Voltage and Temperature Scalable Logic Cell Leakage Models Considering Local Variations Based On Transistor Stacks

Authors

1) Janakiraman Viraraghavan*
email: jramaanv@gmail.com, janakiram@ece.iisc.ernet.in
Electrical and Communication Department
Indian Institute of Science
Bangalore - 560012
Ph: +919449752143
2) V. Visvanathan
email: vish@ti.com
Texas Instruments India
3) Bharadwaj Amrutur
email: amrutur@ece.iisc.ernet.in
Electrical and Communication Department
Indian Institute of Science
Bangalore - 560012
Ph: +918022933172

Voltage and Temperature Scalable Logic Cell Leakage Models Considering Local Variations Based On Transistor Stacks

Abstract

We propose a logic gate leakage model based on transistor stacks, which includes local transistor level process variation parameters along with global process variation parameters and supply and temperature. The stack models include both subthreshold as well as gate leakage and consider the input vector state. We examine cells from an industrial standard cell library and find that most cells can be modeled with simple stacks, which have a linear chain of transistors. However some gates like XOR, Majority or Muxes need complex stacks and we show how these can be modeled. Our experiments show that only 18 different stack models are needed to predict the leakage of all gates in this industrial library. Re-use of the same models for pass transistor logic circuits and multi-finger transistors is also demonstrated. We explicitly include voltage and temperature into the models to support joint estimation of power supply IR drops and leakage currents, as well as enable analysis for dynamic voltage scaling applications. We use artificial neural networks to create unified models which include global and local process variations, supply voltage in the range of $V_{DD}/2$ - V_{DD} and temperature in the range $0 - 100^0C$. These models are very useful for performing statistical leakage analysis of large circuits. Results from the ISCAS'85 benchmark circuits show that neural network based stack models can predict the PDF of leakage current of large circuits across supply voltage and temperature accurately with the average error in mean being less than 2% and that in standard deviation being less than 7% when compared to SPICE.

Index Terms

Leakage, Model, Statistical, Neural, Training, Stack

Voltage and Temperature Scalable Logic Cell Leakage Models Considering Local Variations Based On Transistor Stacks

I. Introduction

LEAKAGE power is approximately 50% of the total active power in the 90nm technology node [1] and is expected to remain a significant fraction of the total active power in scaled technologies [2]. Hence it important to accurately estimate the total leakage current of a digital circuit, not only to estimate the chip's power, but also to properly design the power grid of the chip in order to ensure that the performance targets are met [3]. Hence it has become imperative to accurately model leakage currents for digital circuits.

Leakage currents depend exponentially on certain process and environment parameters, like effective gate length, L_e, oxide thickness, T_{ox}, threshold voltage V_{TH} and supply and temperature. Hence even small variations in these show up as a large variation in the leakage current, with Borkar et.al. in [4] reporting up to 20$\times$ variations in the leakage of manufactured chips.

Process parameter variations consist of both die-to-die and within-die variations. Die-to-die (or inter-die or global) variations affect all transistors within a die in the same way and are modeled as a single random variable which takes on the same value for each gate in the chip. Within-die (or intra-die or local) variations affect different gates within the same chip differently. These can have two components: a spatially correlated component and a random component. For spatially correlated variations, gates within a small region get affected identically. This effect is modeled by breaking the die into many regions, with one random variable per region [5]. The random variable takes on the same value for all the gates within the region. Random local variations affect each gate independently. In fact their origin can be traced to random fluctuations within each transistor's gate length (LER), oxide thickness (OTV) and

threshold voltage due to random dopant fluctuations (RDF) [6], [7]. Atomistic simulations in [7] show that these fluctuations become quite significant in technologies below 35nm, with standard deviation of threshold voltage reaching 100mV for a nominally sized transistor in a 9nm process node. Hence an accurate analysis in 45nm and lower process nodes needs to incorporate the effect of these random local variations. Recent work [8], [9] has shown that lithography inaccuracies result in random non rectangular gate(NRG) structures which introduce further variations in the threshold voltage of the transistors. They also suggest efficient techniques to model the effects of these atomistic variations in circuit simulators like SPICE.

Existing leakage analysis frameworks use a simple model for a logic gates leakage consisting of an exponential of a linear or quadratic polynomial in the parameters (length, oxide thickness, threshold voltage) [5], [10], [11]. However, these models use a single random variable per gate to model the impact of local variations, ignoring the fact that random local variations due to LER, OTV and RDF are a transistor level phenomenon and need to be considered per transistor within the gate. Lumping local variations at the gate level works well for spatially correlated local variations, but leads to significant errors in the presence large random local variations. Hence we examine the inclusion of these per-transistor local variation parameters in the gate level leakage models. The main challenge is to handle the large number of parameters to model the leakage of a gate.

Leakage estimation considering power supply and temperature variations requires leakage models to include supply and temperature [12]. Existing leakage models explicitly model the dependence of leakage on either the process (global and spatially correlated local parameters) or the supply and temperature for a given set of process parameters. We propose a leakage model based on artificial neural networks (ANN) to capture the dependence of leakage on global, spatially correlated local, and random per-transistor process parameters like L_e, V_{TH}

and T_{OX} as well as explicitly include the supply voltage and temperature. This will enable voltage and temperature aware statistical leakage analysis. ANNs ability to model large amounts of non-linearity enables one to use these models in process nodes less than 65nm where the random local variability is expected to increase significantly [7]. ANNs can also handle large ranges in supply and temperature which enables analysis of chips that implement dynamic voltage scaling(DVS).

Accurate treatment of input vector dependence requires a separate model per input vector of the gate. However as proposed in [5], we only need to model transistor stacks which can then be reused across many gates. We extend the work of [5] by covering a larger variety of logic gate structures which exist in a standard cell library like pass-gate based cells, multi-finger transistors etc. and show how stack based models suffice to model the entire library. We further show how transistor gate tunneling leakage currents can also be included in these stack models.

The large number of parameters introduced due to the consideration of random local variations at a per-transistor level, can be mitigated somewhat by including the parameters of only the OFF transistors of a stack along the lines of [13]. However we improve upon the accuracy of their approach by incorporating the transistor variation parameters of even weakly inverted transistors, which were considered as ON and ignored in [13]. We validate our models with SPICE and demonstrate a few applications of voltage and temperature scalable statistical leakage analysis on some benchmark ISCAS'85 circuits.

The rest of the paper is organized as follows: Section II reviews the existing leakage models. Section III describes how modeling leakage through stacks can be used for leakage characterization of different kinds of gates. Following that we explain how leakage through different gates is obtained with these stacks. We then present, in section IV, a neural network based leakage current model for stacks followed by various gate level results obtained when

we tested these stack models on different gates comparing them with accurate SPICE simulations across a range of voltage, temperature and technology nodes. We then demonstrate the accuracy of our model, in section VI at a circuit level by using it to predict the mean and standard deviation of leakage of the ISCAS'85 benchmark circuits and comparing with SPICE and conclude in section VII.

II. Leakage models - Review

The leakage(I^v_{LEAK}) of a logic gate for an input vector v can be formally expressed as:

$$I^v_{LEAK} = \sum_{s \in LeakageComponents} I^v_s e^{f^v_s(\Delta P^G, \Delta P^L, V_{DD}, T)} \tag{1}$$

Here Leakage Components refer to the sub-threshold, gate tunneling and diode components of leakage [14]. I^v_s are the nominal leakages for each component. ΔP^G are the deviations in the global parameters which are usually $\{L^G_E, V^G_{TH}, T^G_{OX}\}$, i.e. a global deviation in channel length, threshold voltage and oxide thickness, which is the same for all the transistors in all the gates. ΔP^L captures the deviations due to local fluctuations and consists of the set of channel length, threshold voltage and oxide thickness, $\{L^t_E, V^t_{TH}, T^t_{OX}\}$, with one per gate. V_{DD} and T are the local supply and temperature of the logic gate and are usually not explicitly included in the model.

The authors in [5], [10], [11], [15], [16] propose exponential polynomial models for the subthreshold leakage which includes the effect of local variations at the logic gate level. For example Rao et. al. in [15] express leakage of a gate as

$$I_{SUB} = I_{SUB,NOM} e^{-\frac{L_g + c_2 L_g^2 + c_3 V_g}{c_1}} e^{-\frac{L_l + \lambda_2 L_l^2 + \lambda_3 V_l}{\lambda_1}} \tag{2}$$

Here, (L_g, V_g) are zero mean random variables representing the global offset in length and threshold voltage and (L_l, V_l) represent the random local offsets at the gate level. While the

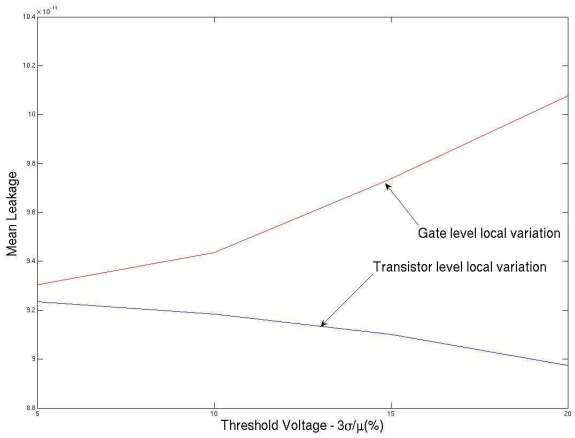

Fig. 1. Comparison of existing(gate level local variation models) and exact transistor level local variation model for a NAND4 gate. Existing local variations over estimate mean leakage for transistor stacks with the error increasing with increased local variations

model is satisfactory for handling global variations, the gate level local offset variables need to capture the effects of random local variations of every transistor within the logic gate, clearly leading to a loss in fidelity.

Consider a NAND4 gate with its input set to 0. We did two different SPICE simulations:- One with gate level local variation model(GLV) and the other with transistor level local variation model(TLV). On the lines of [7] we lumped the impact of LER, OTV and RDF into the V_{TH} parameter and varied ($\frac{3\sigma}{\mu}$) of local variations from 5% to 20% and measured the mean leakage in each case. Shown in Fig 1 is the mean leakage obtained from the two methods. As can be seen from the graph the GLV model used in all existing leakage SLA frameworks is pretty accurate as long as the $\frac{3\sigma}{\mu}$ of local variations is less than 10%. However, mean leakage error increases as local variations increase which will be the case with future technologies. Extrapolating from Fig 1 we see that the mean leakage can be over estimated by as much as 30-40% by the GLV model as compared to the actual TLV model. An intuitive explanation for why the GLV model fails for increased local variations is as follows. On a transistor stack, the weakest transistor determines the leakage. Hence, leakage through the stack can increase only if the threshold voltage of all transistors on the stack decrease. Due to the correlation assumption, the GLV model allows this with high probability while in reality

it occurs with very low probability(since the probabilities multiply). The error introduced by the GLV model decreases with the number of transistors on the stack and is identical to the TLV model for an inverter since there is only one transistor on the stack.

Recently Gu et. al. in [17] have derived an expression for the effective threshold voltage of parallel stack of transistors considering the effect of RDF. They consider the case of "wide"transistors implemented as multiple fingers and derive accurate expressions for the PDF of the leakage with variations introduced by RDF. While this model works for a parallel stack, extending this to a series stack is not straight forward due to complicated nature of the stacking effect [13]. In this paper we over come this problem by explicitly modeling the dependence of leakage of a stack on the process parameters of each transistor. The exact technique of modeling will be explained in section IV.

Su et. al. in [12] propose iterative techniques to solve for leakage currents in the presence of IR drops. A similar iterative procedure is followed for the temperature profile as well. Such a solution requires temperature and voltage to be explicitly included in the leakage model. The authors propose the model in Equation 3 for small deviations in voltage and temperature from the nominal. Since the leakage depends exponentially on the voltage and temperature, this equation will be valid for only very small deviations in either of them. We explore the feasibility of handling much larger ranges, simultaneously including global and random local process parameters, to support DVS applications.

$$\frac{I_{LEAK}(\Delta V, \Delta T)}{I_{LEAK}(0,0)} = 1 + \sum_{i=1}^{2}\sum_{j=1}^{2} a_{i,j}\Delta T^i \Delta V^j \tag{3}$$

As shown in Table I, the leakage current of a logic gate can change by an order of magnitude depending upon the input vector. Hence separate models are required for different input vectors. However by modeling leakage currents through transistor stacks [13], [18], the

TABLE I
NAND4 LEAKAGE

$Input$	$I_{SUB}(pA)$	$Input$	$I_{SUB}(pA)$
0000	**88.0**	0001	123.3
0010	124.1	0011	204.2
0100	124.0	0101	205.2
1000	124.0	1001	205.1
0110	210.4	0111	591.6
1010	210.3	1011	602.4
1100	210.3	1101	621.5
1110	**719.5**	1111	519.7

number of required models can be reduced drastically, by reusing the same models across different gates by appropriate scaling [5].

However, there are no details presented in [5] on the impact of DIBL and the V_{TH} drop across the top most NMOS transistor on the stack, as well as how to handle pass-gate structures and multi-finger transistors.

Consideration of per transistor local parameter variation in the model will increase the model complexity, which can be mitigated by only considering OFF transistors in a stack as in [13]. Here they propose an analytical expression for subthreshold leakage through transistor stacks for the case without process variations. They reduce the model complexity by only considering the OFF transistors in the stack. They consider a transistor to be OFF, only if its gate input is 0 (for NMOS and V_{DD} for PMOS). However even an NMOS trying to pass a logic high(V_{DD}) with its gate connected to V_{DD} is weakly inverted and hence OFF if its source rises to $V_{DD} - V_{TH}$ Thus even such transistor's local process parameter variations needs to be explicitly considered in the model. In this paper, we adopt the approach of [13] to model only OFF transistors in the series stack, but expand the model to consider global and local parameter variations of each OFF transistor in the stack, including even the weakly inverted ones.

Transistor gate tunneling leakage, though a smaller component, can also be modeled by considering the stacking effect [18]. The technique proposed by Yang et. al. in [18] models

gate tunneling leakage through an isolated transistor in different configurations and express the gate tunneling leakage of a stack in terms of these elementary models. To account for stacking effect they make an approximate estimate of the intermediate node voltages and scale the gate tunneling leakage of each transistor accordingly. We adopt a similar approach in our model, by incorporating the gate tunneling leakage of the stacks chosen for subthreshold leakage modeling. We however avoid making approximate estimates of intermediate gate voltages, thus enabling us to model the gate tunneling leakage more accurately. We believe our method is also easily extensible to include the much smaller diode leakages, though we do not consider it in this paper.

III. Leakage modeling based on transistor stacks

We model the sub-threshold leakage through the transistor stack as in Equation 1, where the process parameters of only the OFF transistors are considered. For these, $\Delta P^G = \{L_E^G, V_{TH}^G, T_{OX}^G\}$, i.e. a global (and spatially correlated local) deviation in channel length, threshold voltage and oxide thickness, which is the same for all the transistors in the stack. ΔP^L captures the deviations due to local fluctuations and consists of the set of channel length, threshold voltage and oxide thickness, $\{L_E^t, V_{TH}^t, T_{OX}^t\}$, one per each OFF transistor t in the stack. We also explicitly include supply voltage and temperature in the stack's model. We call a transistor, a *statistical* transistor if its local parameters have a large influence on the leakage current of the gate. A statistical transistor is necessarily in the OFF state, but not all OFF transistors are *statistical*. For example an OFF transistor in parallel with an ON transistor will not contribute much to fluctuations in leakage current of the gate. Any transistor which is not a *statistical* transistor is labeled as a *static* transistor. Thus a leakage model needs to only consider the local parameters of all the *statistical* transistors.

However, unlike sub-threshold leakage, gate leakage happens through any transistor which

has a large potential difference between the gate and the other nodes. Thus even ON transistors need to be considered for gate leakage. Hence we extend the model of a transistor stack to also include the gate leakage in each stack.

A simple stack consists of a linear chain of transistors. However in a standard cell library, the OFF state leakage current can go through complex stacks, where there could be multiple parallel OFF transistors embedded within an otherwise simple linear sequence of transistors. We investigate the use of stack models for four major classes of standard cells namely,

- Simple stacks for CMOS gates like NAND-NOR
- Complex stacks in CMOS gates like Majority gates
- Complex stack for Pass transistor based gates like multiplexers, flops etc
- Complex stacks for Multi finger transistor based standard cells

We will first explain the concepts for simple stacks using a 4-input NAND gate as an example. We will then address complex stacks for gates like Majority gates as well as pass transistor logic. We will defer the discussion on multi-finger transistors to the appendix.

A. Simple Stacks - Sub threshold leakage

Consider a four input NAND gate shown in Fig 2. We will examine how its leakage can be predicted with elementary stacks across all input vectors by examining the effect of each vector in detail.

1) Input vector(0000): In this state all four NMOS transistors are turned OFF while all four PMOS transistors are turned ON and are as good as short circuits [13]. Thus to predict the probability density function(PDF) of the leakage of a NAND4 gate for the input vector 0000 we need to model a four transistor NMOS stack, which we will refer to as *n4/0*. The “0”in the model name refers to the input vector applied to the stack. In our notation, the transistor closest to the output is the LSB(least significant bit) and the one closest to V_{DD}/gnd is the

MSB.

2) Input vectors (1000 / 0100 / 0010): Here the top most transistor, $N4$, has its gate connected to gnd and the one of the other 3 NMOS transistors have their gates connected to V_{DD}. At least one PMOS transistor is completely turned on in all three cases and hence forces the other three parallel PMOS transistors to behave as static transistors. Similarly the one NMOS transistor that has its gate connected to V_{DD} behaves as a short circuit and can be treated as a static transistor. Thus to predict the leakage for this set of input combinations we need to model a 3 transistor NMOS stack with all inputs grounded i.e. we need to model an *n3/0*. However the width of the transistors on the 3 transistor stack is thrice the unit width while the width of the transistors on four input gate is four times the unit width. Thus we have to scale the currents by an appropriate factor to account for this width difference. The scaling factor cannot be (4/3) since the effective widths are lesser than the actual widths. The question is how to we accurately obtain the scaling factor? If the effective width of a transistor on a four transistor stack is $W_{eff}^{(4)}$ and that on a three transistor stack is $W_{eff}^{(3)}$ the scaling factor is $\frac{W_{eff}^{(4)}}{W_{eff}^{(3)}}$. It is well known that the leakage of a transistor is directly proportional to its effective width. With nominal process settings if the leakage of a four transistor stack is $I_{nom}^{(4)}$ and that of a three transistor stack in $I_{nom}^{(3)}$ then the scaling factor($\frac{W_{eff}^{(4)}}{W_{eff}^{(3)}}$) equals $\frac{I_{nom}^{(4)}}{I_{nom}^{(3)}}$.

Note that the same *n3/0* model will be used to predict the leakage of a NAND3 gate with input 000 as well with the difference being in the scaling factor which is unity.

3) Input vector (0001): Here again all PMOS transistors can be treated as static transistors. With the gate of the top most transistor $N4$ connected to V_{DD}, it is trying to pass V_{DD} and it will *turn off* as soon as its source charges up to $V_{DD} - V_{TH}$. We are thus left with a case where the bottom three transistors are turned off as their gates are grounded and the top most transistor is turned off even though its gate is connected to V_{DD}. Hence we need to model a four transistor NMOS stack with the gate of the top most transistor connected to V_{DD} which

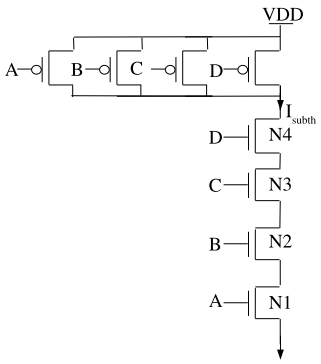

Fig. 2. Four input NAND gate

we will call *n4/1*.

4) Input vectors (1100 / 1010 / 0110): As in [13] the two transistors with their gates connected to V_{DD} can be treated as short circuits and hence as static transistors which reduces the NMOS stack to a 2 transistor stack *n2/0*. Accounting for the difference in the widths of the 2 transistor NMOS stack and the NAND4 gate we can scale the currents predicted the n2/0 model to predict the NAND4 leakage for these input vector combinations. Here again note that the same n2/0 model will be used to predict the leakage for the NAND3 gate with input 100 and NAND2 gate with input 00.

5) Input vectors (1001 / 0101 / 0011): Just as in the 0001 case we observe that the top most transistor is turned off even though its gate is connected to V_{DD}. Hence we have to have a model *n3/1* with appropriate scaling to predict the leakage for this combination.

6) Input vectors (0111 / 1011 / 1101): The leakage current is dominated by a single transistor whose gate is connected to ground. We would expect the model n2/1 to predict the leakage for these input combinations. But unfortunately the DIBL effect on the transistor whose gate is grounded is different in all three cases and hence there is a significant deviation in the PDF predicted by the n2/1 model. As can be seen from Fig 2, transistor N1 dominates the leakage in the 0111 case. Let the drain source voltage across it be V_{DS1}. In the 1011 case N2 dominates the leakage, let the drain source voltage across N2 be V_{DS2}. Similarly the leakage

in the 1101 case is dominated by N3 and let the drain source voltage across N3 be V_{DS3}. In the 0111 case all four transistors are OFF and the V_{DD} drop is across all four transistors. In the 1011 case the bottom transistor, N1, is ON and is essentially a short circuit and hence the V_{DD} drop is across three transistors (N2, N3 and N4), similarly, in the 1101 case the V_{DD} drop is across N3 and N4 alone. Hence $V_{DS1} < V_{DS2} < V_{DS3}$, which implies that the effect of DIBL increases as the OFF transistor moves up the stack i.e. $I_{0111} < I_{1011} < I_{1101}$ as can be seen from Table I. We, therefore, need a separate model for each of these three input vectors.

7) Input vectors (1110 / 1111): With the input set to 1110 the bottom three transistors(N1, N2 and N3) can be considered as short circuits and hence the leakage will be predicted by an $n1/0$ model with appropriate scaling. Note that the voltage drop across N1 - N3 can be pretty significant and can introduce some error but this is not too much of a problem and will be dealt with in greater detail in the results section. With the input set to 1111, N1-N4 are all turned ON and are static transistors. Hence the leakage is determined by the parallel PMOS stack which can be predicted with a $p1/1$ model.

B. Simple Stacks - Gate Leakage

As explained above we have identified a set of stacks needed for accurate sub-threshold leakage modeling and we now examine how these stacks can be made use of for accurate gate leakage estimation. The crucial difference between sub-threshold leakage and gate leakage is that for the former the ON transistors don't leak any extra current as they are in series with the other OFF transistors in the stack while for the latter ON transistors are in parallel and hence their gate leakage also have to be considered. Thus we can express the gate leakage through any logic gate as the sum of the gate leakages through the stacks needed for sub-threshold leakage estimation and the gate leakages through other ON transistors. However

an ON transistor can have different source/drain node potentials depending on is location within the stack and hence may have significantly different gate leakages depending on its position in the stack. We will examine a few input vector combinations of the NAND4 gate to explain this.

1) Input vector(0000/0001): The sub-threshold stacks are n4/0 and n4/1 respectively. While generating the leakage numbers for the sub-threshold leakage current, we can also generate the gate leakage numbers for these stacks. However we need to account for the additional gate leakage of the PMOS transistors which are completely turned ON (Fig 3(D)). In the 0000 case we have all four PMOS transistors contributing to the gate leakage while in the 0001 case only three PMOS transistors contribute to the gate leakage.

2) Input vector(1000/ 0100/ 0010): The sub-threshold stack is n3/0. Thus the total gate leakage is the sum of three components

- The leakage of the the n3/0 stack, scaled by the ratio of effective widths
- The gate leakage of the 3 PMOS transistors which are completely turned ON
- The gate leakage of the NMOS transistors which is completely turned ON.

In the 1000 case, N1 is completely turned ON and the voltage at its drain and source is very nearly 0. Also the width of N1 is four times that of the unit width transistor thus the gate leakage of N1 is obtained by scaling the gate leakage of the unit transistor in Fig 3(B) by the ratio of effective widths. In the 0100 case the first two components of gate leakage are the same as the 1000 case. However, in the 0100 case, the NMOS transistor that is turned ON is N2 and its drain/source potential is greater than 0. Apart from scaling the leakage of the unit transistor in Fig 3(B) by the ratio of effective widths we also have to scale it so as to account for the difference in the drain/source voltage also.

Yang et. al. in [18] make approximate estimates of these node voltages and use the sensitivities to the drain/source potential to obtain the leakage current accordingly. They make

such estimates for each transistor on the stack. Though we could have used a similar method to estimate the leakages of the remaining ON transistors, we use a simple linear scaling factor to account for the difference in drain/ source potential. The scaling factor is a function of the intermediate node voltage which will vary with process parameters. However, we found this ignoring the variation in the scaling factor gives acceptable results. The reason is the following. Our method partitions the leakage into two components namely the gate leakage component through the stack and the gate leakage component through the other ON transistors. The leakage through the stack is modeled as a whole where the stacking effect is implicitly taken care off and the approximation needs to be made only for the remaining ON transistors whose drain/source is at non zero potential. For a NAND4 gate shown in Fig 2 only eight of the sixteen input combinations result in ON transistors on the stack with non zero drain/source potential. Even among these eight cases, only the gate leakages of N2 and/or N3 need to be approximated (since N4 always part of a stack model and N1 is either off when its part of a stack model or its ON with its drain/source at ground potential). Thus with the gate leakage of the other six transistors estimated accurately, the error introduced by two transistors alone don't show up significantly. This is evident from the results presented in section V. This is an improvement over the technique proposed by Yang et. al. in [18] since we err in estimating the leakage of fewer transistors on the stack.

Thus gate leakage of all CMOS gates can be predicted accurately with elementary stack models and four other gate leakage models shown in Fig 3.

From the above discussion gate leakage estimation can be summarized as

- Identify the sub-threshold stack and estimate the gate leakage for that stack
- Other transistors in the in the logic gate that have significant gate leakage have to be one of the four configurations in Fig 3.
- Scale the gate leakage of other transistors by the necessary scaling factor to account for

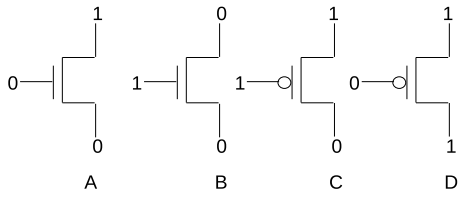

Fig. 3. Different configurations for gate leakage

the width difference and the drain potential difference.

C. Simple Stacks in a Standard Cell Library

From the above discussions it is clear that we need to identify the suitable sub-threshold leakage stack for a gate for a given input vector. The gate leakage model for such a stack can then be easily obtained as explained above and added to the sub-threshold leakage model. We can formulate the following rules to identify the right stack

- An NMOS/PMOS transistor trying to pass a logic 0/1 is completely turned on and hence can be treated as a short circuit [13]
- An NMOS/PMOS transistor trying to pass a logic 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 [19].
- 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 II. 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.

The naming convention for the commonly used stacks are as follows.

All model names in Table II 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 - **n** for NMOS and **p** for PMOS
- 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

Here the widths of the transistors on the stack are the unit inverters transistor widths which is then scaled by the stack size

Simple stacks are not sufficient to model all the gates in the library. We will next consider modeling of complex stacks.

D. Complex stacks - Majority Gate

Consider an example of a three input majority gate shown in Fig 4. As will be explained below, apart from the stacks listed in Table II, we need to model four more stack configurations to handle such gates. The leakage of the majority gate in Fig 4 is determined by three different stacks, whose currents we denote 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.

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

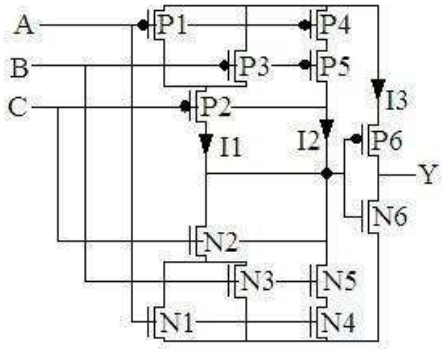

Fig. 4. Three input majority gate

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 II because of the parallel combination of N1/P1 and N3/P3 in series with N2/P2. Thus we need two more models.

2) 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.

3) Other Input vectors: Any of the other four input vectors will result in a set of stacks listed in Table II. Similar to the explanation given earlier for the NAND4 gate we can deduce which of the models in Table II are needed for each input vector.

The above example illustrates two facts

- The stacks that need to be modeled are library dependent
- When a new CMOS gate is added in the library, even if the present set of stacks is not sufficient to capture the leakage across all inputs, we can identify the new stack pattern

TABLE II
LIST OF THE COMMON STACK MODELS USED ACROSS ALL GATES WITH NEURAL NETWORK TRAINING DETAILS. NUMBER OF LOCAL PARAMETERS(N_L), NUMBER OF GLOBAL PARAMETERS(N_G), NUMBER OF INPUTS(N_I), NUMBER OF HIDDEN NODES(N_H), NUMBER OF COEFFICIENTS IN THE INPUT LAYER(N_{Coeff}^{Input}), NUMBER OF COEFFICIENTS IN THE OUTPUT LAYER(N_{Coeff}^{Output}). THESE DETAILS ARE COMMON TO THE 130NM AND 45NM SUB-THRESHOLD LEAKAGE AND GATE LEAKAGE MODEL

Model	**Size**	**Type**	**Input**	N_L	N_G	N_I	N_H	N_{Coeff}^{Input}	N_{Coeff}^{Output}
n1/0	1	NMOS	0	3	3	8	9	81	10
n2/0	2	NMOS	00	6	3	11	13	156	14
n2/1	2	NMOS	01	6	3	11	13	156	14
n3/0	3	NMOS	000	9	3	14	17	255	18
n3/1	3	NMOS	001	9	3	14	17	255	18
n3/3	3	NMOS	011	9	3	14	17	255	18
n4/0	4	NMOS	0000	12	3	17	21	378	22
n4/1	4	NMOS	0001	12	3	17	21	378	22
n4/7	4	NMOS	0111	12	3	17	21	378	22
p1/1	1	PMOS	1	3	3	8	9	81	10
p2/3	2	PMOS	11	6	3	11	13	156	14
p2/2	2	PMOS	10	6	3	11	13	156	14
p3/7	3	PMOS	111	9	3	14	17	255	18
p3/6	3	PMOS	110	9	3	14	17	255	18
p3/4	3	PMOS	100	9	3	14	17	255	18
p4/15	4	PMOS	1111	12	3	17	21	378	22
p4/14	4	PMOS	1110	12	3	17	21	378	22
p4/8	4	PMOS	1000	12	3	17	21	378	22

and still model the leakage accurately while re-using the existing models for most of the other input vectors.

Similar to the majority gate, we had to model one more PMOS stack for the two input XOR gate(implemented as a CMOS circuit) to accurately predict the leakage when the input vector is 0. For the inverters, 2-4 input NAND/NOR and AND/OR gates we could use the stacks models listed in Table II to predict the leakage across all their input states.

E. Complex Stacks for Pass transistor logic - Multiplexer

All multiplexers and flip flops are implemented with pass transistors. We look at the example of a two to one multiplexer(two inputs and one select input) and investigate the sub-threshold leakage paths from V_{DD} to gnd for each of the eight input vectors(two inputs and one select line). The schematic of a 2:1 mux is shown in Fig 5. As we can see the output is the same as the input(A) when the select line(S) is 0 and the output is B when the select line is 1. We observe from Fig 5 that the voltage at the intermediate node Y_b is well defined at all times since either of the pass transistors is turned ON. Thus for any given input vector combination, only one of the pass transistors can leak current **if** the potential difference across that pass

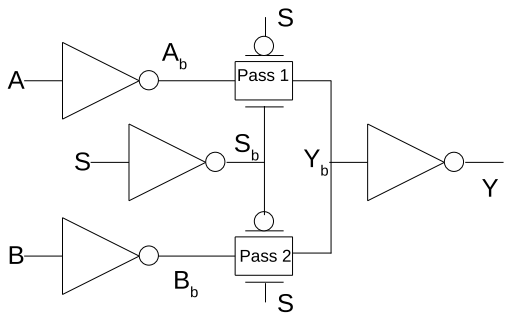

Fig. 5. Two to one multiplexer

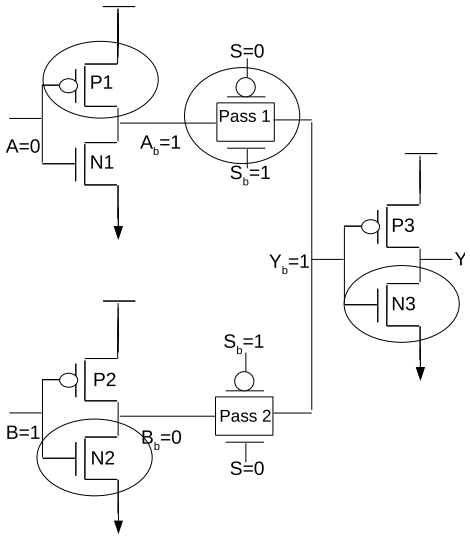

Fig. 6. Two to one multiplexer with S=0, A=0, B=1. Circled transistors are completely turned ON. Select line inverter is not shown

transistor is V_{DD}. This point will be made clear by looking at some examples in greater detail. Also note that the leakage through the inverters can be estimated with the $n1/0$ model or the $p1/1$ model depending on whether the input to it is 0 or 1 respectively.

1) S=0, A=0, B=0: We see that the pass gate (pass1) is turned on and is a short circuit. $A_b = Y_b = V_{DD}$ similarly $B_b = V_{DD}$. Clearly the potential difference across the pass2 is zero and hence there is no additional leakage through pass2. Thus the total leakage for this input combination is just the sum of leakage currents through the four inverters which can be predicted with the n1/0 and the p1/0 models.

2) S=0, A=0, B=1: Here again pass1 is turned on and is a short circuit and $A_b = Y_b = V_{DD}$. However B_b is at ground potential. The potential difference across pass2 is V_{DD} which will result in additional sub-threshold leakage current. The question we need to answer is, which stack model will appropriately model the leakage through the pass transistor. Shown in Fig 6 is the transistor equivalent of the multiplexer with the inputs S=0, A=0 and B=1. The inverter for the select line is not shown for brevity but the leakage through this inverter is independent of the other circuitry. For this input combination, the transistors that are turned on completely, and hence are as good as short circuits, are circled in Fig 6. Note that the leakage current that flows through the pass transistor, *pass2*, has to originate at V_{DD}, flow through P1 then through *pass1* followed by *pass2* then through N2 and finally to ground. Note that at the node Y_b we assume that the current coming from pass1 does not flow into the gates of transistors P3 and N3 but instead flows through pass2 completely. This is a valid assumption as sub-threshold leakage current is the sum of sub-threshold leakages of isolated NMOS and PMOS transistors which is much greater than the gate leakage. With this explanation we can easily see that there are four leakage paths from V_{DD} to ground for the circuit shown in Fig 6.

- Through N1 of the inverter for the input A
- Through P2 of the inverter for the input B
- Through P3 of the inverter at the output
- Through the pass transistor pass2

The leakage of the mux will also include the leakage through the select line inverter which is not shown in the figure. The leakage through the pass transistor can be further broken down into a parallel NMOS transistor leakage and a PMOS transistor leakage. Thus we see that even though we have to consider the leakage through the pass transistor we did not have to model a new stack. Thus the elementary stack models are re-used even in such pass transistor based logic gates. The above discussion can be summarized as follows

1) The voltage levels at both ends of a pass transistor in gate are either at V_{DD} or ground. This implies that both nodes of a pass transistor have low resistance paths to V_{DD} or ground.
2) Sub-threshold leakage current can flow through a pass transistor only if one end is at V_{DD} and the other is at ground

Similar explanations can be offered for other input vector combinations as well.

Among the different types of gates, we have analyzed all but one namely the multi finger transistor gates. Analyzing multi finger gates requires some empirical results and hence will be discussed in the appendix. We now look at how leakage of a logic gate can be derived in terms of the leakage through the elementary stacks.

F. 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. 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, S_j^{SUB} and S_j^{GATE} are the sub-threshold and gate leakages through the j^{th} stack, the leakage through the i^{th} gate in a circuit for a given input vector v can be written as

$$X_i^v = \sum_{j=1}^{M} (\alpha_{ij}^v S_j^{SUB} + \beta_{ij}^v S_j^{GATE}) \tag{4}$$

Where α_{ij}^v is the sub-threshold leakage scaling factor due to difference in effective widths between the stack used in the gate and the stack model. α_{ij}^v = 0 if the j^{th} stack does not

appear for that gate for the input vector v. Similarly β_{ij}^{v} is the scaling factor for the gate leakage.

In certain applications the primary input is known a priori. For example, in the idle state, a combinational circuit is set to a particular input state which results in least leakage. Equation 4 can be used in such cases. However, in many other applications, the input is not known and the state of the input becomes probabilistic [21]. In such cases, Eqn 4 needs to be modified to accommodate the probability of an input vector occurring. This can be easily done in our formulation. Let the probability of occurrence of the input v for the i^{th} gate be p_i^v. The average leakage of the i^{th} gate can be written as [21]

$$X_i = \sum_{\forall v} p_i^v X_i^v \tag{5}$$

Substituting Eqn. 4 we get

$$X_i = \sum_{j=1}^{M} \alpha_{ij}^{avg} S_j^{SUB} + \beta_{ij}^{avg} S_j^{GATE} \tag{6}$$

Where, $\alpha_{ij}^{avg} = \sum_{\forall v} p_i^v \alpha_{ij}^v$, is the average scaling across all input vectors for sub-threshold leakage and β_{ij}^{avg} is the average scaling for the gate leakage . Thus, this formulation can be modified very easily for such applications as well.

In the next section we investigate the feasibility of using neural networks to model the leakage through these stacks. The details have been explained with sub-threshold leakage as example. The same explanations hold for gate leakage as well.

IV. Leakage modeling - Neural Networks

Existing leakage models [10], [11], [15], [22] capture the effect of process on leakage for a given supply voltage (V) and temperature (T). These fit the log of the leakage to a polynomial - usually a quadratic [10]. However such an approach doesn't work very well when all the cross terms between the process parameters are considered. Incorporating (V)

and (T) into the model to support large ranges in these, further enhances the difficulty of getting simple polynomials to work. Instead we propose to model the log of the leakage current ($f_s^v(\Delta P^G, \Delta P^L, V_{DD}, T)$) using an artificial neural network (ANN). These are well known to fit highly non-linear but continuous functions well [23]. For example, Beyene in [24] used an ANN to model various characteristics of the received waveform at the end of a lossy interconnect wire. They show that the ANN is a good choice to model the amplitude of the received waveform, which is a highly non-linear function, and simple polynomial models fail to accurately capture the dependence on the process parameters.

A. Structure of the ANN

The ANN generally has an input and output layer along one or more hidden layers [23]. The number of hidden units in each hidden layer is also a design choice and we shall investigate how we can choose these. It is well know from the theory of neural networks [23] that an ANN with a single hidden layer can approximate any functional continuous mapping from one finite dimensional space to another, provided the number of hidden units is sufficiently large. The structure of a feed forward ANN with a single hidden layer is shown in Fig 7. There are N_I inputs and N_H hidden units in the hidden layer. The output z_j of each hidden layer is given by

$$z_j = \phi(\sum_{i=1}^{N_I} w_{ij} x_i + w_{0j}) \tag{7}$$

Where w_{ij} is the weight from the i^{th} input to the j^{th} hidden layer which has a bias w_{0j}. The function $\phi(.)$ is called the activation function. In this paper we have chosen the $tanh()$ as the activation function. The output y of the ANN is given by

$$y = \sum_{j=1}^{N_H} \alpha_j z_j + \alpha_0 \tag{8}$$

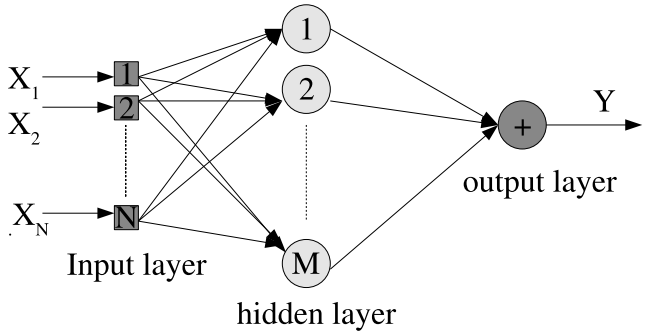

Fig. 7. A feed-forward neural network with a single hidden layer

The choice of the number of hidden units in the hidden layer is not as straight forward and is iteratively obtained in the training process.

B. Training and Testing the ANN

The weights of the ANN w_{ij} and α_j of the ANN are to be learnt by providing appropriate training samples to the ANN. It is well known that random sampling [23] of the input space is a good technique to obtain training samples. Once the network is trained we need to test it with samples that were not presented to the ANN while training. This is known as generalization. The exact algorithm to train and test an ANN to model log of the leakage through a stack is as follows(in brackets we have indicated values used in this work).

1) Generate N(1500) training samples for the P random input parameters from their distributions (Global and local 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 $\mathbf{X_P}$ column wise - 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 [23]
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 $\Delta > \delta$ (0.15) *Increase the number of hidden nodes by ONE* and GO TO Step 8 ELSE ANN model is trained

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. In this work we have considered global and local variations in L_e, V_{TH} and T_{OX}. 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. For example in the 130nm technology node we considered a temperature range of $0 - 100^0C$ and a voltage range of 0.6V - 1.2V. With 1500 training samples, we divide the entire temperature range in uniform steps of 0.067^0C. 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 with the Levenberg Marquardt algorithm as given in [23]. 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 [23]. 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 15%, we have to increase the number of hidden nodes and re-train the network. However, we found that we did not have to do this retraining even once. With this theory of stacks and neural networks we will now look at the results in detail.

V. Gate Level Results

A. Gates with simple stacks

1) Sub-threshold leakage in 130nm: We trained the ANN to model leakage through the simple stacks listed in column 1 of Table II. We then used them to predict the PDF of the different gates. Results for the basic NAND-NOR gates are listed in Table III. Each model captures the effect of Process (global and local) 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 [1] with 3σ = 10% of their mean.

For each gate in col 1 of Table III we have listed the mean and standard deviation error, compared to SPICE, for the input vector with minimum error and the one with maximum

[1]Note that the ANN does not assume any particular functional form for the PDF of process parameters and hence can be used even if actual process data from obtained from a fab is not some well known distribution. However, due to lack of actual data from the fab we have assumed independent Gaussian distributions for all process parameters

TABLE III
SUMMARY OF RESULTS FOR ELEMENTARY NAND-NOR GATES - 130NM
MEAN ERROR ($\Delta\mu$)AND STD DEV ERROR ($\Delta\sigma$) BETWEEN SPICE AND OUR ANN MODEL FOR DIFFERENT GATES. V = 0.9V AND T = 50^0C

Gate	Input	Model	$\Delta\mu$ (%)	$\Delta\sigma$ (%)
nand4	0	n4/0	0.08	0.12
	14	n1/0	10.71	20.67
nand3	0	n3/0	0.04	0.14
	6	n1/0	9.19	17.70
nand2	0	n2/0	0.16	0.67
	2	n1/0	4.74	10.66
nor4	15	p4/15	0.04	0.04
	1	p1/1	7.22	9.48
nor3	7	p3/7	0.02	0.02
	1	p1/1	6.27	8.97
nor2	3	p2/3	0.06	0.25
	1	p1/1	4.35	5.94
inv	0	n1/0	0.09	0.19
	1	p1/1	0.12	0.36

error. The results in Table III are for 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.

From Table III we see that the maximum standard deviation error across all the gates, across all input vectors, is around 20% for the NAND4 gate with its input vector set to $14(1110_2)$. As per our stack approximation rules 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 since 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 as its source voltage changes.

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.

The results presented till now were compared with SPICE simulations with a 130nm **industrial** model file in which gate leakage is not very prominent. We now look at the performance

TABLE IV
SUMMARY OF SUB-THRESHOLD LEAKAGE RESULTS FOR THE NAND-NOR GATES IN PTM 45NM. V_{DD}=0.5V, T=25^0C
MEAN ERROR ($\Delta\mu$)AND STD DEV ERROR ($\Delta\sigma$) BETWEEN SPICE AND OUR ANN MODEL.

Gate	$\Delta\mu$ **(%)**	$\Delta\sigma$ **(%)**
nand4	3.32	3.38
nand3	2.56	3.06
nand2	1.59	1.89
nor4	0.09	0.19
nor3	0.35	0.79
nor2	0.33	3.47

of the ANN model with the Predictive Technology Model files [25] in 45nm technology for both sub-threshold leakage and gate leakage

2) Sub-threshold Leakage - 45nm PTM: Results for sub-threshold leakage are exhaustively presented with an industrial 130nm model. However we present a few sub-threshold leakage results on a 45nm PTM as well for the sake of completeness. We modeled the 2-4 transistor NMOS/PMOS stacks and used them to predict the sub-threshold leakage of NAND and NOR gates. The results are summarized in Table IV. As can be seen from the table the mean error and standard deviation error is quite negligible even in 45 nm technology.

3) Gate Leakage - 45nm PTM: We modeled the gate leakage of all simple stacks with the 45nm predictive technology model and used them to predict the gate leakage of a NAND4 gate across all input vectors and compared them with accurate SPICE simulations. The results are summarized in Table V. As can be seen from Table V the mean and standard deviation error increases as the NMOS transistor that is ON moves up the stack (inputs like 2, 3, 4, 5 and 6). As mentioned before this error is due to the inaccurate estimation of intermediate node voltages. When the NMOS transistors at the bottom of the stack are turned ON(inputs like 8, 12, 14 and 15) the model predicts the leakage accurately since the intermediate node voltages are estimated accurately. Thus the maximum error in mean and standard deviation are around 20% for the input 3 (0011) which seems quite high but the average mean error and standard deviation error for a NAND4 gate are 2.93% and 4.41% respectively. This clearly indicates that our model fares well across most input vectors and hence is acceptable.

TABLE V
SUMMARY OF GATE LEAKAGE RESULTS FOR THE NAND4 GATE IN PTM 45NM. V_{DD}=0.5V AND T=25^0C
μ, σ(SPICE MEAN AND STD DEV) MEAN ERROR ($\Delta\mu$)AND STD DEV ERROR ($\Delta\sigma$) BETWEEN SPICE AND OUR ANN MODEL.

Input	μ (pA)	$\Delta\mu$ (%)	σ (pA)	$\Delta\sigma$ (%)
0	411.09	0.04	360.78	0.15
1	193.95	0.00	161.27	0.15
2	742.46	4.43	483.10	9.37
3	151.96	20.57	133.21	20.29
4	1056.98	6.24	671.78	8.11
5	742.31	0.77	490.78	18.06
6	1247.62	5.78	787.30	5.16
7	107.35	0.18	94.19	1.23
8	1627.60	1.62	1282.96	0.53
9	1444.82	0.75	1275.58	1.97
10	2177.16	2.32	1532.46	0.86
11	1326.18	1.06	1100.39	1.07
12	2853.73	1.70	2009.71	0.09
13	2650.38	0.65	1955.86	1.06
14	4173.23	0.42	2818.82	1.26
15	4937.69	0.42	3275.25	1.13

B. Gates with complex stacks

Optimized transistor level implementations of complex gates lead to new stack patterns. as discussed in section III-D. There we considered two such gates, the two input XOR shown in Fig 8 and the three input majority gate shown in Fig 4. Table VI summarizes the result for these gates. The last column in the table indicates the number of new stack models needed to model the leakage for that given input vector. As we can see with the addition of two different gates (12 different leakages states) we could re-use seven of the existing stack models listed in Table II. The stack models needed for the majority gate were explained in section III-D. On similar lines we see that the XOR gate needs one new stack model. When the binary input of the XOR gate is set to 11, we see that the $\{$P2, P3, P4, P5$\}$ stack in Fig 8 determines the leakage. Note that this configuration of PMOS transistors cannot be broken down into two separate 2 transistor PMOS stacks since the drains of P3 and P5 are shorted. Thus by modeling this complex PMOS stack separately we can estimate the leakage of the XOR gate for the input 11 accurately.

TABLE VI
SUMMARY OF RESULTS FOR THE XOR AND MAJORITY GATES - 130NM
MEAN ERROR ($\Delta\mu$)AND STD DEV ERROR ($\Delta\sigma$) BETWEEN SPICE AND OUR ANN MODEL. NUMBER OF NEW STACK MODELS NEEDED(# NEW MODELS)

Gate	Input	$\Delta\mu$ (%)	$\Delta\sigma$ (%)	# New Models
XOR	0	3.11	5.11	0
	1	1.62	3.71	0
	2	1.66	3.86	0
	3	0.06	0.11	1
Majority	0	0.04	0.27	1
	1	0.16	0.37	1
	2	2.01	4.80	0
	3	3.77	5.96	0
	4	3.73	8.51	0
	5	2.00	3.30	0
	6	0.02	0.41	1
	7	0.03	0.06	1

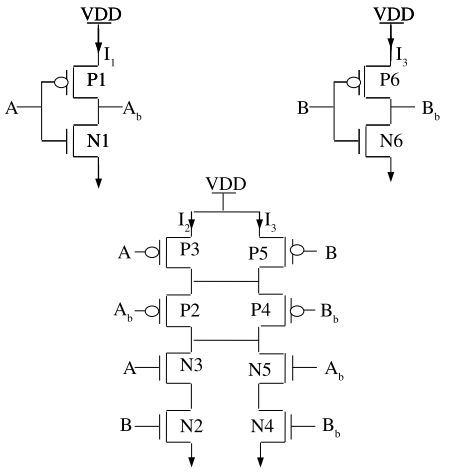

Fig. 8. Two input XOR gate

C. Voltage and Temperature Scalability

Results in Table III shows the performance of our ANN model across all gates and different input vectors but at a supply of 0.9V and temperature of 50^0C. The ANN model is both voltage and temperature scalable, we show a few examples of its prediction across different voltages and temperatures. The plot in Fig 9 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 10 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 simulations which almost exactly overlaps with the predicted PDF showing that our model performs well over a large temperature and voltage range.

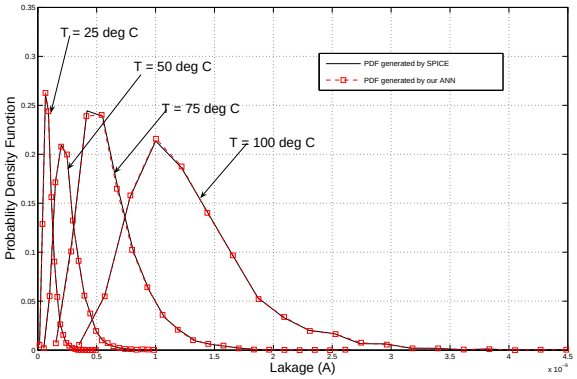

Fig. 9. Temperature scalable plots: PDF from our model, for NAND4 gate, follows SPICE closely across different temperatures - 130nm

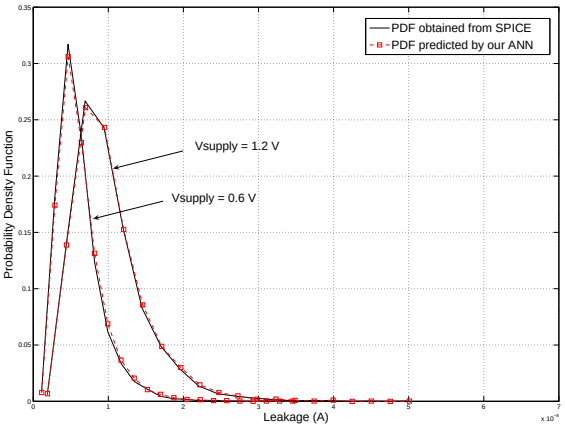

Fig. 10. Voltage scalable plots: PDF from our model, for an XOR2 gate, follows SPICE closely across different voltages - 130nm

D. Comparison with exponential polynomial model

As a comparison with the exponential polynomial model, 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 global parameters and $3 \times 4 = 12$ local 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

TABLE VII
SUMMARY OF RESULTS FOR A 2:1 MUX. $V_{DD} = 0.9V$, T = 25^0C
MEAN ERROR ($\Delta\mu$)AND STD DEV ERROR ($\Delta\sigma$) BETWEEN SPICE AND OUR ANN MODEL.

Select	Input	μ (nA)	$\Delta\mu$ (%)	σ (nA)	$\Delta\sigma$ (%)
S=0	0	1.26	0.58	0.61	0.57
	1	1.80	0.35	0.77	0.51
	2	1.85	0.34	0.83	0.43
	3	1.23	0.57	0.56	0.49
S=1	0	1.23	0.57	0.59	0.61
	1	1.83	0.31	0.82	0.30
	2	1.82	0.35	0.86	0.89
	3	1.20	0.54	0.59	0.42

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 for any (V, T).

E. Pass Transistor Logic

We used the elementary stack models in Table II to predict the PDF of the leakage of each of the eight leakage states of the 2:1 mux. The results are summarized in Table VII. We see from the table that the stack approximation works well across all inputs with the maximum error in mean being less than 0.6% and that in standard deviation being 0.61%. This explanation can be carried over to the mux based flops as well since flops are generally implemented as multiplexers with the output fed back to one of the inputs.

F. Results from a industrial standard cell library

We examined all the cells for an industrial standard cell library in the 130nm process. We needed to use just 18 Simple stacks listed in Table II to model the leakage of all 119 gates in the library. The gates in the library consisted of simple NAND-NOR type gates to complex Majority gates. All gates were implemented in the static CMOS logic and there were 53 such gates. Other 66 gates included pass transistor logic based circuits like multiplexers, flip flops and xor gates.

VI. Circuit Level Simulation Results

A. Benchmark Results

Having modeled gates accurately we now see how the model performs in terms of speed and accuracy with large circuits. We tested our models with the ISCAS'85 benchmark circuits by comparing our results with accurate SPICE simulations. The ISCAS'85 circuits are made from a library with the following gates:- Inverter, 2-9 input NAND, NOR, AND OR and two input XOR gates. All circuits are available in verilog format. All gates with 5 or higher inputs were broken down into smaller gates. We then converted the verilog code into SPICE code with one model file per transistor in the circuit. This was done to allow simulations with both global and local variations. As mentioned earlier we considered global and local variations in L_e, V_{TH} and T_{OX}, each drawn from a Gaussian distribution with their 3σ being 10% of the mean. We simulated the circuit with 15000 Monte Carlo simulations and obtained the actual sub-threshold leakage values from which we evaluated the actual mean and std dev. All circuits were simulated using HSPICE with an industrial 130nm high speed model file at a temperature of 25^0C, supply voltage of 0.9V and the primary input set to 0. To obtain the leakage of the circuit from our stack models we need the input at each gate in the circuit. Since the ISCAS'85 circuits are available in verilog form we simulated the circuit with the primary input set to 0 and obtained the input state of each gate in the circuit. Given the gate and its input we know which stack model(s) to use for the gate. We then do a Monte Carlo with our ANN model and obtain the leakage values of the circuit by adding the leakage obtained from each gate/stack.

Table VIII summarizes the results obtained with the ISCAS'85 circuits. Note that the number of gates reported here is after the larger gates have been broken down into smaller gates. We see that the error in mean and standard deviation across all gates is less than 4% and 12% respectively. Also note that the number of local parameters is as high as 45000 and yet

TABLE VIII
SUMMARY OF RESULTS FOR ISCAS85 BENCHMARK CIRCUITS- V = 0.9V, T = 25^0C AND INPUT = 0. HIGH INPUT GATES ARE BROKEN DOWN INTO SMALLER GATES

Circuit[b]	N	N_L	$\mu_{SP}(\mu A)$	$\mu_{ANN}(\mu A)$	$\Delta\mu(\%)$	$\sigma_{SP}(\mu A)$	$\sigma_{ANN}(\mu A)$	$\Delta\sigma(\%)$	**MC time(s)**
c499	1780	5340	0.295	0.302	2.126	0.138	0.122	11.340	145
c880a	1802	5406	0.191	0.192	0.676	0.079	0.076	3.766	146
c1355	2260	6780	0.239	0.240	0.496	0.096	0.094	2.101	161
c1908	3946	11838	0.457	0.459	0.511	0.190	0.183	3.745	283
c2670	4990	14970	0.640	0.656	2.463	0.258	0.254	1.313	384
c3540	07604	22812	0.946	0.955	1.006	0.384	0.372	2.964	596
c7552	15512	46536	2.060	2.068	0.403	0.886	0.801	9.522	1330
		Average mean error = 1.24%				**Average std dev error = 6.67%**			

[b] N = Number of gates. N_L - Number of Local parameters. μ_{SP}, σ_{SP} - SPICE mean and std dev. μ_{ANN}, σ_{ANN} - Mean and std dev predicted by our ANN model. MC Time - Time taken to do MC on our model

our model is very accurate. The maximum time for the MC simulation for the c7552 circuit with our model is 1330 seconds on MATLAB while SPICE took close to 60 hours to do MC simulations for the same number of samples. The average mean and standard deviation error across all the ISCAS circuits considered are close to 1.3% and 6.67%.

We also used our model on the c432 circuit to compare the mean and standard deviation of the leakage across different voltage and temperatures. Shown in Fig 11 are the mean and standard deviation values of the leakage current obtained from SPICE and from our ANN based stack model across three temperature points(30^0C, 60^0C, 90^0C). The graph clearly shows that the maximum error is at a temperature of 90^0C where the mean error is less than 5% and standard deviations error is even lesser. Note that the results in Table VIII are reported at a supply voltage of 0.9V while the graph in Fig 11 was generated at 1.2V. These two results together show how our model is both voltage and temperature scalable.

B. Impact Of Local Variations

In the previous section we validated our model thoroughly by comparing each result with SPICE. In this section we will use the model directly to investigate the impact of local variations. Modeling local variations introduces the problem of considering the effect of many independent random variables. Doing a worst case corner analysis based on using

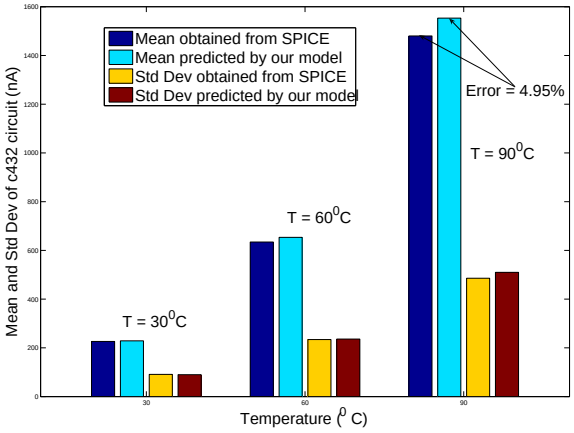

Fig. 11. Mean and Std Dev plot at a supply voltage of 1.2V and three different temperatures

TABLE IX
IMPACT OF LOCAL VARIATIONS ON THE C1355 CIRCUIT

Global	Local	Mean(nA)
0	0	197.4
$-\sigma$	0	342.6
$-\sigma$	$N(0,\sigma)$	380.1

extreme values for these variables will yield very pessimistic leakage current numbers which will be statistically unlikely as these are independent variables. Table IX lists three values for the mean of the c1355 circuit. The first one is the nominal leakage of the circuit where no process variations are considered which happens to be 197.4 nA. the second one is where the global process parameters are set to $-\sigma$ where σ is the std dev of global/local parameters. We chose 3σ to be equal to 10% of the mean. The mean in this case goes up drastically to 342.6 nA. The third case where global parameters are set to $-\sigma$ and local variations are analyzed in a statistical fashion by drawing the local process parameters from independent Gaussian distributions with standard deviation σ. The mean of 10000 such random Monte Carlo simulations goes up by about 11% to 380.1 nA. This gives us a clear indication that local variations introduce an error in leakage prediction if not considered statistically which drives the need for more sophisticated statistical leakage models.

C. Neural Network Complexity

There are two kinds of complexity associated with an ANN. One is training complexity and the other is run time complexity. Training is a one time issue and hence has less impact. We found that amongst all the models presented in this work, the maximum training time was less than 70 seconds. Run time complexity details can be inferred from Table II. Table II shows the details of the trained neural network for the common stacks. The table lists the number of inputs(N_I) to the ANN, the number of hidden units(N_H), the number of local(N_L) and global(N_G) parameter inputs. As far as storage is concerned, the ANN is characterized by the coefficients or the weights between the input and hidden layer and the hidden and output layer. The last two columns give the number of coefficients that characterize the ANN. N_{Coeff}^{Input} is the number of weight values that connect the input layer to the hidden layer. Similarly N_{Coeff}^{Output} is the number of coefficients that connect the hidden layer to the output layer. Thus the sum of these two numbers, $N_{Coeff}^{Input} + N_{Coeff}^{Output}$, is the total number of coefficients that need to be stored on disk. Apart from this the normalization coefficients also have to be stored. There are N_I input normalization coefficients and one output normalization coefficient. When a set of input parameters(N_I component vector) is presented to the ANN, the evaluation of the leakage current through that stack happens as follows.

- The N_I component vector is normalized by dividing them with the input normalization coefficients
- The input to each hidden unit is evaluated using simple matrix multiplication
- The output of each hidden unit is evaluated as $tanh()$ of its input as per Equation(7)
- The output of the ANN is evaluated as a weighted sum of the output of each hidden unit as per Equation(8)
- The output obtained in the previous step is multiplied by the output normalization coeffi-

cient

- Since we are modeling log of the leakage we have to take an exponential of the value obtained in the previous step to get the actual leakage current

From Table II we see that the four transistor stack models have the largest number of coefficients. It has 378 coefficients at the input layer, 22 coefficients at the output layer. It also has 18 normalization coefficients. Thus for the four transistor stack models we need to store a total of 418 floating point coefficients. The best estimate of the time complexity of the ANN based leakage model is obtained from last column of Table VIII. It gives the time taken to evaluate the leakage of a circuit for 15000 random input patterns to each stack model in the circuit. We observe that the time taken to evaluate the leakage for the largest circuit, c7552, is 1330 seconds. The c7552 circuit has 15512 gates and thus the average time to evaluate the leakage of a gate with our model, for a given set of input parameters, is $5.73\mu s$. All evaluations were done on an Intel P4 dual proc machine using MATLAB.

VII. Conclusion

We presented a complete PVT model for sub-threshold and gate leakage based on stacks. We showed that 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 18 stack models we were able to predict the leakage of gates in a commercial standard cell library. 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 global and local), voltage (0.6-1.2V) and

temperature($0 - 100^0C$) making it a suitable voltage and temperature scalable model. The stack models identified work well for gate leakage too. We also showed that the same stack approximation is applicable to pass transistor circuits as well. Multi finger inverters can also be modeled with the same stacks while multi finger NAND-NOR gates needed some suitable approximations to use the stack model. When tested on the ISCAS'85 benchmark circuits, the average mean error was less than 2% and the average standard deviation error was less than 7%, when compared to SPICE. Thus these gate level models incorporating local process variations will enable accurate statistical characterization of leakage currents in large CMOS designs with applications to power grid design and chip power estimation.

REFERENCES

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

[2] M. Horowitz, E. Alon, D. Patil, S. Naffziger, R. Kumar, and K. Bernstein *Scaling, Power, and the Future of CMOS IEEE International Electron Devices Meeting, December 2005*

[3] E. Chiprout *Fast Flip Chip Power Grid Analysis Via Locality And Grid Shells ICCAD, pp. 485-488, 2004*

[4] Shekhar Borkar, Tanay Karnik, Siva Narendra, Jim Tschanz, Ali Keshavarzi and Vivek De. *Parametric variations and impact on circuits and microacrchitecture. In Proc. DAC 2003.*

[5] H. Chang and S.S.Sapatnekar *Prediction of Leakage Power under Process Uncertainties ACM Transactions on Design Automation of Electronic Systems, 12(2), 2007*

[6] Shiying Xiong, Jeffrey Bokor, Qi Xiang, Philip Fisher, Ian Dudley, Paula Rao, Haihong Wang, Bill En *Is Gate Line Edge Roughness a First-Order Issue in Affecting the Performance of Deep Sub-Micro Bulk MOSFET Devices? IEEE TRANSACTIONS ON SEMICONDUCTOR MANUFACTURING, VOL. 17, NO. 3, AUGUST 2004*

[7] Asen Asenov, Savas Kaya, and Andrew R. Brown *Intrinsic Parameter Fluctuations in Decananometer MOSFETs Introduced by Gate Line Edge Roughness IEEE TRANSACTIONS ON ELECTRON DEVICES, VOL. 50, NO. 5, MAY 2003*

[8] Yun Ye, Frank Liu, Sani Nassif, Yu Cao *Statistical Modeling and Simulation of Threshold Variation under Dopant Fluctuations and Line-Edge Roughness DAC, 2008*

[9] Tao Li, Wenjun Zhang and Zhiping Yu *Full-Chip Leakage Analysis in Nano-scale Technologies: Mechanisms, Variation Sources, and Verification DAC, 2008*

[10] 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.*

[11] S. Narendra, V. De, S. Borkar, D. Antoniadis, and A. Chandrakasan, *Full-chip subthreshold leakage power prediction model for sub-0.1 m CMOS ISLPED, Monterey, CA, Aug. 2002, pp. 19-23*

[12] 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*

[13] 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.*

[14] K Roy, S Mukhopadhyay and Hamid Mahmood *Leakage current mechanisms and leakage reduction techniques in deep sub-micrometer CMOS circuits* Proceedings of the IEEE, Vol 91, No.2, February 2003

[15] Rao R.R, Devgan A, Blaauw D and Sylvester D. *Analytical Yield Prediction Considering Leakage/Performance Correlation IEEE Transactions on computer-aided design of integrated circuits and systems, VOL. 25, NO. 9, September 2006*

[16] T. Li and Z. Yu *Statistical Analysis of Full-Chip Leakage Power Considering Junction Tunneling Leakage DAC, pp. 99-102, 2007*

[17] J.Gu S.S.Sapatnekar C. Kim *Width-dependent Statistical Leakage Modeling for Random Dopant Induced Threshold Voltage Shift DAC, 2007*

[18] Shengqi Yang, Wayne Wolf, N. Vijayakrishnan, Yuan Xie, Wenping Wang. *Accurate Stacking Effect Macro-modeling of Leakage Power in Sub-100nm Circuits Conference Proceedings. 18th VLSI Design,2005.*

[19] Janakiraman. V Bishnu Prasad Das V Visvanathan Bharadwaj Amrutur. *Leakage modelling of logic gates considering the effect of input vectors. VDAT, 2007.*

[20] Janakiraman. V Bishnu Prasad Das Bharadwaj Amrutur. *Voltage and Temperature Scalable Standard Cell Leakage Models Based On Stacks For Statistical Leakage Characterization. 21th VLSI Design Conference, January 2008.*

[21] Acar Emrah, Devgan Anirudh, Nassif and Sani R. *Leakage and Leakage Sensitivity Computation for Combinational Circuits In ISLPED '03: pages 96-99. New York, NY, USA, 2003. ACM Press.*

[22] Sarvesh Bharadwaj and Sarma Vrudhula. *A Fast and accurate approach for full chip leakage analysis of nano-scale circuits considering Intra-die correlations Conference Proceedings. 20th VLSI Design,January 2007.*

[23] Christopher M. Bishop *Neural networks For Pattern Recognition Oxford university press, NY, 2007*

[24] Beyene, W. T. *Application of Artificial Neural Networks to Statistical Analysis and Nonlinear Modeling of High-Speed Interconnect Systems IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, VOL. 26, NO. 1, JANUARY 2007*

[25] Wei Zhao and Yu Cao, *Predictive technology model for nano-CMOS design exploration, ACM Journal on Emerging Technologies in Computing Systems (JETC), Volume 3 , Issue 1, April 2007 http://www.eas.asu.edu/ ptm/*

APPENDIX-A

. MULTI FINGER TRANSISTORS

A gate with higher drive strength can be realized with a transistor of larger width. However, to reduce the diffusion capacitance and gate resistance, it is generally implemented as multiple parallel unit fingers. At a schematic level they can be thought of as multiple transistors in parallel. The most commonly used multi finger gate is the inverter. Inverters need to be used as buffers to improve drive strength. We look at this case specially before going on to general multi finger gates. Shown in Fig A-1 is an inverter with three fingers. When the input is set to 0, the PMOS transistors are turned ON and the drains of the NMOS are at V_{DD}. The leakage for this state is the sum of leakage currents of each of the NMOS transistors N1-N3. This can be predicted with the *n1/0* model. Similarly for the case when the input is set to 1, the leakage can be predicted with the *p1/1* models. Note that we haven't modeled a new stack for multi finger inverters and this technique is valid for any number of fingers. Table A-1 shows the result for 2-5 finger inverters. As we can see clearly, the error in mean and standard deviation are less than 0.4% and 0.9% showing that our theoretical explanation is valid.

We will next consider stack approximations for other multi finger gates. Consider the two transistor NMOS stack with four fingers shown in Fig A-2. We have three ways of approximating the leakage of this four finger stack.

- Neglect the fact that the drains of N1, N3, N5 and N7 are shorted and treat the four stacks independently and use the n2/0 model for each of the four stacks
- Model the entire four finger stack with a neural network
- Neglect the fact that the drains of N3 and N5 are shorted and use the two transistor NMOS stack with two fingers (n2f2/0)[2] to predict the leakage.

The first technique is too much of an approximation and introduces too much error. The second technique of modeling the entire four finger stack with a neural network will give

[2]We will suffix **f2** to each model name in Table II to refer to the two finger version of each stack. As an example, the two transistor NMOS stack with two fingers and its input set to zero will be called **n2f2/0**. Similarly for three finger stacks and so on.

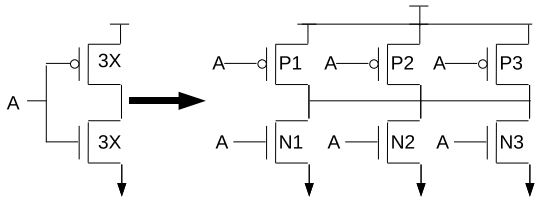

Fig. A-1. An inverter with three fingers

TABLE A-1
SUMMARY OF RESULTS FOR A MULTI FINGER INVERTERS.
$V_{DD} = 0.9V$, T = 25^0C
MEAN(μ) MEAN ERROR ($\Delta\mu$)AND STD DEV(σ) STD DEV ERROR ($\Delta\sigma$) BETWEEN SPICE AND OUR ANN MODEL.

# Fingers	μ (pA)	$\Delta\mu$ (%)	σ (pA)	$\Delta\sigma$ (%)
2	638.19	0.33	354.45	0.36
3	961.48	0.35	497.23	0.56
4	1278.30	0.37	657.50	0.81
5	1599.83	0.34	789.47	0.56

very accurate results but isn't scalable with higher finger stacks. Let us take a closer look at the third option.

While using neural networks to model stacks we said the model has been successfully trained if the maximum sample error is less than some threshold(15% in our case). This means that across **all** testing samples used, the maximum error was less than 15%. By approximating the 4 fingers into two independent 2 finger stacks we obviously cannot approximate the

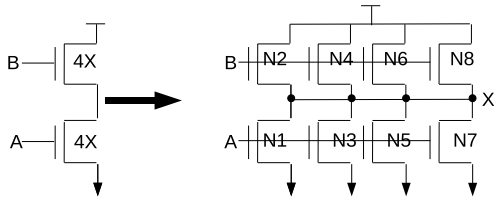

Fig. A-2. A two transistor NMOS stack with four fingers

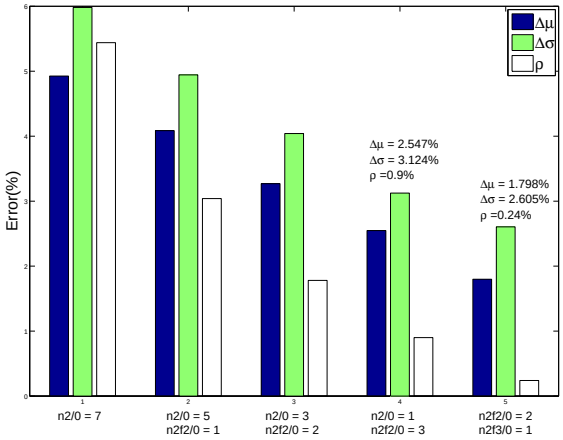

Fig. A-3. Comparison of different partitions for a seven finger stack. Shows mean error($\Delta\mu$), standard deviation error($\Delta\sigma$) and probability of error(ρ)

leakage of the four finger stack accurately for all possible combinations of process parameters. The reason for the error is that the intermediate node X will get affected by the process settings of any of the eight transistors but our approximation does not account for this. However, we justify the approximation as the probability of the answer being erroneous by more than the specified threshold is very low.

However, evaluating this probability for a given threshold is not an easy task since the functional form of the actual leakage is extremely complicated. Instead we look at the trend of the probability of error, which we denote as ρ, for different stack approximations through simulations to pick the right stack

combination. We trained the neural network to model 2 transistor NMOS stack with two fingers and three fingers respectively. We then considered a 7 finger NMOS stack with two transistors and measured the leakage for 5000 SPICE simulations to obtain the actual values. We then looked at different partitions of stack with 1-3 fingers to predict its leakage. Fig A-3 shows the mean error ($\Delta\mu$), standard deviation error ($\Delta\sigma$) and probability of error for the seven finger NMOS stack with two transistors. Below each set of bars, we have indicated the kind of partition i.e. the number of n2/0, n2f2/0 and n2f3/0 models.

The first set of bars show the errors for the partition of seven single finger stacks (n2/0). As we move to the right we keep decreasing the number of n2/0 models and use more of the n2f2/0 models. As we can see clearly, all three errors($\Delta\mu$, $\Delta\sigma$, ρ) decrease as the number of n2/0 models are reduced. This behavior is expected. As mentioned before, by neglecting the fact that the drains of the lower transistors are shorted we are treating the currents through each stack as independent when they are actually not. By considering more n2/0 models we are treating more currents as independent which introduces more error as shown in the graph. Thus it is clear that we need appropriate two finger stack models. Now the question is whether we need a three finger model as well.

From the graph we see that the last set of bars has least error. This is the case when the seven finger stack is partioned as two two finger stacks(n2f2/0) and one three finger stack (n2f3/0). We must note that the saving in error compared to the previous partition (3 two finger stacks(n2f2/0) and one single finger stack(n2/0)) is not too much. The mean error drops from 2.55% to 1.80%. Standard dev error drops from 3.12% to 2.61% and the probability of an incorrect prediction drops from 0.9% to 0.24%. Thus the choice we need to make is whether we want to model the three finger versions of the stacks in Table II as well or do we accept a slightly higher error and stop with two finger models. As mentioned earlier, multi finger inverters are most commonly used and the usage of multi finger NAND / NOR gates are

not as common. Since we have modeled the multi finger inverter very accurately we can tolerate the excess error in higher order multi finger gates. *Thus it is sufficient to model the single finger stacks and the two finger stacks alone*.