Journal of VLSI Design Tools & Technology
Volume 1, Issue 1-3 Compilation 2011

STM JOURNALS

Power Estimation for VLSI Circuits Using Neural Networks

Mr. B. Srinath
Asst. Prof (O.G), Department of ECE, SRM University, Chennai, India

ABSTRACT

Neural network based VLSI power estimation is done which estimates power in VLSI circuits from its input /output and gate information, without simulation and analysis of its detail structure and the interconnections. Artificial neural network is created which helps in estimation of power. Power estimation results from the [2] [3] are used as the training vector for the network .The network is trained using Back-propagation algorithm. A simple recurrent network is also introduced called Elman network which uses the back propagation for training the network .Analysis such performance measures, regression analysis and error analysis are done to justify that the trained network performances well. A comparative analysis on both the networks is done to show that the neural network based approach, estimates power in faster rate. The results, concludes that the Elman network converges faster when compared to the conventional feed forward neural networks.

Keywords: *Neural Networks, VLSI, Back propagation network (BPN), Back-propagation algorithm, Recurrent Network, Elman Neural Network (ENN)*

Author for Correspondence E-mail: bsrinath86@gmail.com

INTRODUCTION

Complexity of VLSI designs has increased rapidly due to a fast growth of semiconductor manufacturing techniques making power consumption a major design concern. Due to limited battery life, possible cooling cost and corresponding reliability issue, power consumption can become a more critical design choice than area or speed of operation. The main conceptual difficulty in power estimation is the trade-off between efficiency and accuracy. The power measured using a specific set vector is assumed to be average power consumption. It is very difficult to collect all the representative vectors for the VLSI circuits and simulated them because it is too exhaustive even though it brings the most accurate results. Gate-level power estimation techniques might not be useful in the sense that it can be too late to find out some problems. Accordingly, higher–level power estimation is necessary and essential to avoid the costly re-design steps. Also, as millions of gates exits on a single chip, it has become practically quite impossible to do a power-aware designs of each and every functional component at a low level of abstraction. So, power estimation techniques for designs have gained a lot of attention in research. Several techniques are proposed for the power estimation in Literature. An improved simulation based Monte Carlo method of power estimation is given in the [12]. Their method considers correlations in time and space between the inputs, the internal nodes, state nodes, etc. In their approach, a mechanism for generating mutually independent samples using multiple copies of circuit which are simulated parallel, using mutually independent input vector streams are considered. Each copy of the machine results

in an independent sample of power, and the samples are then collectively analyzed to determine when to stop the simulation process. This method also saved time comparing to simulation only. Bayesian networks are adopted in switching activity estimation of the VLSI circuits [11]. This method encapsulates all the dependencies both in the internal nodes and inputs, in reasonable time and with the accuracies. The model handles spatio-temporal dependencies of the nodes. This model is accurate and captures higher order correlation among lines and is not restricted to pair-wise correlations. This approach also captures the dependence of power dissipation from its input/output switching activity. The model is used to estimated power consumption from any given input /output signal streams. The three-dimensional look up table consists of average input signal probability, average input transition density and average output zero-delay transitions density. It does not require any specialized analytical equations for power estimation. In [1], least square estimation gives a more time saving method on the basis of Monte Carlo approach. The minimization of mean square error value, each iteration, is performed by two statistical algorithms: sequential least square estimation (SLS) and recursive least square estimation (RLS). This algorithm is proposed over least fitting technique to relax computational requirements. There is also a multi mode power modeling approach adopting Maximum-Like hood estimation [2]. In [4], power macro-model has been for behavioral library components .It is emphasized that high level synthesis is gaining acceptance is the design community. This technique is based on look up table with three dimensions (average input probability, average input transitions probability, and average output transitions probability) .All the methods above should care for detail structure and the partial simulation results for VLSI circuit and the time consumed is always in some kind of proportion to the scale of VLSI circuits.

A neural network based modeling approach is proposed over the previous approaches to efficiently model the high level power consumption. The capability of the neural network is exploited as an algorithm of smart approximation. Historically, artificial neural networks (ANN) have been used in universal approximation (i.e., mapping inputs and outputs) tools capable of learning from their environment, tools for finding non-evident dependencies between the data and so on. Neural networks are modeled after a brain and how it processes the information. Hence, neural networks are very powerful tools. As a first step, we need to train a neural network to make it familiar with the environment.

A Simple back-propagation is used for experiment in [6] .The neural network model proposed in [5] efficiently characterized a macro model for high level power estimation. The corresponding results are presented comparing with the previous approaches. It is justified from the results that the proposed neural network model performs quite

generally independent for the variation of the power consumption for different input patterns. The model in [6] has to take care of some more parameters during macro modeling.

This work puts forward a neural network in order, to formulate all the above said problems. The neural network proposed only the following to train the network. They are:

1. Only information such as input/output and cell number.
2. Detail structure and interconnections of VLSI is not necessary.
3. Needs the power estimation results of the serious benchmark circuits to train neural network.

The network is trained by standard back-propagation algorithm in which the network weights are moved along the negative gradient of the performance function. After training the network from the training and simulation vector extracted from ISCAS89 and previous works certain performance measures like Regression test for R-value evaluation and calculation of Relative Root Mean Square Error (RMSE), Relative error on average (AVGE) as in [6] and N/P ratio, time elapsed per epoch per exemplar. The results are satisfactory in some cases but not in certain states.

There are many variations of the architectures and learning rules. Some of them are simple recurrent networks and Fully Recurrent networks. To overcome the problems in training of the neural networks by standard back propagation algorithm, recurrent networks are employed in this work. It is closely related to the better-known probabilistic neural network. Recurrent network can be used for the typical nonlinear multivariable regression problem as in [14].

There are two types of recurrent neural networks such as fully recurrent networks and partially recurrent networks. Partially recurrent networks called Elman network is used in this work because it consists of feedback path that allows the network to recognize and generate temporal patterns and spatial patterns. This means that after training, interrelations between the current input and internal states are processed to produce the output and to represent the relevant past information in the internal states.

A comparative study is done from the observations from the results with the conventional back propagation network. The result concludes that Elman network trains well when compared with the BPN in both the sample.

NEURAL NETWORKS

Why Neural Networks?

The long course of evolution has given the human brain many desirable characteristics not present in modern parallel computers such as Massive parallelism, Distributed representations and computation, Learning ability, Generalized ability, Adaptivity,

Inherent contextual information processing, Fault tolerance and Low energy consumption. A trained neural network can be thought of as an "expert "in the category of information it has been given to analyze. This "expert" can then be used to provide projections given new situations of interest and answer "what if" questions.

Other advantages are

1. Adaptive learning. 2. Self-Organization. 3. Real-time operation 4.Fault tolerance via redundant information coding.

Analysis in Neural Networks

Analysis of the networks is done in the following ways: 1.Regression test And 2. Performance measures

Regression Test

Here linear regression test is carried over for the analysis of the neural network. A concept of post processes is used to analyze the trained network.

Mathematical modeling of Regression Test:

Two net are adopted in R-value analysis with all 9 columns in the Table 3.1 served as a training unit named as NN1 and NN2. 200 and 100 times training are applied on NN1 and NN2 separately. The 200 R-values of NN1 is called as "Sample_NN1" while the first 100 R-value in "Sample_NN1" called as "Sample1_NN1"and the last 100 R-values of NN1 is called as "Sample2_NN1". The 100 R-values of NN2 is called as "Sample_NN2".

All the R-values of these samples fall in the interval of {-0.5, 1}, and approaches 1.

Let {X1, X2, X3, X4} be the random variable of R-value of "Sample_NN1", "Sample1_ NN1", "Sample2_ NN1" and "Sample_ NN2" separately.

{X1, X2, X3, X4} do not conform to Normal Distribution and we unfold these variables to the interval of {-0.5, 1} mirrored by x=1. Then unfolded random variables{X1', X2', X3', X4'} are generated.

Cumulative distribution function for all the folded and unfolded is plotted, while the unfolded ones are more familiar to normal distribution function curves. Shapiro-Wilk test is applied to the 8 random variables, unfolded random variables are proved normal while original ones not. The analysis of R-value shows the power estimation result of neural network is of a probabilistic process.

Post processing

Post processes the trained network response with a linear regression. The post processing is done by network training set by performing a linear regression between one element of the network response and the corresponding target. In otherwise, a linear regression is performed between the network response and the target, and computes the correlation coefficient (R value) between the network response and the target.

Simulation process adopts the five set of vectors of "MC" left in training process. 5 input vectors served as input of the trained network and target vector is put into a linear regression test with simulation result vector.

Regression R-value directly shows the network performance (R=1 means perfect Correlation).

Performance measures

Two performance measures are considered as in [7] to determine the accuracy and novelty of the proposed technique over the conventional technique.

RMSE, Relative Root Mean Square Error $=\frac{\sqrt{MSE}}{AVG}$, where *AVG* is the average power on the test sample. **AVGE**, Relative error on average $=|AVG_{model} - AVG|/AVG$, where AVG_{model} is average power on the test sample employing a model. RMSE provides information on how well the pattern dependence of power dissipation is modeled; AVGE is a measure of the accuracy in estimation of average power.

BACK PROPAGATION NETWORK

Back propagation is a systematic method for training multi-layer artificial neural networks .It has a mathematical foundation that is a strong if not highly practical .It is a multi-layer forward network using extend gradient-descent based delta-learning rule, commonly known as back propagation (of errors) rule. Back propagation provides a computationally efficient method for changing the weights in a feed forward network, with differentiable activation function units, to learn a training set of input-output examples. Being a gradient descent method it minimizes the total squared error of the output computed by the net. The network is trained by supervised learning method. The aim of this network is to train the net to achieve a balance between the ability to respond correctly to the input patterns that are used for training and the ability to provide good responses to the input that are similar.

Training

The training algorithm of back propagation involves four stages, viz.

1. Initialization of weights
2. Feed forward
3. Back propagation of errors
4. Updation of the weights and biases.

During 'first stage' which is the initialization of weights, some random values are assigned. During feed forward stage each input unit receives an input signal and transmits this signal to each of the hidden units. Each hidden unit then calculates the activation function and sends its signal to each output unit. The output unit calculates the activation function to form the response of the net for the given input pattern.

Training the proposed Network

Input training vector of the neural network net is vector gatherings of ISCAS89 benchmark information, with no circuit information, interconnect information and simulation results involved.

Training target vectors is the power estimation results of the former works [2] [3], as in the table 3.1, "MC" and "SIM".

"SIM "is used to test the network convergence. 19 benchmark results are picked up from "MC" to serve as target training vector with corresponding benchmark information serve as input training vectors. All the input vectors are normalized with the output vector to ensure training convergence.

RECURRENT NETWORKS

A recurrent network (RNN) is a class of neural network where connection between units forms a directed cycle. This creates an internal state of the network which allows it to exhibit dynamic temporal behavior. Recurrent neural networks must be approached differently from feed forward neural networks, both when analyzing their behavior and training them. Recurrent networks can also behave chaotically. Usually , dynamical systems theory is used to model and analyze them, while a feed forward network propagates data linearly from input to output, recurrent networks also propagates data from later processing stage to earlier stages. The Elman and Jordan networks are known as "Simple Recurrent networks". In this work, Elman network with back propagation is used to train the input vector.

Elman Networks

The Elman network commonly is a two-layer network with feedback from the first-layer output to the first-layer input. This recurrent connection allows the Elman network to both detect and generate time-varying patterns. The Elman network has "tansig" neurons in its hidden (recurrent) layer and "purelin" neurons in its output layer. This combination is special in that two-layer networks with these transfer functions can approximate any function (with a finite number of discontinuities) with arbitrary accuracy. The only requirement is that the hidden layer must have enough neurons. More hidden neurons are needed as the function being fitted increases in complexity. Note that the Elman network differs from conventional two-layer networks in that the first layer has a recurrent connection. The delay in this connection stores values from the previous time step, which can be used in the current time step. Thus, even if two Elman networks, with the same weights and biases, are given identical inputs at a given time step, their outputs can be different because of different feedback states. Because the network can store information for future reference, it is able to learn temporal patterns as well as spatial patterns. The Elman network can be trained to respond to, and to generate, both kinds of patterns.

Training to Elman Networks

Elman networks can be trained with either of two functions, train or adapt. When you use the function train to train an Elman network the following occurs:

At each epoch,

1. The entire input sequence is presented to the network, and its outputs are calculated and compared with the target sequence to generate an error sequence.
2. For each time step, the error is back propagated to find *gradients* of errors for each weight and bias. This *gradient* is actually an approximation, because the contributions of weights and biases to errors via the delayed recurrent connection are ignored.
3. This gradient is then used to update the weights with the chosen back prop training function. The function trained is recommended.

When you use the function adapt to train an Elman network, the following occurs. At each time step,

1. Input vectors are presented to the network, and it generates an error.
2. The error is back propagated to find gradients of errors for each weight and bias. This gradient is actually an approximation, because the contributions of weights and biases to the error, via the delayed recurrent connection, are ignored.
3. This approximate gradient is then used to update the weights with the chosen learning function. The function learngdm is recommended.

SIMULATION RESULTS

With the input /output and number of gate information the network is trained. Totally, there are about 26 bench marks circuits are available. These bench marks are divided and the first 9 bench marks are named as Sample 1 and the next set of the 9 are named as Sample 2. The remaining bench marks can be applied to any of these trained networks so that it converges faster.

Performance Measures

The table 1 shows the Root Mean Square Error values of s table sample 1 and 2 while trained with BPN and Elman networks Table 2 shows the Average error for the sample 1 and sample 2 using BPN . Table 3 shows the Average error for the sample 1 and sample 2 using Elman Network.

Table I Root Mean Square Error values of s table sample 1 and 2 while trained with BPN and Elman networks.

Root Mean Square Error Values	
Back Propagation Network	
Sample 1	*0.41907*
Sample2	*0.07148*
Elman Network	
Sample 1	*0.3876*
Sample2	*0.1372*

Table II Average error for the sample 1 and sample 2 using BPN

Back propagation Network			
Sample 1		***Sample 2***	
Benchmarks	***AVGE***	***Benchmarks***	***AVGE***
S208	*0.7931*	*S641*	*0.9357*
S298	*0.7297*	*S713*	*0.9337*
S344	*0.9451*	*S820*	*0.9498*
S349	*0.9688*	*S838*	*0.9771*
S382	*0.9448*	*S953*	*0.9565*
S386	*0.952*	*S1238*	*0.8876*
S400	*0.9682*	*S1423*	*0.8728*
S420	*0.7321*	*S1488*	*0.9*
S444	*0.96533*	*S1494*	*0.5864*

Table III Average error for the sample 1 and sample 2 using Elman Network.

Elman Network			
Sample 1		***Sample 2***	
Benchmarks	***AVGE***	***Benchmarks***	***AVGE***
S208	*0.7624*	*S641*	*0.9344*
S298	*0.9249*	*S713*	*0.9344*
S344	*0.9249*	*S820*	*0.9344*
S349	*0.9249*	*S838*	*0.9344*
S382	*0.9249*	*S953*	*0.9344*
S386	*0.9249*	*S1238*	*0.9344*
S400	*0.9249*	*S1423*	*0.7295*
S420	*0.7627*	*S1488*	*0.9344*
S444	*0.9249*	*S1494*	*0.7295*

Table IV Performances from the Regression analysis

Networks	**R-Value**	**Comments**
Elman Network		
Sample 1	R=1.0	Posses perfect Linear correlation
Sample 2	R=1.0	Posses perfect Linear correlation
Back Propagation Network		
Sample 1	R=1.0	Posses perfect Linear correlation
Sample 2	R=0.9	Do not posses perfect Linear correlation

Regression Analysis

As said above the regression analysis is done for both BPN and the Elman Networks. Their performances from the Regression analysis are tabled in the table 4.

From the table 4 it is observed that the Elman network fits the normal distribution curve and posses perfect correlation for the both sample networks. But in the case of the Back propagation network which also showed perfect correlation for the sample 1 and not for the sample 2 says that Back propagation networks works well for limited circuits.

CONCLUSION

This paper forwards a neural network based method on VLSI power estimation. Experiments were made on ISCAS89 benchmark circuit. Different type of networks such as Back propagation network (BPN) and Elman Neural network (ENN) are used for the estimation of the power in VLSI. Performance Measures for the both type of networks are done and the calculation are made for analyzing the performance of the proposed networks. Regression analysis is done in the Elman network. It is observed that Elman network perfectly fits the normal distribution curve with regression values R=1 in both sample 1 and sample 2 which BPN does not. This shows that the proposed Elman network is trained properly and it performances well. The two types of error analysis are done which also shows that Elman networks possess less error gradient compared with the BPN. Five different runs are carried over with random initialization of weights in both ENN and BPN networks. This concludes that the Elman network is shows better BPN. Future work focuses on reducing time elapsed in the Elman Network.

REFERENCES

1. Ashok K. Murugavel, Ranganathan N., Chandramouli R. et al. *IEEE Transactions on Very Large Scale Integration(VLSI) systems* 2002. 12(1) 55–58p.
2. Chandramouli R. & Vamsi K .Srikantan *IEEE Transactions on Very Large Scale Integration (VLSI) systems* 2002. 12(11) 1244–1248p.
3. Hou, Ligang Wu & Songbo Wu on *Journal of Information and Computer Science* 2004. 1. 101-106p.
4. Benini L., Bogliolo A., Favalli M. et al. *Regression Models for Behavioral power estimation* in Proceedings of Integrated Computer-Aided Engineering. 1988. 5(2) 95–106p.
5. LiMin Fu. *Neural Networks in Computer Intelligence* Tata McGraw-Hill Publishing Company Limited, New Delhi.
6. Kuntal Roy. *Neural network based Macro-models for high level power estimation* on Proceedings of International Conference of

Computational Intelligence and Multimedia Applications. 2007. 12(13) 159-163p.

7. Mankar V. K. & Ghatol A. *Design of adaptive filter using Jordan/Elman Neural networks in a typical EMG signal Noise removal* on research article Hindawi Publishing Corporation on Advances in Artificial Neural systems. 2009.
8. Ananda Rao M. & Srinivas J. *Neural Networks algorithms and applications* Narosa publishing House, New Delhi.
9. Richard Burch & Farid N. Najm *IEEE Transactions on Very Large Scale Integration (VLSI) systems* 1993. 1(1) 63–71p.
10. Sanjukta Bhanja. & Ranganathan N. *IEEE Transactions on Very Large Scale Integration (VLSI) systems* 2003. 11(4) – 558567p.
11. Saxena V., Farid N. Najm & Ibrahim N. Najm Proceeedings on ED&TD 97. 416–420p.
12. Sivanandam S. N., Sumathi S., Deepa S. N. Tata McGraw-Hill Publishing Company Limited, New Delhi.
13. Tomasz J. Cholewo & Jacek M. Zurada *Neural Networks tools for Stellar light prediction* on Proceedings of IEEE Aerospace Conference Snowmass, Colo., USA. 1997. 3. 415–422p

APPENDICES

Training and simulation vector extracted from ISCAS89 and previous works

Bench	IN	OUT	DFF	INV	Gate	AND	NAND	OR	NOR	SIM[2]	MC[1]
S27	4	1	3	2	8	1	1	2	4	125	-
S208	10	1	8	38	66	21	15	14	16	459	0.00698
S298	3	6	14	44	75	31	9	16	19	819	0.00912
S344	9	11	15	59	101	44	18	9	30	1024	0.01846
S349	9	11	15	57	104	44	19	10	31	1035	0.01856
S382	3	6	21	59	99	11	30	24	34	1132	0.01048
S386	7	7	6	41	118	83	0	35	0	1132	0.0162
S400	3	6	21	58	106	11	36	25	34	-	0.01065
S420	18	1	16	78	160	49	29	28	34	-	0.00903
S444	3	6	21	62	119	13	58	14	34	-	0.01172
S641	35	24	19	272	107	90	4	13	0	-	0.03629
S713	35	23	19	254	139	94	28	17	0	-	0.03743
S820	18	19	5	33	256	76	54	60	66	-	0.02831
S838	34	1	32	158	288	105	57	56	70	-	0.01292
S953	16	23	29	84	311	49	114	36	112	-	0.02458
S1238	14	14	18	80	428	134	125	112	57	-	0.06347
S1423	17	5	74	167	490	197	64	137	92	-	0.07181
S1488	8	19	6	103	550	350	0	200	0	-	0.05648
S1494	8	19	6	89	558	354	0	204	0	3933	0.06018
S5378	35	49	179	1775	1004	0	0	239	765	12004	0.23357
S9234	19	22	228	3570	2027	955	528	431	113	-	0.28004
S13207	31	121	669	5378	2573	1114	849	512	98	37748	0.35404
S15850	14	87	597	6324	3448	1619	968	710	151	39985	0.51991
S35932	35	320	1728	3861	12204	4032	7020	1152	0	122048	-
S38417	28	106	1636	13470	8709	4154	2050	226	2279	-	1.14518
S38584	12	278	1452	7805	11448	5516	2126	2621	1185	112514	1.87987