Voltage and Temperature Scalable Standard Cell Leakage Models Based On Stacks For Statistical Leakage Characterization

Janakiraman Viraraghavan
janakiram@ece.iisc.ernet.in

Bishnu Prasad Das
bpdas@cedt.iisc.ernet.in

Bharadwaj Amrutur
amrutur@ece.iisc.ernet.in

Indian Institute of Science. Bangalore, India - 560012

Abstract

With extensive use of Dynamic Voltage Scaling (DVS) there is increasing need for voltage scalable models. Similarly, leakage being very sensitive to temperature motivates the need for a temperature scalable model as well. We characterize standard cell libraries for statistical leakage analysis based on models for transistor stacks. Modeling stacks has the advantage of using a single model across many gates there by reducing the number of models that need to be characterized. Our experiments on 15 different gates show that we needed only 23 models to predict the leakage across 126 input vector combinations. We investigate the use of neural networks for the combined PVT model, for the stacks, which can capture the effect of inter die, intra gate variations, supply voltage(0.6-1.2V) and temperature($0-100^0C$) on leakage. Results show that neural network based stack models can predict the PDF of leakage current across supply voltage and temperature accurately with the average error in mean being less than 2% and that in standard deviation being less than 5% across a range of voltage, temperature.

1 Introduction

Statistical leakage analysis is gaining importance with scaling transistor dimensions. Leakage power will contribute approximately 50% of the total power in the 90nm technology node [1] and shrinking transistor sizes makes this leakage power more difficult to predict . Further, process variations i.e. variations in effective gate length, L_e, oxide thickness, T_{ox}, and threshold voltage V_{TH} can result in up to 20 $\times$ variations in the leakage of the manufactured chips [2]. Authors in [2] also show how environmental factors affect leakage.

Process Variations occur due to non uniformity in the manufacturing of the chips. These include Inter-die, Intra die and intra gate variations. Inter die variations [3] refer to variations that occur across dies, wafers or lots. The variation introduced is the same across the entire die. Intra-gate variations [7]refers to transistor to transistor variations within a gate. The impact of intra-gate variations on statistical analysis is a drastic increase in the number random variable needed to model them. Each transistor within a gate will now have one random variable per intra-gate process parameter. Intra-die variation, as the name suggests, refers to variation of a particular parameter within the die. Intra-die variations are usually spatially correlated and all transistors within a gate have the same value. In the past few years considerable amount of work has been done in building models that predict leakage accurately as function of process parameters. The empirical technique described in [3] captures the effect of inter die variations in gate length L_e and provides a simple analytical method to statistically analyze leakage but considers inter die variations only in L_e. It also does not take into account the effect of intra gate variations.

Currently industrial chips implement DVS in the range of $V_{DD}/2 - V_{DD}$ which drives the need for voltage scalable models. Further, depending on the location of a gate in the chip its operating temperature can vary from $25^0C - 120^0C$ [6] [2]. If the temperature profile is available a temperature scalable model enables more accurate analysis. We present one such model which uses Neural Networks to capture the effect of process, voltage and temperature (PVT). Authors in [8] modeled the leakage of a stack with a neural network but the extreme non-linearity in leakage due to temperature introduced too much error in the model when both voltage and temperature were included. Instead, we model *log* of the leakage current [3] and are able to include the effect of both voltage and temperature, along with process, in the same model.

Statistical leakage characterization, like static leakage characterization, involves characterizing every gate for every input vector. Each such characterization will involve a substantial number of SPICE runs. We look at the problem of reducing the number of such models by modeling different kinds of stacks present in the library.

The rest of the paper is organized as follows: section 2

1063-9667/08 $25.00 © 2008 IEEE
DOI 10.1109/VLSI.2008.38

describes how modeling leakage through stacks can be used for statistical leakage characterization. In section 3 we explain how leakage through different gates are characterized with these stacks. We then present in section 4 a neural network based leakage current model for stacks. In section 5 we present the results obtained when we tested these stack models on different gates comparing them with accurate SPICE simulations across voltage and temperature and conclude in section 6.

2 Leakage modeling

2.1 Standard Cell Library Characterization for Statistical Analysis

The intent of statistical leakage characterization is to fit the leakage of every gate, for every input vector, to a convenient functional form which is usually an exponential of a quadratic polynomial [3]. If there are N gates in the library and the i^{th} gate has M_i inputs we would require $\sum_{i=1}^{N} 2^{M_i}$ models. One way to reduce the number of models is to model the average leakage of a gate across all its input vectors. This assumes that each input vector is equally likely, which is too simplified an assumption as the probability of occurance of an input vector for a gate depends on the circuit structure. Thus, there is definitely a need for modeling each gate for every input vector. Stack modeling enables re-use of models which reduce the required number of models.

Authors in [3] model log of the leakage using a quadratic polynomial in L_e. However, they do not consider the effect of intra gate variations [7]. With intra gate variations it is expensive to consider the effect of all transistors in the gate for every input vector. Depending on the input vector being considered, some transistors may not not affect the leakage statistically. Stack modeling helps in this respect too.

2.2 Modeling stacks

It is well known that the leakage of a gate is primarily determined by the stacks through which the current flows. Considerable amount of work has been done in trying to find closed form expression for the leakage current through stacks [4]. The authors in [4] considers the case of an NMOS transistor being turned off only when its gate is grounded. But an NMOS transistor on the top of a stack trying to pass V_{DD} will turn off as soon as its source potential rises to $V_{DD} - V_{TH}$ hence we need to extend the idea in [4] to estimate the leakage across all input vectors. However, we use the ideas mentioned in [4] to identify the transistors on a stack that affect the leakage in a statistical sense. [8] uses this concept to model leakage for a few CMOS gates across all input vectors. The key ideas in [8] for using the stack model to predict the leakage of any gate across all input vectors can be summarized as

- An NMOS/PMOS transistor trying to pass a 0/1 is completely turned on and hence can be treated as a short circuit [4]
- An NMOS/PMOS transistor trying to pass a 1/0 turns off as soon as its source potential increases to $(V_{DD} - V_{TH})/(V_{TH})$ and hence is turned off even though its gate is connected to (V_{DD}/GND)
- In an N transistor NMOS/PMOS stack, when there is exactly ONE NMOS/PMOS transistor whose gate is connected to GND/V_{DD} i.e. the gates of other $N-1$ transistors are connected to V_{DD}/GND, the leakage will increase significantly as this transistor moves towards the output due to DIBL [8].
- When the leakage through a logic gate is predicted using a stack, the leakage obtained from the stack model has to be scaled by the ratio of effective widths of the transistors in the gate and those on the stack
- Leakage due to parallel stacks simply add up

Using the above mentioned stack rules we built models for the 18 commonly found stacks listed in the first column of Table 1. These stack models are the most basic stacks and are widely used. We assume that the standard cell library does not have gates with stack size greater than 4 as the increased logical effort will affect the delay of the circuit. For the gates tested in [8], any input combination will result in a set of stacks which are a subset of the list we have modeled in Table 1. In this paper we look at a few more gates that have different kinds of stacks. The naming convention for the commonly used stacks are as follows.

All model names in Table 1 are of the form, {Stack type}{Stack size}/{Input to the stack}

- Stack type indicates if it is an NMOS stack or a PMOS stack
- Stack size is the number of transistors on the stack
- Input is the decimal value of the input vector being applied to the stack with the LSB of the input vector applied to the transistor closest to the output
- The widths of the transistors on the stack are the unit inverters transistor widths scaled by the stack size

Consider an example of a three input majority gate shown in Fig 1. Apart from the stacks listed in Table 1 we need to model four more stacks to handle such gates. The leakage of this gate is determined by three different stacks, whose currents are denoted as I_1, I_2, I_3, and the total leakage is given by the sum of these three currents. Let us examine the input vectors that result in the use of these different stacks.

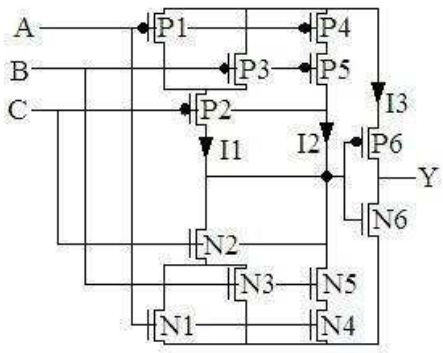

Figure 1. Three input majority gate

1. **Input vectors (000/111)**: When the input is 000/111 PMOS/NMOS transistors {P1-P5}/{N1-N5} are completely turned on. N6/P6 is also turned on as the output of the majority gate is 0/1. I_2 is determined by the leakage through the stack {N4,N5}/{P4,P5} and can be estimated using the {n2/0}/{p2/3} model. Similarly I_3 can be estimated using the {p1/1}/{n1/0} model. I_1 is primarily determined by the stack {N1, N2, N3}/{P1,P2,P3}. The stack formed by {N1 N2 N3}/{P1, P2, P3} cannot be approximated by any of the stack models listed in Table 1 because of the parallel combination of N1/P1 and N3/P3 in series with N2/P2. Thus we need two more models.

   Input vectors (001/110): The only difference between the previous input vector case and this case is that the input to C is 1/0. Thus I_2 and I_3 are identical to the previous case. I_1 is now determined by the stack {N1 N2 and N3}/{P1, P2, P3} with the input to N2/P2 being 1/0. As mentioned earlier, N2/P2 cannot be treated as short circuits as NMOS/PMOS transistors cannot pass 1/0 fully. This again requires two more stack models to model these two states.

   Other Input vectors: Any of the other four input vectors will result in a set of stacks listed in Table 1. Similar to the explanation for the NAND4 gate in [8] we can deduce which of the models in Table 1 are needed for each input vector.

Similarly, we had to model one more PMOS stack for the two input XOR gate to accurately predict the leakage when the input vector 0. However for the inverters and AND/OR gates we could use the stacks models listed in Table 1 to predict the leakage of all their input states.

While static leakage characterization needs to be done for every gate for every input vector, it is more efficient to characterize different kinds of stacks[1] for statistical leakage characterization rather than characterize every gate for every input vector.

[1]The list of different stacks is specific to the library

Table 1. List of the common stack models used across all gates with Neural Network Training Details. Training time(**Time**), maximum testing error (**Err**), and number of hidden nodes(**H**)

Model	Size	Type	Input	Err(%)	Time(s)	H
n1/0	1	NMOS	0	7.03	3.16	9
n2/0	2	NMOS	00	4.39	9.70	13
n2/1	2	NMOS	01	4.02	9.36	13
n3/0	3	NMOS	000	7.76	16.17	17
n3/1	3	NMOS	001	5.43	21.92	17
n3/3	3	NMOS	011	7.20	9.45	17
n4/0	4	NMOS	0000	3.65	52.11	21
n4/1	4	NMOS	0001	4.61	49.26	21
n4/7	4	NMOS	0111	3.09	6.01	21
p1/1	1	PMOS	1	4.77	8.65	9
p2/3	2	PMOS	11	4.01	7.66	13
p2/2	2	PMOS	10	3.10	2.3	13
p3/7	3	PMOS	111	4.17	64.45	17
p3/6	3	PMOS	110	6.10	27.73	17
p3/4	3	PMOS	100	3.26	5.26	17
p4/15	4	PMOS	1111	4.37	65.70	21
p4/14	4	PMOS	1110	9.78	12.95	21
p4/8	4	PMOS	1000	6.31	5.42	21

3 Deriving leakage of a logic gate from stack leakage

All static CMOS gates can be broken down into elementary stacks and hence characterization of all gates in the standard cell library will only involve mapping of their different leakage states to the corresponding stacks that cause the leakage. Thus, the leakage currents predicted by the elementary stack models are analogous to basis vectors in linear algebra. We have demonstrated the idea with a small subset of 15 gates in the standard cell library. However, by scanning the entire library and modeling the different stacks that appear in it, leakage current through any gate, for a given input vector can then be written as linear weighted sum of the currents through these stacks. If M is the total number of unique stack models present in the library and S_j is the leakage through the j^{th} stack, the leakage through the i^{th} gate in a circuit for a given input vector q_i can be written as

$$X_i^{q_i} = \sum_{j=1}^{M} \alpha_{ij}^{q_i} S_j \quad (1)$$

Where S_j is the leakage through the j^{th} stack model and $\alpha_{ij}^{q_i}$ is the scaling factor due to difference in effective widths

between the stack used in the gate and the stack model. $\alpha_{ij}^{q_i}$ = 0 if the j^{th} stack does not appear for that gate for the input vector q_i.

In certain applications the primary input is known apriori. For example, in the idle state, a combinational circuit is set to a particular input state which results in least leakage. Equation 1 can be used in such cases. However in many other applications, the input is not known and the state of the input becomes probabilistic [10]. In such cases, Eqn. 1 needs to be modified to accommodate the probability of an input vector occurring. This can be easily done in our framework. Let the probability of occurance of the input q_i for the i^{th} gate be $p_i^{q_i}$. The average leakage of the i^{th} gate can be written as [10]

$$X_i = \sum_{\forall q_i} p_i^{q_i} X_i^{q_i} \tag{2}$$

Substituting Eqn. 1 we get

$$X_i = \sum_{j=1}^{M} \alpha_{ij}^{avg} S_j \tag{3}$$

Where, $\alpha_{ij}^{avg} = \sum_{\forall q_i} p_i^{q_i} \alpha_{ij}^{q_i}$, is the average scaling across all input vectors. Thus, this formulation can be modified very easily for such applications as well. To test our formulation we need to use a model for these elementary stacks. In the next section we investigate the feasibility of using neural networks to model the leakage through these stacks.

4 Leakage modeling - Neural Networks

Existing leakage models capture the effect of process on leakage for a given supply voltage (V) and temperature (T) and these models are indexed by voltage and temperature. Also for (V, T) values not present in the look interpolation is used. We can do away with the look up table based approach, to account for voltage and temperature variations, if we have a unified PVT model. Thus, there is a clear need for a model which can capture the effect of Process (Inter die and Intra gate), Voltage and Temperature on leakage all in one model. Fitting log of the leakage to a second order polynomial, of process parameters alone, does work well when all possible cross terms are considered. A PVT quadratic polynomial model failed to fit the leakage when temperature was also considered and hence we tried out Artificial Neural Networks (ANN) for the combined model. Authors in [8] used ANN to model the sub-threshold leakage of a logic gate. But the exponential dependence of leakage on PVT introduces too much non-linearity and hence the model gives unacceptable error. Instead, in this paper we are able to model the effect of PVT on leakage by modeling *log* of the leakage current. The exact algorithm to train an ANN to model the leakage through a stack is shown in Fig2.

1. Generate N(1500) training samples for the P random input parameters from their distributions (Inter die and intra gate process parameters) - Produces an $N \times P$ matrix, $\mathbf{X_P}$
2. Divide the temperature range ($0 - 100^0 C$) and voltage range (0.6 - 1.2V) into N equally spaced samples
3. Randomly pair the i^{th} temperature point with j^{th} voltage point - Produces an $N \times 2$ matrix $\mathbf{X_{VT}}$
4. Append matrix $\mathbf{X_{VT}}$ to the matrix $\mathbf{X_P}$ columnwise - Produces an $N \times (P + 2)$ matrix $\mathbf{X} = [\mathbf{X_P} \quad \mathbf{X_{VT}}]$
5. Repeat steps $1 - 4$ for another M(500) testing samples
6. Simulate the stack, to be modeled, using SPICE with the $N + M$ (2000) samples to obtain the corresponding leakage current values $\mathbf{Y_N^{SPICE}}$ and $\mathbf{Y_M^{SPICE}}$
7. Normalize the input and output according to equation $X_{NORM} = X/[max(X) - min(X)]$
8. Initialize the Neural network
9. Train the ANN with N *normalized* training samples according to the back propagation algorithm [5]
10. Feed the ANN with the M testing samples and obtain the outputs predicted by the ANN. Produces Y_M^{ANN}
11. Evaluate $\Delta = Max\{\frac{|Y_M^{ANN} - Y_M^{SPICE}|}{Y_M^{SPICE}}\}$
12. IF $\Delta < \delta$ (0.1) *Increase the number of hidden nodes by ONE* and GO TO Step 8
13. ELSE ANN model is trained

Figure 2. Algorithm to train the ANN model - In brackets we have indicated values used in this work

Steps 1 - 6 generate the necessary training and testing samples which are obtained through SPICE simulations. As indicated we had to do 2000 SPICE simulations to successfully train a stack. Step 1 generates all the process parameter values from their distribution. Step 2 and 3 generate the necessary temperature and voltage samples. Our ANN model is expected to predict the leakage accurately at any supply voltage and any temperature in the ranges we train the network with. Having considered a temperature range of $0 - 100^0 C$, with 1500 training samples, we divide the entire range in uniform steps of $0.067^0 C$. Similarly the voltage range of (0.6V - 1.2V) is divided in steps of 0.4mV. In step 3 we randomly pair these uniformly generated points in order to make sure that the training set contains as many different V,T pairs as possible. The testing set has a similar but smaller number of such points which in our case was 500. The reason we have to consider so many points is the extreme non linearity introduced by temperature. Step 6 creates the training and testing output data set through SPICE simulations. In step 7 we normalize both the input and output data set in order to improve the training. In step 8 we train the ANN through standard techniques as given in [5]. We used the Neural Network tool box provided by MATLAB for this purpose. We only had to choose the initialization parameters and train the network. The standard practice in Neural Networks is to train the network until the mean square error of one epoch is below the chosen threshold [5]. We then validate the trained network with a disjoint

testing data set in step 9 and compute the maximum percentage error between the actual SPICE value and the one predicted by our *trained* ANN. If the error is above the chosen threshold, which in our case was 10%, we have to increase the number of hidden nodes and re-train the network. However, among the stacks listed in Table 1 we did not have to do this retraining even once. This kind of re-training is a common procedure with neural networks [5].

5 Results

We trained the ANN to model the leakage through the different stacks listed in col 1 of Table 1 and the 5 extra models for the majority gate and the xor gate. We then used them to predict the PDF of the different gates listed in Table 2. Each model captures the effect of Process (inter die[1] and intra gate variations) in L, T_{OX} and V_{TH}, supply voltage (0.6 - 1.2V) and temperature ($0 - 100^0C$). All process parameters were sampled from Gaussian distributions with 3σ = 10% of their mean. Table 1 shows the details of training the network. From the table we observe that the maximum training time amongst all models is around 66 seconds. For each gate in col 1 of Table 2 we have listed the mean and standard deviation error, compared to SPICE, for the input vector 0 at a supply voltage of 0.9V and an operating temperature of 50^0C. All SPICE simulations were done with HSPICE using an industrial 130nm model file. From Table 2 we see that the error for most gates is less than 1% while the error for the the xor gate is 5%. This is a result of the stacking approximation. The maximum standard deviation error across all the gates across all input vectors is around 20%(not shown in Table 2) for the NAND4 gate with its input vector set to $14(1110_2)$. We see that the model used to predict the leakage of a NAND4 gate with an input vector 14 is the n1/0 model, which is nothing but a single NMOS OFF transistor. As per the stack approximation the bottom three transistors are treated as short circuits. This assumption is not completely true. The voltage drop across the three ON transistors, albeit very small, does affect the leakage of the top most OFF transistor due to body effect. Similarly the error is maximum for the NAND3 and NAND2 gates when the n1/0 model is used to predict their leakage. Since the number of transistors on the stack progressively decrease from NAND4 to NAND2 the error drops from 20% for a NAND4 gate to 10% for a NAND2 gate. Similar results are observed for NOR gates as well. However, the average mean error and average standard deviation error, for a gate, across all input vectors, are less than 2% and 5% respectively. A library consisting of an inverter, 2-4 input NAND-NOR-AND-OR gates, an XOR2 gate and a three input majority gate, needed just 23 stack models to predict the leakage across 126 leakage states. The idea of using stacks as elementary models being independent of the model used, we tried out the conventional exponential polynomial approach before using the ANN.

[1] Intra gate variations make sense only when a gate is placed in a circuit. For an isolated gate level model, both inter die and intra die variations are one and the same

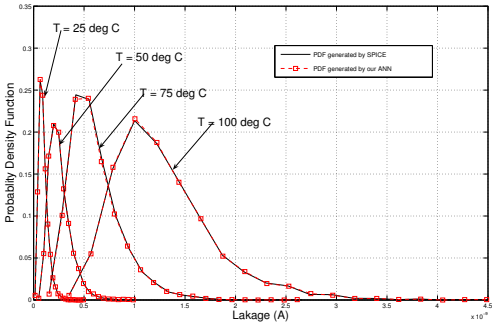

Figure 3. Temperature scalable plots: PDF from our model, for NAND4 gate, follows SPICE closely across different temperatures

We first tried to model log of the leakage with a quadratic polynomial, in the process parameters alone, for a four transistor NMOS stack which has 3 inter die parameters and $3 \times 4 = 12$ intra gate parameters. The model did work well but quadratic polynomial will have total of 15 first order terms and $15 \times 8 = 120$ second order terms. To fit this polynomial with 135 coefficients we require at least 135 SPICE simulations. We actually needed 150 SPICE simulations to fit the leakage accurately, using the least square fit algorithm, and 50 disjoint SPICE simulations to test it. Thus if we want to use this exponential polynomial model and index it by voltage and temperature, in voltage steps of 50mV and temperature steps of 25^0C, we would need 48 different exponential quadratic models in a voltage range of 0.6-1.2V and $0 - 100^0C$. This requires $150 \times 48 = 7200$ SPICE simulations for a four transistor NMOS stack. However, our ANN model required just 2000 SPICE simulations. Further, the natural ability of the ANN to interpolate accurately also gives us the freedom to use it any (V, T). Since our ANN model is both voltage and temperature scalable, we also show that it can predict the leakage across different voltages and temperatures. The plot in Fig 3 shows how accurately the ANN model is able to predict the PDF of the leakage across a range of temperatures $(25 - 100^0C)$ for a NAND4 gate. Similarly, Fig 4 shows a voltage scalable plot for an XOR2 gate at 0.6V and 1.2V. Both figures also show the exact PDF obtained through MC SPICE simulation.

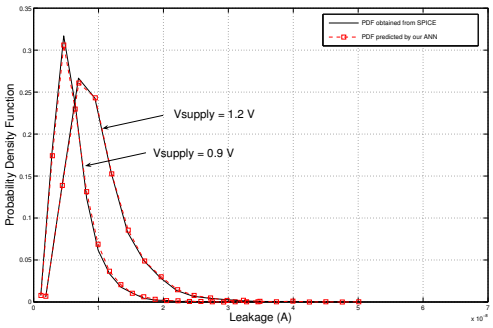

Figure 4. Voltage scalable plots: PDF from our model, for an XOR2 gate, follows SPICE closely across different voltages

6 Conclusion

Modeling stacks greatly helps in reducing the number of models required to characterize a standard cell library for statistical analysis. Once all kinds of stacks present in the library have been modeled, characterization of a gate, across all its input vectors, only involves mapping each leakage state to the corresponding stack models that cause the leakage. With 23 stack models we were able to predict the leakage for 15 gates in the standard cell library across 126 input vector combinations. We used Neural Networks to model the leakage through stacks as conventional techniques failed to capture the effect of process, temperature and voltage together. It was found that neural networks were able to model the leakage through stacks taking into account variations in process (both inter die and intra gate), voltage (0.6-1.2V) and temperature($0 - 100^0C$) making it a suitable voltage and temperature scalable model. The average mean error across all gates across all input vectors was less than 2% and the average standard deviation error was less than 5%.

Table 2. Mean and Std Dev Error between SPICE and our ANN model for different gates. V = 0.9V and T = 50^0C Input = 0.

Gate	$\mu_{SPICE}{}^a$ (pA)	$\Delta\mu$ (%)	σ_{SPICE} (pA)	$\Delta\sigma$ (%)
nand4	263.08	0.1	118.87	0.1
nand3	272.90	0.0	129.96	0.1
nand2	316.88	0.2	168.02	0.7
nor4	3234.38	0.1	1505.08	0.2
nor3	2402.75	0.1	1161.32	0.2
nor2	1605.88	0.1	825.47	0.3
and4	1029.09	0.1	537.64	0.3
and3	1036.30	0.1	579.48	0.4
and2	1104.27	0.1	611.00	0.4
or4	3999.14	0.1	1710.71	0.1
or3	3149.84	0.0	1372.48	0.1
or2	2364.97	0.0	1072.01	0.0
xor2	6127.25	3.1	2889.81	5.1
inv	815.60	0.1	492.99	0.2
maj3	1605.56	0.0	724.25	0.2

[a] $\mu_{SPICE}/\sigma_{SPICE}$ - Actual mean/std dev from SPICE. $\Delta\mu/\Delta\sigma$ % error in mean/ std dev between SPICE and our ANN

References

[1] Intel Corp. http://www.intel.com/cd/ids/developer/asmona/eng/strategy/182440.htm?page=2.

[2] Sherkar Borkar et al. *Parametric variations and impact on circuits and microacrchitecture. In Proc. DAC 2003*.

[3] R. Rao, A. Srivastava, D. Blaauw and D. Sylvester. *Statistical analysis of sub-threshold leakage current for VLSI circuits IEEE. Trans. Very Large Scale, 12(2);131-139, 2004.*

[4] Z. Chen, M. Johnson, L. Wei and K Roy. *Estimation of standby leakage power in CMOS circuits considering accurate modeling of transistor stacks In ISLPED '98: Proceedings of the 1998 international symposium on Low power electronics and design, pages 239-244. New York, NY, USA, 1998. ACM Press.*

[5] Simon Haykin *Neural networks a comprehensive foundation, 2 edition, Prentice-Hall Engineering, USA, 1998*

[6] H.Su, E. Acar and S. R. Nassif. *Full chip leakage estimation considering power supply and temperature variations In ISLPED '03: Proceedings of the 2003 international symposium on Low power electronics and design, pages 78-83. New York, NY, USA, 2003. ACM Press*

[7] K. Okada, K.Yamaoka and H.Onodera (2003). *A Statistical Gate-Delay Model considering Intra-gate Variability,. ICCAD, pp.908-913, 2003*

[8] Janakiraman. V et al. *Leakage modelling of logic gates considering the effect of input vectors. In press VDAT, 2007.*

[9] Hongliang Chang and Sachin Sapatnekar. *Full chip analysis of leakage power under process variations, including spatial correlations In Proc of DAC 2005*

[10] E. Acar et al. *Leakage and Leakage Sensitivity Computation for Combinational Circuits In ISLPED '03: pages 96-99. New York, NY, USA, 2003. ACM Press*