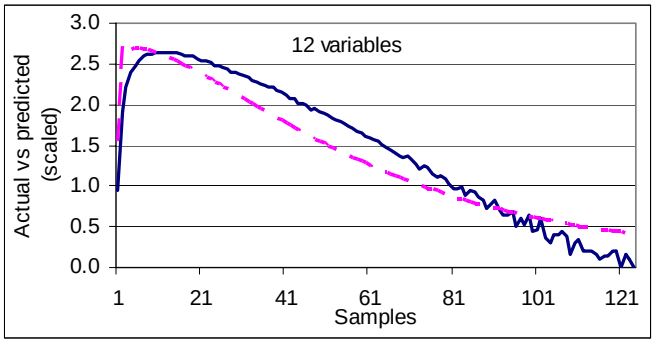

Fig. 8. Actual (solid line) vs. RNN predictions (dashed line) for 12 variables.

IV. References

[1] G. E. Moore, "Progress in Digital Integrated Electronics," IEEE IEDM, 1975, pp. 11-13.

[2] R.E. Bryant, "Graph-based algorithm for Boolean function manipulation," IEEE Trans. Computers, Vol. 35, 1986, pp.677-691.

[3] S. Mahk, A, Wang, R. Brayton, and A, Sangiovanni-Viicentelli, "Logic Verification using Binary Decision Diagrams in a Logic Synthesis Environment," in Proc. Int. Conf. CAD (KCAD-88), 1988, pp. 6-9.

[4] 0. Coudert. C. Berthec and J. Madre, "Verification of Sequential Machines using Boolean Function Vectors,"in IMEC-IFIP lnt. Workshop on Applied Formal Methods for Correct VLSI Design, 1989.

[5] S. Muroga, Y. Kambayashi. H. Lai, and J. Culliney, "The Transduction Method," IFTEE Trans. Cow., vol. 38, 1989, pp. 1404-1424.

[6] M. A. Breuer and A. D. Friedman, 1976 "Diagnosis and Reliable Design of Digital Systems," Computer Science Press.

[7] P. McGeer and R. Brayton, "Efficient Algorithms for Computing the Longest Viable Path in a Combinational Network," in Proc. 26th Design Automation Conference, 1989, pp.561-567.

[8] K. Cho, "Test Pattern Generation for Combinational and Sequential MOS Circuits by Symbolic F&t Simulation," PhD thesis, Carnegie Mellon University, 1988.

[9] Shannon, "The synthesis of two-terminal switching circuits," Bell Syst. Techn. J.28, pp. 59-98.

[10] M. Nemani, and F.N. Najm, "High-level power estimation and the area complexity of Boolean functions," Proc. of IEEE Intl. Symp. on Low Power Electronics and Design, 1996, pp: 329-334.

[11] A. Assi, P.W. C. Prasad , B. Mills, and A. El-Chouemi, "Empirical Analysis and Mathematical Representation of the Path Length Complexity in Binary Decision Diagrams", in Journal of Computer Science, Science Publications, Vol. 2(3), 2005, pp. 236-244.

[12] 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, pp. 973-978.

[13] L. Franco, "Role of function complexity and network size in the generalization ability of feedforward 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.

[14] A. Assi, P. W. Chandana Prasad, and A. Beg, Modeling the Complexity of Digital Circuits Using Neural Networks, WSEAS Transactions on Circuits and Systems, June 2006.

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

[16] M. Boden, "A guide to recurrent neural networks and backpropagation", (website) http://www.itee.uq.edu.au/~mikael/papers/rn_dallas.pdf, Sept. 2006.

[17] L. Medsker, L. Jain, "Recurrent Neural Networks: Design and Applications", CRC Press, 1999.

[18] K. Yale, "Preparing the right data for training neural networks," IEEE Spectrum, Vol. 34, Issue 3, Mar. 1997, pp. 64-66.

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

[20] D.L. Tuck, "Practical polynomial expansion of input data can improve neurocomputing results", ANNES'93, Los Alamitos, CA, 1993, pp. 42-45.

[21] T. Masters, *Signal and Image Processing with Neural Networks*" John Wiley & Sons, Inc., 1994.

[22] M. Hansen, H. Yalcin and J. P. Hayes, "Unveiling the ISCAS-85 Benchmarks: A Case Study in Reverse Engineering," IEEE Trans. On Design and Test, 1999, vol. 16, pp. 72-80.