Volume 3, Issue 11, November 2013 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

BPNN Based Power Estimation of Sequential Circuits

S.Arun Kavi Arasu[1], ,Fiona.E.Josy[3],
[1,3]*P.G. Scholar,*
Department Of Applied Electronics,
Dr.Pauls Engineering College, Villupuram, India

N.Manibharathi[2], Kanimozhi Rajasekaran[4]
[2,4] *P.G. Scholar,*
Department Of Applied Electronics,
I.F.E.T College Of Engineering, Villupuram, India

Abstract— Early Power estimation is important in VLSI circuits, because it has a significant impact on the reliability of these circuits. Power estimation is a tradeoff between precision and estimation time. Simulation based power estimation techniques are time consuming. This work reports an artificial neural network based method for power estimation of ISCAS'89 Benchmark circuits, by employing back propagation algorithm (BPNN). This method can estimate power quickly and precisely from I/O and gate information of the VLSI circuit, without requiring detailed structure of the circuit and its interconnection. Power estimation results reported in the literature for 20 ISCAS' 89 Benchmark circuits were used to the train the neural network. The trained network is tested on 5 circuits left out during the training process. The results of the tested circuits were validated by performing regression analysis. The BPNN was trained with training functions namely Traingdx and Traingdm.

Keywords:

I. INTRODUCTION

The dramatic reduction in device sizes and increased features added on portable communication and computing systems have made power consumption as a major criterion to be considered in VLSI circuit design. Therefore, there is a need to estimate the power consumption during the design phase itself. It is impractical to stimulate a large VLSI circuit exhaustively with all possible representative input vectors, to measure power. Hence power is measured for a specific set of random vectors, and is termed as average power consumption. Power dissipation depends on the operating environment of the circuit, namely the input vectors being fed in. Probabilistic methods [1, 2] to estimate power dissipation overcome the input pattern dependency problem, but to achieve good accuracy, it is essential to model the correlation between the logic signals which is expensive. Monte Carlo based techniques [3] can model the correlation between the internal signals precisely. This technique is most preferable for combinational circuits but not for sequential circuits. Monte Carlo technique decouples the combinational portion of the sequential circuit from the flip-flops and analyze them separately which leads to inaccuracies in power estimation. An improved Monte Carlo method [4] was presented which considered the correlations in time and space between the inputs, internal nodes and state nodes. This method saved time compared to simulation only method. Least square estimation [5] provided more time savings when compared to Monte Carlo methods. This technique minimized the mean square error during each iteration by using two statistical algorithms namely Sequential Least Square (SLS) and Recursive Least Square (RLS). Bayesian Networks [6] to estimate the switching activity in VLSI circuits encapsulates all the dependencies both in internal nodes and inputs within a reasonable time and accuracies. Genetic Algorithm [7] based method for peak power estimation derives a small set of input patterns associated with peak power. However correlation between Peak Switching Frequency (PSF) and Peak Power value are not reported. PSF is used to represent the peak power consumption in VLSI circuits. All the above methods require the detailed structure of the VLSI circuit and partial simulation results. This accounts for a significant amount of time, which is proportion with the scale of integration of the circuit. Neural Network based power estimation method [8] for ISCAS Benchmark circuits is proposed using Back-Propagation algorithm with network training function named Levenberg-Marquardt (trainlm) algorithm. The regression analysis for the best case reported has slope $m = 0.915$, y-intercept $b = -0.00963$ and R-value $R = 0.994$. In this work, in order to estimate the power of ISCAS'89 Benchmark circuits, BPNN is trained using training algorithms traingdm and traingdx and the performance for the testing data set is analyzed. Network was trained with the 20 sets of input and output data sets reported in Monte Carlo method [4]. 5 set of vectors left out during the training process were used for testing.

II. BACK PROPAGATION NEURAL NETWORK

The back propagation method is a technique used in training multilayer neural networks in a supervised manner. Fig. 1 shows the structure of a BPNN.

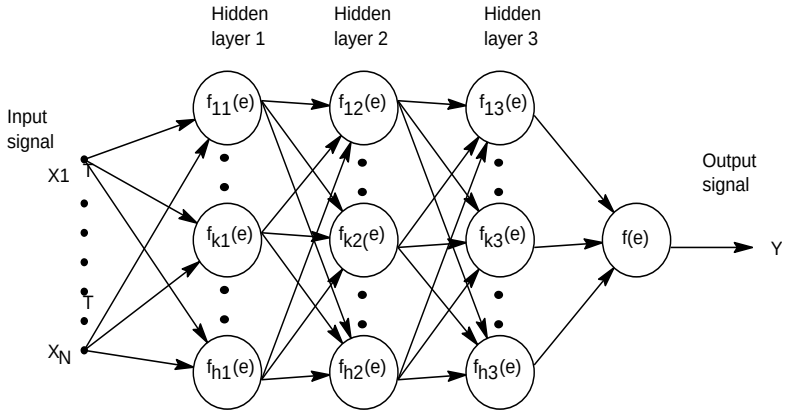

Fig. 1 Back propagation neural network

The Back Propagation (BP) method, also known as the error back propagation algorithm, is based on the error-correction learning rule. It consists of two passes through the different layers of the network: a forward pass and a backward pass. In the forward pass, an activity pattern is applied to the input nodes of the network, and its effect propagates through the network layer by layer. Finally, a set of outputs is produced as the actual response of the network. During the forward pass, the synaptic weights of the networks are all fixed. During the backward pass, the synaptic weights are all adjusted in accordance with an error-correction rule. The actual response of the network is subtracted from a desired response to produce an error signal. This error signal is then propagated backward through the network. The synaptic weights are adjusted to make the actual response of the network move closer to the desired response in a statistical sense. The weight adjustment is made according to the generalized delta rule to minimize the error. Two commonly used neuron activation functions for the neurons are sigmoidal and tansig functions. Both functions are continuously differentiable everywhere and typically have the following mathematical forms:

$$\text{Tan Sig moidal}: \quad f(e) = \frac{2}{1+\exp(-2e)} - 1, \qquad (1)$$

$$\text{Pure Linear}: \; f(e) = \beta e, \text{ for } \beta > 0 \qquad (2).$$

where a and b are constants and e is the input activation function f (e).

III. BPNN TRAINING ALGORITHMS USED FOR POWER ESTIMATION

The back propagation neural network is trained with nine different network training functions. The weight / bias learning function was choosen as 'learngdm' and the performance function was choosen as 'mse'.

III.1 Variable Learning Rate BP with Momentum

III.1.A . Traingdx

The learning rate parameter is used to determine how fast the BPNN method converges to the minimum solution. The higher the learning rate, the bigger the step and the faster the convergence. However, if the learning rate is made too high the algorithm will become unstable. On the other hand, if the learning rate is set too low, the algorithm will take a long time to converge. To speed up the convergence time, the variable learning rate gradient descent BP utilizes higher learning rate α when the neural network model is far from the solution and smaller learning rate α when the neural net is near the solution. The new weight vector W_{k+1} is adjusted in the same way as in the gradient descent with momentum, but with a varying α_k. Typically, the new weight vector W_{k+1} is defined as

$$W_{k+1} = W_k - \alpha_{k+1} g_k + \mu W_{k-1} \qquad (3)$$

$$\alpha_{k+1} = \beta \alpha_k \qquad (4)$$

Where β is 0.7 if the new error is greater than 1.04 (old error), β is 1.05 if the new error is less than 1.04 (old error), μ is the momentum factor, and g_k is the gradient.

III.1.B. Traingdm

This algorithm can be used to train any network as long as its weight, net input, and transfer functions have derivative functions. This network training function updates weight and bias values according to gradient descent with momentum. In traingdm, derivatives of performance is calculated with respect to the weight and bias variables. Each variable is adjusted according to gradient descent with momentum,

$$dX = mc * dXprev + lr * (1 - mc) * dperf / dX \qquad (5)$$

where dXprev is the previous change to the weight or bias.

IV. PROPOSED METHOD FOR ESTIMATION

IV.1 Training Process:

Step 1: 20 input vectors extracted from ISCAS'89 Benchmark circuits were used to the train the BPNN. The input vectors contain the information regarding count of the number of inputs, outputs, D flip-flops, inverters and gates present in the circuit

Step 2: Each parameter in the input vectors and their corresponding target vectors were normalized in the range -1 to +1. This is done because input vectors and output vectors have very large values and very small values.

Step 3: Normalized input vectors and their corresponding normalized target vectors were used to train the BPNN.

IV.2 Testing Process:

Step 1: 5 input vectors left out in the training process were used for testing.

Step 2: The various parameters in the input test vectors are also normalized in the same manner as that of normalization employed during training.

Step 3: Network will generate normalized outputs vectors for these normalized test input vectors.

Step 4: Normalized output vectors are converted back to their original value by applying the reverse normalization process.

Step 5: The output vectors obtained for these test inputs were compared with the expected outputs and validation was performed using Regression analysis.

V. BPNN DESIGN AND SIMULATION FOR POWER ESTIMATION

A four layer BPNN with three hidden layers and one output layer is implemented in MATLAB. The neurons in the hidden layer were varied. Activation functions namely, Pure Linear was applied to the first hidden layer and Tan-sig was applied to the rest of the hidden layers and output layer. BPNN is trained using three different training algorithms namely traingdm and traingdx. Error goal of the BPNN is fixed at 10^{-8}, because error goal below this was not reached by the network. Simulation is carried out for the two algorithms by fixing up the Learning Rate (α), momentum constant (mc) and varying the Number of Hidden Layer Neurons (HLN). Learning rate is varied in the range 0.1 to 0.8. A high Learning Rate (Lr) leads to rapid learning, but the weights oscillate. A lower learning rate leads to slower learning and hence, an optimized learning rate is used. The stopping criterion is selected using extensive MATLAB simulation. Momentum constant is varied from 0.1 to 0.9.Table 1. lists training data set for the BPNN. Table 2. lists test data set. Table 3. lists validation data for testing process by performing regression analysis. The circuits considered for testing were S344, S382, S641, S1488 and S13207. Simulations were done exhaustively for various layer sizes, learning rate, momentum constant, epochs and iterations.

TABLE 1.
TRAINING VECTOR EXTRACTED FROM ISCAS89 AND PREVIOUS WORKS [4],[8]

BENCH	IN	OUT	DFF	INV	GATE	AND	NAND	OR	NOR	MC power
S208	10	1	8	38	66	21	15	14	16	0.00698
S298	3	6	14	44	75	31	9	16	19	0.00912
S349	9	11	15	57	104	44	19	10	31	0.01856
S386	7	7	6	41	118	83	0	35	0	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
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
S1494	8	19	6	89	558	354	0	204	0	0.06018
S5378	35	49	179	1775	1004	0	0	239	765	0.23357
S9234	19	22	228	3570	2027	955	528	431	113	0.28004
S15850	14	87	597	6324	3448	1619	968	710	151	0.51991
S35932	35	320	1728	3861	12204	4032	7020	1152	0	1.22048
S38417	28	106	1636	13470	8709	4154	2050	226	2279	1.14518
S38584	12	278	1452	7805	11448	5516	2126	2621	1185	1.87987

TABLE 2.
TESTING VECTOR EXTRACTED FROM ISCAS89 AND PREVIOUS WORKS [4],[8]

BENCH	IN	OUT	DFF	INV	GATE	AND	NAND	OR	NOR
S344	9	11	15	59	101	44	18	9	30
S382	3	6	21	59	99	11	30	24	34
S641	35	24	19	272	107	90	4	13	0
S1488	8	19	6	103	550	350	0	200	0
S13207	31	121	669	5378	2573	1114	849	512	98

VI. RESULTS AND DISCUSSION

TABLE 3.: RESULTS OBTAINED FROM SIMULATION

TRAINGDM									
# I/P	Layer Size	Learning Rate	Momentum Constant	Epochs	No.of Iterations	M (slope)	B (y_intercept)	R (regression)	MSE
9	8:15:15:1	0.35	0.9	400	10	0.9506	0.0017	0.9972	0.0001352
9	9:15:15:1	0.2	0.2	400	10	0.9906	0.0027	0.9936	0.0002298
9	8:14:15:1	0.2	0.1	400	100	1.0823	5.76E-04	0.9953	0.0003767
9	8:13:14:1	0.1	0.7	400	10	0.9443	0.0054	0.9993	7.296E-05
8	9:15:15:1	0.1	0.9	400	10	1.0412	0.0027	0.9999	3.485E-05
TRAINGDX									
# I/P	Layer Size	Learning Rate	Momentum Constant	Epochs	No.of Iterations	M (slope)	B (y_intercept)	R (regression)	MSE
8	7:15:15:1	0.26	0.8	225	10	0.9753	0.0055	0.9981	8.161E-05
8	8:14:15:1	0.2	0.9	225	1	1.0117	5.45E-05	0.9967	0.0001201
8	8:17:15:1	0.108	0.9	223	1	0.9727	0.008	0.9983	0.0000968
8	8:17:15:1	0.108	0.9	224	1	1.0277	0.0061	0.997	0.0001342
8	8:13:15:1	0.6	0.9	260	1	0.9045	0.0044	0.9998	0.0001836
8	8:13:15:1	0.4	0.9	265	1	1.045	0.0051	0.9999	0.000127
8	8:15:15:1	0.4	0.6	225	10	1.0247	0.0066	0.9993	0.0001142
8	8:15:15:1	0.4	0.9	225	10	1.0148	0.0012	0.9983	6.337E-05

Fig. 2 and Fig. 3 show the regression analysis graph for the training functions traingdm and traingdx respectively

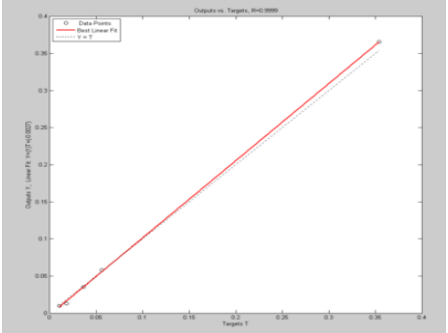

Fig. 2 Regression analysis for traingdm

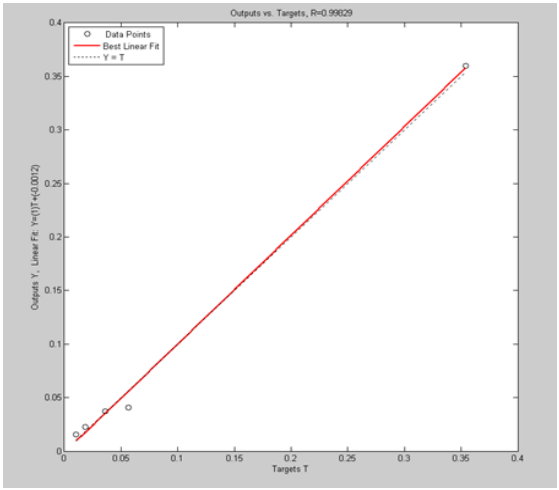

Fig. 2 Regression analysis for traingdx

TABLE 4.
COMPARISON OF PROPOSED WORK WITH EXISTING LITERATURE

Training Function	Layer Size	Number of Inputs	Epochs	Iterations	M (Slope)	(B) y-intercept	R-Value
Trainlm [1]	6-15-15-1	9	2139	NA	0.915	-0.000963	0.994
Trainlm [1]	9-15-15-1	9	879	NA	0.464	0.00882	0.753
TRANGDM	**9-15-15-1**	**8**	**400**	**10**	**1.0412**	**0.0027**	**0.9999**
TRANGDX	**8-13-15-1**	**8**	**265**	**1**	**1.045**	**0.0051**	**0.9999**

VII. CONCLUSION

For attaining good power estimation the ideal values of R, M and B are 1, 1 and 0 respectively. The proposed work achieves results that are very close to ideal values. Hence BPNN with training functions Traingdm and Traingdx are better to estimate power in sequential circuits. The mean square error (MSE) reported for the training functions in the scale of 10^{-5}, indicate that the estimation of the power for test circuits are very accurate. This method is suitable for estimating power quickly and more precisely.

REFERENCE

[1] Najm F, "Transition density : A new measure of activity in digital circuits," IEEE Transactions on Computer-Aided Design, vol.12 no.2, pp.310-323, February 1993.

[2] Xakelllis M and Najm F, "Statistical estimation of switching activity in digital circuits," 31st ACM / IEEE Design Automation Conference, San Diego, CA, pp.728-733, 1994.

[3] Burch R, Najm F, Yang P and Trick T, "A Monte Carlo Approach for power estimation," IEEE Transactions on VLSI systems, vol.1, no.1, pp63-71, March 1993.

[4] Saxena V, Najm F M and Hajj I M, "Monte Carlo Approach for Power Estimation in Sequential circuits", proceedings of ED\&TC 97, pp.416-420, 1997.

[5] Murugavel A K , Ranganathan N and Chandramouli R, "Least Squares Estimation of Average Power in Digital CMOS circuits", IEEE Transactions on VLSI Systems, pp.55-58, 2002.

[6] Bhanja S and Ranganathan N, "Switching Activity Estimation of VLSI circuits using Bayesian Networks", IEEE Transactions on VLSI Systems, pp. 558-567, 2003.

[7] Yi-Ling Liu, Chun-Yao Wang, Yung-Chih Chen and Ya-Hsin Chang, "A Novel ACO based Pattern Generation for Peak Power Estimation in VLSI Circuits," 9th International Symposium on Quality of Electronic Design, pp.317-323, 2009.

[8] Ligang Hou, Liping Zheng and Wuchen Wu, "Neural Network Based VLSI Power Estimation" IEEE Conference, pp. 1919-1921, October 2006.

