On Efficient and Accurate Surrogate Models of Leakage in CMOS Gated Circuits

Ph.D. Thesis

by

Lokesh Garg

ID - 2010REC101

Under the Supervision

of

Dr. Vineet Sahula

Professor

Department of Electronics and Communication Engineering

Malaviya National Institute of Technology, Jaipur, INDIA

May 2016

Certificate

This is to certify that the thesis entitled "***On Efficient And Accurate Surrogate Models of Leakage in CMOS Gated Circuits***", being submitted by ***Lokesh Garg*** to the Department of Electronics and Communication Engineering, Malaviya National Institute of Technology, Jaipur, for the award of the degree of Doctor of Philosophy, is a bonafide research work carried out by him under my supervision and guidance. The results obtained in this thesis have not been submitted to any other university or institute for the award of any other Degree.

Dr. Vineet Sahula

Professor

Department of Electronics and Communication Engineering

Malaviya National Institute of Technology,

Jaipur-302017, India.

Abstract

In nanoscale technologies, leakage consumption is a major constraint for large systems while continuous performance increment and area reduction of digital CMOS integrated circuits. Leakage in digital circuits depends on the different parameters such as process variations - length, threshold voltage, oxide thickness in a transistor, input vector applied to the logic gate, supply voltage and temperature variations, and also width of transistors in a logic gate. Digital circuits are synthesized using standard cell library. Standard cells are generally pre-characterized in terms of power, delay, area. This necessitates accurate and efficient models for characterization. Since process variations, supply and temperature variations were not significant in earlier technologies. Thus, previous models have used BSIM device equations to only characterize leakage of single NMOS/PMOS transistors. Leakage of a gate is calculated by scaling the leakage of characterized single NMOS/PMOS transistor on the basis of either 'OFF' transistors on a transistor stack or finding out the node voltage at internal nodes of a gate. Characterization of only two models consumes very less time but models are neither accurate nor scalable to be used for technologies considering variations. BSIM device equations are also not sufficient to estimate mean and standard deviation of leakage. Empirical models such as Exponential Quadratic (EQ), Polynomial Equation (Poly) have been proposed earlier to estimate mean and standard deviation of leakage for a CMOS gate. Mean and standard deviation from different CMOS gates can then be statistically added to find out the full chip leakage. These available methods in literature are fast but result in large error because EQ and Poly models are second order Taylor expansion of the exact equation that can be fitted with the SPICE simulated data. These models also results in large characterization time due to use of separate model for each input vector per gate. Artificial Neural Network (ANN) based stack models have been proposed to capture non-linearity due to variations with reduced number of the models. However, the effects of 'ON' transistors in ‘OFF’ networks have not been considered accurately, resulting in large error in leakage estimation. We have proposed Kernel based Support Vector Machine (SVM) surrogate model to provide estimate of leakage based on transistor stacks. Additionally, improved effective width estimation method is proposed to estimate leakage of complex gates consisting of parallel transistors or stack of parallel transistors. We only required 56 models to capture subthreshold and gate tunneling leakage across 20 CMOS gates with 176 input vector combination, achieving error within 1%. Best kernel with optimum tuning parameters is employed after deliberate exploration to model each transistor stack. Error driven active learning methodology is proposed to adaptively select samples only in the large error areas. Our proposed methodology results in SVM models build up by using quite small number of training samples, thus reducing model simulation time without suffering the accuracy.

Power gating is one of the leakage optimization techniques, in which extra transistors are added between supply rail and actual circuit. When circuit is in sleep mode extra transistors go to cut-off mode, thus reducing total voltage across circuit and consequently the leakage current. In ground gating case, virtual ground voltage (V_{gnd}) has been used as a key parameter to exploit trade-off among leakage power saving, ground bounce noise, delay degradation, etc. Thus, accurate and efficient estimation of V_{gnd} is of prime importance in power gating circuits. Previous work results in large error due to: 1) conservative exponential linear (EL) and 3^{rd} order polynomial (Poly3) leakage model 2) inaccurate assumption for applied voltages at the input of CMOS gates. We have proposed Kernel based SVM models with accurate consideration of input voltage conditions to overcome these limitations. Our proposed method results in less than 1% error in V_{gnd} as compared to 10% error as reported in literature.

Acknowledgments

The completion of my dissertation and subsequent Ph.D. has been a long journey. Completion of this doctoral dissertation was possible with the support of several people. First and foremost I want to thank my research guide, Dr. Vineet Sahula, Professor, ECE Department MNIT Jaipur, for supporting me during these past five years through his guidance and consistent encouragement. I consider it as a great opportunity to do my doctoral programme under his guidance and to learn from his research expertise. He is a person with an amicable and positive disposition. I would like to thank you for all your contributions of time, insightful discussion about the research, giving me a freedom to pursue my work and allowing me to grow as a research person. Thank you Sir!

I would also like to give a heartfelt, special thanks to Dr. Dharmendar Boolchandani, Professor, MNIT Jaipur, for extending his tremendous support and encouragement. His patience, flexibility, genuine caring and concern, and faith in me during the dissertation process, hold me boosted.Thanks for all motivational talks, guidance and listening to me all the times. Thank you Sir!

I would like to thanks members of my Departmental Research Committee (DREC), Dr. Dharmendar Boolchandani (Professor, MNIT Jaipur), Dr. Lava Bhargava (Associate Professor, MNIT Jaipur), Dr. C. Periasamy (Assistant Professor, MNIT Jaipur) and Dr. Satyasai Nanda (Assistant Professor, MNIT Jaipur) for providing me valuable comments and suggestions during presentation. Thank you!

The thesis would not have come to a successful completion, without the help I received from the faculty and staff of the ECE Department of Institute, especially Head of the Department Professor K. K. Sharma. I like to thank them for all non-technical and technical support to carry out my research work. Thank you!

Another stanch supporter and is a special friend of seven years, Sapna Khandelwal.

Her support and belief in me were a treasure. She's personally endured my long hours on the computer and was patient with my proverbial absence most of the time. I would like to thank her for all discussions, encouragement, editing my papers and listening to me patiently every time. Thank you!

My colleagues, Renu Kumawat, R. A. Patil, Ramakrishna, Niketa, Ashish, Shashikant, Yogendra, have all extended their support in a very special way, and I gained a lot from them, through their personal and scholarly interactions, their suggestions at various points of my research programme. I would like to thank them for all tea time discussions, all pizza parties and all editing work. Thank you!

My time at MNIT was made enjoyable and enriched by many of the M.Tech students, who come across these last five years. I take this time to thank Pramod, Gauri, Nupur, Swaraj, Kamini, Somendra, Prakhar, Arjun, Vikas, Gaurav and Arun for all those precious memories, discussions at canteen, lunches & dinners and memorable trips. I am also grateful for the time spent with my hostel mates Jitendra, Vinay, Vikas, Vipin, Leelesh, Rakesh and many more. Without all of you my PhD would not have be so much fun. Thank you!

Lastly, I would like to thank my family, my parents, brother, sister, sister-in-law and brother-in-law, who supported me in every possible way to see the completion of this work. I am very much indebted for all their love and encouragement. My niece Khushi and nephew Vihaan are twinkle of my eyes. Though I didn't find too much time to spent with them, but listening about their kiddish acts and seeing their pics, re-energized and given me a much-needed reprieve from work.Thank you!

Above all, I owe it all to Almighty God for granting me the wisdom, health and strength to undertake this research task and enabling me to its completion. Thank you God!

(Lokesh Garg)

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

CMOS	Complementary Metal-Oxide-Semiconductor
NMOS	N-Channel Metal-Oxide-Semiconductor
PMOS	P-Channel Metal-Oxide-Semiconductor
Stack	A set of series transistors
BSIM	Berkeley Short-channel IGFET Model
EL	Exponential Linear
EQ	Exponential Quadratic
Poly	Polynomial
ANN	Artificial Neural Network
SVM	Support Vector Machine
SVC	Support Vector Classification
SVR	Support Vector Regression
RBF	Radial Basis Function
MLP	Multi-Layer Perceptron
V_{GS}	Gate to Source Voltage of transistor
V_{BS}	Bulk to Source Voltage of transistor
V_{DS}	Drain to Source Voltage of transistor
V_T	Thermal Voltage
V_{gnd}	Virtual Ground Voltage
V_{vdd}	Virtual Supply Voltage
V_{dd}	Supply Voltage
V_{th}	Threshold Voltage of a transistor
L	Length of a transistor

T_{ox}	Oxide-thickness of a transistor
T	Temperature
I_{sub}	Subthreshold leakage
I_{gate}	Gate Tunneling leakage
I_{BTBT}	Band to Band Tunneling leakage
DIBL	Drain Induced Barrier Lowering
D2D	Die-to-Die
W2W	Wafer-to-Wafer
L2L	Lot-to-Lot
WID	With-In-Die
RV	Random Variable
PVTW	Process-Voltage-Temperature-Width
UDSM	Ultra-Deep Sub-micron
T_{model}	Model Characterization Time
μ	Mean
σ	Standard Deviation
SRM	Structural Risk Minimization
ERM	Empirical Risk Minimization
GA	Genetic Algorithm
DE	Differential Evolution
LHS	Latin Hypercube Sampling
I_{LC}	Leakage current through logic cluster
$V_{steady,state}(sleep)$	Steady state leakage during sleep mode
GP/LP	Global/Local Process Parameters
PDF	Probability Density Function
MC	Monte Carlo

PDN	Pull-Down Network
PUN	Pull-Up Network
R_{ON}	‘ON’ resistance of a transistor
MSE	Mean Square Error
T_{train}	Model training time
T_{sim}	Model simulation time
DOE	Design of Experiments
CPL	Complementary Pass Transistor Logic
TG	Transmission Gate
PTL	Pass Transistor Logic
I_{footer}, $V_{th,footer}$, W_{footer}	Leakage current, threshold voltage and width of footer transistor
V_p	Maximum value of V_{gnd} that can be predicted by analytical models
DSTN	Distributed Footer Transistor Network
$V_{steady,state}$	Steady state virtual ground voltage
ρ	Correlation coefficient

Chapter 1

Introduction

The world has witnessed phenomenal growth in semiconductor industry in past few decades and the increased role of CMOS in digital integrated circuits. In pursuit to follow Moore's law [8] scaling has been done by a factor of 0.7 at each technology node to satisfy the triad of power, performance and area. However by relentless scaling of transistor dimensions, two major concerns have started to invade the VLSI industry. First is the static power dissipation due to increased leakage current which is produced when transistor is 'OFF'. Earlier only dynamic and short circuit power were accounted in power dissipation of digital circuits. But now by scaling transistor dimension, supply voltage (V_{dd}) need to be scaled to satisfy dynamic and short circuit power demands. Further threshold voltage (V_{th}) has to be reduced to maintain sufficient gate drive for performance. This V_{th} reduction primarily attributes 7.5× total leakage increment per generation [9], which results in degraded static power.

Second is the process variation, the deviation from nominal value of a transistor or circuit parameter. After fabrication, transistor parameters like threshold voltage (V_{th}), channel length (L), oxide thickness (T_{ox}) vary across the chips and within the chip. This in turn varies the 'ON' and 'OFF' currents of transistor which finally affects the power and performance. There are several process and environmental factors on which leakage depends exponentially. This includes variation in L, V_{th}, T_{ox}, V_{dd} and temperature (T). By including process variation, leakage current can vary about 6.5× from its nominal value [10] . Hence, even small variations in these parameters leads to large change in leakage current.

1.1 Sources of Leakage Power

Leakage current refers to the current flow through transistor in steady state condition of a circuit. In bulk-CMOS technologies, several leakage sources those exist in a transistor as shown in Figure 1.1 but dominant leakage mechanism are:

i). Subthreshold leakage (I_{sub}) ii). Gate Tunneling leakage (I_{gate}) iii). Band to Band Tunneling leakage (I_{BTBT})

1.1.1 Subthreshold Leakage

In CMOS logic any of the NMOS or PMOS network will remain 'OFF', thus low resistance path from V_{dd} to ground never exists. To reduce dynamic power dissipation supply voltage is reduced. Along with it, threshold voltage is also scaled to maintain

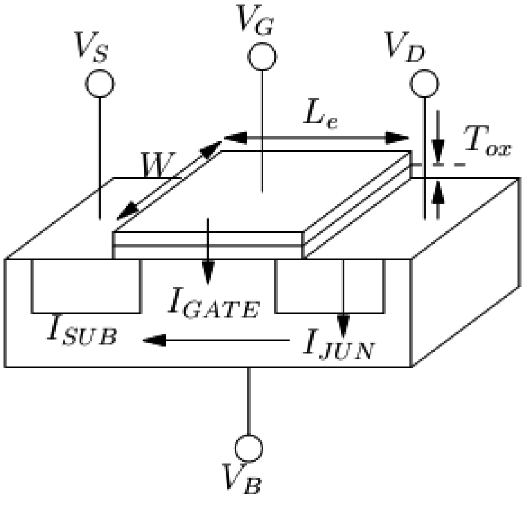


Figure 1.1: Different Leakage Component in NMOS Transistor

sufficient drive strength [11]. This scaling results in non zero current which flows between drain and source, even when transistor is off. When transistor operates in weak inversion region (when gate voltage is below V_{th}) then charge carriers move by diffusion process due to longitudinal electric field between drain and source. This diffusion current is named as sub-threshold leakage (I_{sub}). This I_{sub} [12] can be modeled as:

$$I_{sub} = Ae^{\frac{1}{mv_T}(V_{GS}-V_t)}(1-e^{\frac{-V_{DS}}{v_T}}) \tag{1.1}$$

Here

$$A = \mu_0 C_{ox}\frac{W}{L}(v_T)^2 e^{1.8} e^{\frac{-\triangle V_{th}}{\eta v_T}} \tag{1.2}$$

and

$$V_t = V_{th0} - \gamma' V_{sb} + \eta V_{ds} \tag{1.3}$$

Here $v_T = \frac{kT}{q}$ is the thermal voltage, m is the sub-threshold swing coefficient, μ_0 is the zero bias mobility, W and L are the width and effective length of the transistor C_{ox} is the gate oxide capacitance, V_{th0} is the zero bias threshold voltage, γ' is the linearized body coefficient, η represent the effect of V_{ds} on V_{th}.

There are many parameters those affect sub-threshold leakage:

1. Drain voltage modulates the width of depletion region that results in low energy barrier and increased movement of charge carriers, hence I_{sub} increases. This is called Drain Induced Barrier Lowering (DIBL) effect which also degrades V_{th} and attributes to high leakage.

2. I_{sub} is also affected by applied input vector on logic gate, which is called as stacking effect. This effect modify the I_{sub} by modulating V_{th} through changing body bias voltage of the transistor and modifying the effective drive strength at gate terminal.

3. Degradation in V_{th} due to channel length reduction known as V_{th} roll off [12], which increases I_{sub}.

4. I_{sub} is proportional to temperature and V_{th} is inversely proportional to temperature. So combined effect of these two phenomenon results in high I_{sub}.

5. Transistor size (Width) also affects leakage (increase/decrease). Narrow width transistor modulates V_{th} which in turn changes I_{sub} [12].

6. Reduction in oxide thickness attributes to enhanced I_{sub}.

1.1.2 Gate Tunneling Leakage

In nanometer regime, depth of the transistor needs to be scaled in order to maintain the gate control over drain current to improve the performance of transistor gate oxide and junction. This scaling raises the electric field which results in direct tunneling of charge carriers in drain/source overlap region and substrate to the gate through gate oxide. This is called gate tunneling leakage (I_{gate}) [13]which can be expressed as follows.

$$I_{gate} = A \times E_{ox}^2 \times e^{\frac{B}{E_{ox}}} \tag{1.4}$$

Here, A and B are the parameters related to mass of electrons and barrier height of conduction band, E_{ox} is the electric field across the gate oxide which depends on oxide material and gate oxide thickness.

The other phenomenon which contributes to I_{gate} are:

1. Fowler - Nordheim (FM) tunneling which occurs when voltage applied across the oxide is more than ϕ_{ox}.

2. Hot carrier effect in which charge carriers gain sufficient kinetic energy to overcome the oxide band gap.

UnlikeI_{sub}, I_{gate} occurs in both conditions, either transistor is ‘ON’ or ‘OFF’. In ‘ON’ state direct tunneling occurs. In ‘OFF’ state, drain potential causes high electric field across the gate, resulting in tunneling which remains localized at gate-drain overlap. This is called Edge Directed Tunneling (EDT) gate leakage [14] and has less magnitude.

1.1.3 Band to Band Tunneling Leakage

In short channel devices, depletion width majorly contributes to decrease in channel length. As drain potential rises, depletion width increases which reduces part of the channel length to be inverted, resulting in lower V_{th}. As V_{th} decreases, I_{sub} increases. Drain/Source regions are heavily doped to make smaller depletion region, in order to reduce DIBL effect. These heavily doped regions create reverse biased PN junction with the body. A large electric field is produced between the body and drain/source regions because of different doping concentrations, resulting in a current flow due to tunneling of charge carriers. This tunneling occurs between the conduction band of 'n' region and valence band of 'p' region, referred as band to band tunneling (I_{BTBT}) leakage. It's magnitude is very less compared to I_{sub} and I_{gate}. The current density of I_{BTBT} [12] can be given by:

$$J_{b-b} = A\frac{EV_{app}}{\sqrt{E_g}}e^{-B\frac{E_g^{\frac{3}{2}}}{E}} \tag{1.5}$$

Here $A = \frac{\sqrt{2m^*}q^3}{4\pi^3 h^2}$ and $B = \frac{4\sqrt{2m^*}}{3qh}$, m^* is the effective mass of the electron, $E = \sqrt{\frac{2qN_aN_d(V_{app}+V_{bi})}{\varepsilon_{si}(N_a+N_d)}}$ [15] is the electric field across the junction, where N_a and N_d are the doping levels of the p and n side, V_{bi} is the built in potential, ε_{si} is the permittivity of silicon,V_{app} is the applied reverse bias, E_g is the energy band gap.

1.2 Process Variation

Aggressive scaling leads to variations in key MOSFET parameters, and are increasing at an unacceptable level. There are two major sources of variations [16]. First is environmental which includes temperature and supply voltage variations, during circuit functioning. Second is the physical which includes lack of process control during fabrication and results in permanent variations. Process variations occur due to variability in process parameters such as dopant concentration, L, W, T_{ox} and dielectric thickness at the time of manufacturing. Process variation can be broadly divided into two categories [17, 18]:

i) Inter Die Variation ii) Intra Die Variation

Figure 1.2 shows both type of variations.

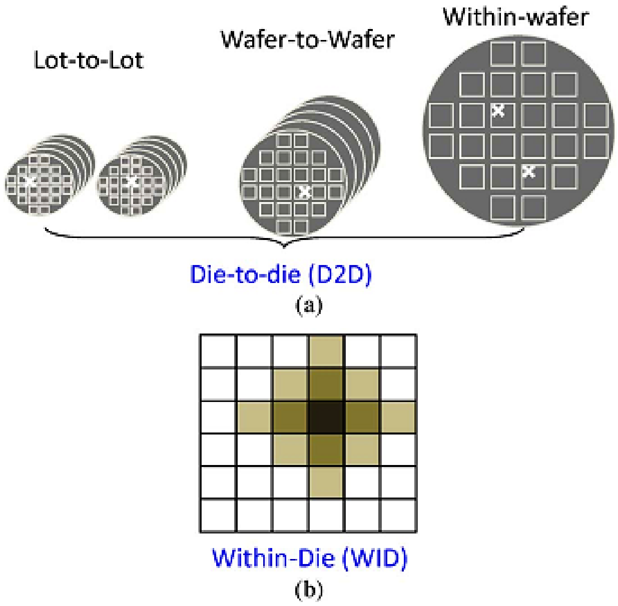


Figure 1.2: Types of Variations (a) Inter Die Variation (b) Intra Die Variation [1]

1.2.1 Inter Die Variation

It includes variation from Die-to-Die (D2D), Wafer-to-Wafer (W2W) and Lot-to-Lot (L2L). For older technology nodes, this was the major source of variation. Effects of these variations are same for all devices inside the die, but varies from one die to another [19]. Due to the impact of these variation, all devices inside a single die get affected in a similar manner. So their effect can be added by defining a single parameter to measure the effect of these variations. This variation shifts the mean value of the parameter for e.g. causing the length of all transistors in a single die to be smaller or larger than the nominal value [1]. This is called as globally correlated variation. Let the nominal value of any parameter P is P_{NOM} and inter die variation can be modeled as a single parameter, $\triangle P_{Global}$. The value of the resulting parameter after adding inter die variation can be denoted as:

$$P = P_{NOM} + \triangle P_{Global} \tag{1.6}$$

1.2.2 Intra Die Variation

It includes With-In-Die (WID) variations and arises due to the offset created in fabrication process. In the present and future technology nodes, this is the major source of variation. Two identically designed devices at two different location in a single chip (die) can have different parametric variations. Thus, these variations affect the matched properties of transistors within a chip. These variations are called local since each transistor gets affected differently, causing some transistors to have larger length while others to have smaller than the nominal value. They affect the mean of variation distribution. These variation can be classified into two other types of variations [20]:

i) Systematic Variation ii) Random Variation

1.2.2.1 Systematic Variation

These variations are layout dependent, therefore deterministic in nature. For example, variations in transistor length due to lithographic limitation during manufacturing. This component is defined using distance between the devices in order to find out correlation between them. This spatial correlation is locally circuit and layout dependent and globally location dependent. Thus spatially close transistors have similar offset effects. These variations can be modeled by dividing the chip into grids [21]. Transistors within the grid are fully correlated, but as distance between grid increases correlation decreases. Here, effect of variation is modeled by one parameter per grid. Thus in the i^{th} grid, process parameter P is given as:

$$P_i = P_{NOM} + \triangle P_{Global} + \triangle P^i_{Spatial} \tag{1.7}$$

1.2.2.2 Random Variation

These variations are inherent statistical fluctuations in process parameters such as oxide thickness, line edge roughness and random dopant fluctuations (N_a) [22]. They introduce variation between different devices in the same chip or between different chips. Some of these fluctuations (L) are either spatially correlated while some other (T_{ox}, N_a) have zero correlation [1]. These variations are unpredictable in nature. It basically includes variations which are either truly random in nature or which can't be modeled as systematic variations. Their behavior can be represented in terms of probability distribution, which may either be implicit (Log-normal or Gaussian distribution) or explicit (using large number of samples from fabrication line measurement). Here the effect of variation is modeled by one random variable per transistor per process parameter. Thus for the k^{th} transistor in the j^{th} logic gate situated in i^{th} grid, process parameter P is given as:

$$P_i = P_{NOM} + \triangle P_{Global} + \triangle P^i_{Spatial} + \triangle P^{kj}_{Random} \tag{1.8}$$

Systematic variations can't be determined until the layout is complete, so they are treated as random variations in early stages of design. Due to the aggressive scaling, Intra Die variation is more significant than Inter Die variation.

1.3 Effect of Environmental Factors, Process Variation and Input Vectors on Leakage

At transistor level, leakage current manifests itself in various ways and impacts the entire system. Impact of leakage at every level is affected by environmental factors such as temperature (T) and supply voltage (V_{DD}), variation in process parameters and the applied input vectors [11]. In nanometer technology, leakage component exponentially depends on V_{DD}, T , L and T_{ox}. So it is necessary to model the leakage in the presence of environmental & process variations, for different input vector combinations.

1.3.1 Environmental Factors Effect

I_{sub} of a logic gate depends on environmental factors like T ,V_{DD}.

1. V_{DD} changes I_{sub} significantly due to DIBL effect. As drain potential increases, depletion width at the drain end also increases which results in low energy barrier for the charge carriers to travel. This in turn increases I_{sub}. This effect can be modeled as in [23]:

$$V_{th} = V_{th0} - \eta V_{DS} \tag{1.9}$$

Here η is the DIBL coefficient. Reduction in V_{th}, either due to channel length reduction or by increasing depletion region width, results in increased I_{sub}. There is an exponential relationship which exists between I_{sub} and V_{DD}. A 20% variation in V_{DD} changes leakage by 2× [24].

2. T is coupled with leakage in a significant manner. T increases leakage power in a non linear manner, which produces heat and further boosts the temperature. This consequently increases the leakage in a loop, until generated power balances the

removed power. Increased T also reduces V_{th} which also contributes to increased leakage. A $30C$ change in T increases I_{sub} by 30% [24].

1.3.2 Process Variations Effect

In cutting edge technology effect of process variation on process parameter is increasing. Values of these process parameters are modeled as random variables [21] which follows probability density function (PDF). These variations turn into upsetting the values of performance parameters. Leakage power dissipation of a circuit no longer remains constant because leakage is exponentially related to process parameters. Hence, leakage can be modeled as a random variable with log normal distribution, which varies significantly in the presence of process variations. Leakage is mostly sensitive to L, T_{ox} and V_{th}. By applying process variations on some benchmark circuit [25] shows that I_{sub} reflects 30% increment than the mean value considering channel length variation. In $0.18\mu m$ technology, experimental measurements from manufactured chips shows that $30mV$ threshold voltage variation results into 20× variation in leakage [26].

1.3.3 Input Vectors Effect

Biasing conditions or terminal voltages in the form of input vectors affect the leakage current. The leakage in transistor stack is a function of input pattern and number of transistors. As source potential increases, decrement in V_{GS} and V_{BS} reduces leakage. In a series transistor stack, when two or more transistors are 'OFF', source voltage of lower transistor will be less than the upper transistor. This leads to low V_{DS}, negative V_{GS} and negative V_{BS} of upper transistor which finally increases V_{th} and reduces leakage. This is termed as stack effect. In spite of single 'OFF' transistor, stacking of two 'OFF' transistors provide less leakage [11]. Stack effect in natural stack (exists in logic gates) reduces leakage in static mode by loading a suitable input vector which maximizes the number of PMOS & NMOS 'OFF' transistors [11]. For all technology nodes, input vector dependent leakage remain same, but different input combinations yield different leakage. Different input patterns turn 'OFF' and 'ON' different number of transistors in a stack, which shows variation in resistance between power lines and produces various leakage values. Any input combination that makes maximum transistors 'OFF' between power lines, produces maximum resistance and minimum leakage. Thus particular input combination can be used to reduce leakage in standby mode without any area overhead.

1.4 Motivation

Leakage current in digital circuits should be minimized to keep scaling the transistors in latest technology nodes. But, due to process and environmental variations, leakage current is not a constant quantity and thus, fabricated IC's will have different value of leakage power consumption. Some IC's may consume greater leakage than the allowed limit, reducing the yield of fabricated chips. Leakage is highly nonlinear with respect to variation in process and environmental parameters. Accurate and efficient models are required to estimate the leakage considering variations for each input vector applied to the CMOS circuit. Previously proposed models result in significant error due to the inaccurate assumption of node voltage conditions in the circuit, without/inaccurate consideration of 'ON' transistors in CMOS gates, conservative model assumption relating leakage with variational parameters. The number of models required to characterize leakage for each input considering input vector dependence can be very large. Thus, it is earnestly required to reduce the number of models. These reduced set of models should be able to estimate leakage of all CMOS gates presented in standard cell library without incurring large error. Previous work on number of models reduction results in large error due to inaccurate extracted models. Function form of leakage in terms of variations can be accurately represented by machine learning based methods such as Artificial Neural Networks (ANN), Support Vector Machine (SVM), etc. These are the black box kind of models and can be used in place of SPICE simulation in sampling based full-chip leakage estimation techniques. These black box models should be developed with minimum number of training samples and optimal set of parameters. Thus, effective strategies are required to build these models.

Power gating is employed for the leakage optimization, but results in delay and energy increase, ground bounce noise introduction etc. Application of power gating changes the internal node voltages in CMOS gates. Leakage models for more CMOS gates in power gated circuits will be different than simple CMOS circuits due to change in input voltage conditions. Actual node voltage conditions are need to be predicted at the inputs of CMOS gates for valid analysis of power gating methods. Moreover, the conservative leakage models should not be used for accurate performance estimation of parameter. Models based on machine learning approaches offer better choices. Variations in footer transistor (extra transistor added in ground gating case) parameters results in trade-off with different performance parameters. Any model for power gated circuits should be inclusive of a function of footer transistor parameters. Thus, an accurate and efficient methodology may be evolved to analyze power gated circuits.

1.5 Scope of the Work

In this thesis, we have concentrated on characterizing I_{sub} and I_{gate} for 20 CMOS gate standard cell library using small number of models yet constraining the leakage error under tolerable limits ($< 1\%$). We have used kernel based Support Vector Machine

(SVM) to characterize each leakage model. Each of the leakage model is developed considering 10% variation assuming Gaussian distribution in L, V_{th} and T_{ox}, 10% uniform variation in V_{dd} and T, whereas W is also uniformly varied between $28nm$ to $500nm$. These are the black box kind of models and do not provide the equation relating the input and output parameters. These models can be used in sampling based full chip leakage estimation methods.

Kernels with their different parameter values largely affect the accuracy and efficiency of a SVM model. Many kernels have been proposed in the literature [27]. We focus on commonly used kernels such as Radial Basis Function (RBF), Multi-Layer Perceptron (MLP), Log, Linear and Poly [27]. For SVM based leakage model, best kernel with optimal tuning parameter set must be used. We exploit the properties of both grid search based methods and global optimization methods and use both of them in a single optimization loop. Active learning method based on dominant sample selection is used to find out minimal set of samples to train SVM model. It helps in reducing the model evaluation time by cutting down the number of computations required to simulate unknown testing samples. Non-dominant samples are further removed to make model sparse, resulting in extra saving in training samples.

In ground gating case of power gated circuits, virtual ground voltage (V_{gnd}) is used as a parameter to explore trade-off among different performance parameters. We focus on using SVM based black box models considering the effect of input vectors, accurate voltage consideration at input of CMOS gates and accurate leakage models of power gated circuits. These V_{gnd} models are generated with respect to the footer transistor and depend on footer parameters of transistor such as threshold voltage, sleep signal voltage, width of transistor etc. Separate models for static and dynamic V_{gnd} are developed, which can further be used for evaluating the merits of power gating methods.

1.6 Our Contribution

- Improved stack based leakage estimation with effective width based methodology is developed, resulting in more accurate calculation with fewer models.
- Training samples are actively selected to prepare SVM models with minimum number of samples, resulting in less complex model and lower runtime.
- Methodology of static virtual ground modeling, while removing the limitations related to previous models; resulting in more accurate and faster model evaluation.
- We use SVM as a modeling method for leakage characterization and capacitance modeling of CMOS gates in power gated circuits. SVM classifier model is efficiently used for data generation.
- We also propose a methodology to estimate dynamic characteristics of V_{gnd} at virtual ground node, resulting in accurate energy estimation during mode transition in power gated circuits.

1.7 Thesis Outline

In Chapter 2, we present review of the work in the area of static leakage modeling of CMOS gates. Advantages and disadvantages of different models are compared with respect to model characterization time and model runtime. Importance of kernels and active learning methodology for efficient and accurate development of machine learning based models is elaborated. We also discuss previous work on modeling approaches for power gated circuits.

Chapter 3 discusses our proposed transistor stack based methodology for leakage modeling of CMOS gates. Common stacks are identified for subthreshold and gate tunneling leakage of different CMOS gates of a 20 gate standard cell library. Effective width estimation methodology is presented for the gates comprising parallel transistors or series of parallel transistors.Through various experiments, we show that our stack based models results in much less error for leakage estimation of gates with different input vector combinations.

In Chapter 4, kernel based SVM models are proposed for modeling of extracted transistor stacks in Chapter 3. We use efficient version of SVM i.e. Least Squares SVM, which takes less runtime than its previous version. Simulation results show that the SVM models are more accurate than EQ models and scaling based models. Different kernels are explored and best kernel with optimum tuning parameters are used in each SVM model using combined grid search based method and global optimization methods. Training samples are actively selected to prepare SVM models with minimum number of samples, resulting in less complex model and lower runtime.

In Chapter 5, we elaborate our methodology of static virtual ground modeling while removing the problems related to previous models. We use SVM as a modeling method for leakage characterization and capacitance modeling of CMOS gates in power gated circuit. SVM classifier model is efficiently used for data generation for reduction of time to generate training samples.

In Chapter 6, we explain our methodology to estimate dynamic characteristics of V_{gnd} at virtual ground node in power gated circuits. We also develop SVM based capacitance models for virtual ground node. These capacitance models are used in addition to leakage models and static V_{gnd} model to provide accurate dynamic V_{gnd} characteristics.

We conclude in Chapter 7 and various future research directions are discussed in this area.

Chapter 2

Leakage Current, Virtual Ground Voltage Estimation and VLSI surrogate modeling: A Review

For large System-on-Chip (SoC) designs, leakage contribution is more than 50% of the total power [28]. Power per unit area has been a threat for possibly leading to thermal runaway. Different floorplans significantly affect the temperature and leakage of large designs [28]. Supply, temperature and process variations may lead CMOS gates to operate at different and continuous ranges of voltage, temperature and process parameters than they have been designed for. Thus, it is necessary to build models which can provide quick estimation of leakage in any of these unpredictable conditions with high accuracy. Since the size of standard cell library is becoming very large for fine grained power and performance optimization in current technology nodes, it is highly desirable to reduce the number of models to achieve smaller characterization time. Thus, we analyze the previous leakage modeling approaches in the context of reduced model characterization time and higher accuracy.

2.1 Process Variation Aware Leakage Models

Gu *et. al.* [29] have used BSIM device model equations to estimate subthreshold leakage (I_{sub}) of a transistor stack consisting of one, two and three transistors as shown in Equation (2.1).

$$I_{s1} \; : \; I_{s2} \; : \; I_{s3} = 1.8(\eta V_{DD}/nV_T) \; : \; 1.8 \; : \; 1 \tag{2.1}$$

The current I_{sub} of transistor stacks with more than three transistors has been assumed to be zero. This model could not provide a method to handle other complex CMOS gates containing parallel transistors or stacks of parallel transistors such as AOI22, OAI32. Effects of 'ON' transistors in 'OFF' Pull down network (PDN) or Pull up network (PUN) have also not been considered. 'ON' transistors have simply been removed from the 'OFF' network and remaining 'OFF' transistor's leakage is calculated.

Consider a 4 input NAND gate with input '1110', where, bottom 3 NMOS transistors are 'ON' and a single NMOS transistor has '0' input. For '1110' input, NMOS network will be used to calculate I_{sub}. In this model, I_{sub} for '1110' input has been assumed to be same as single NMOS transistor with '0' input. For some of the conditions in Process-Voltage-Temperature-Width (PVTW) space, I_{sub} of 4 input NAND gate with input '1110' will be the same as I_{sub} of single NMOS transistor with '0' input and will be different for other conditions. This assumption of zero I_{sub} for more than three transistors in a stack and neglecting 'ON' transistors provide large error in I_{sub} estimation as represented in Figure 2.1. Here, 10% Gaussian distribution in process parameters and 10% Uniform distribution is assumed for supply voltage and temperature. Width of all transistors is taken as $28nm$. Predictive Technology Model (PTM) in $28nm$ technology is used to calculate leakage distribution with 20000 Monte Carlo simulations. Error in percentage is calculated using Equation (2.2), which varies from 0 to 220%. Negative sign indicates that the single NMOS transistor over-estimates the I_{sub} of NAND4 gate.

$$Error\,(\%) = \frac{I_{leak,n4-1110} - I_{leak,n1-0}}{I_{leak,n4-1110}} \times 100\% \tag{2.2}$$

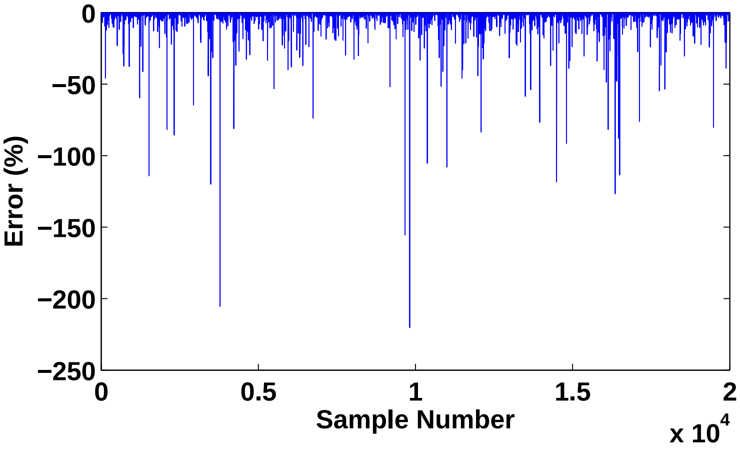


Figure 2.1: Error in percentage for I_{sub} of NAND4 gate with '1110' input is modeled by single NMOS transistor with '0' input.

Chen *et. al.* [30] have presented the analytical models for I_{sub} estimation based on BSIM device equation. This method first calculates the drain to source (V_{DS}) of second transistor from top node of the stack using Equation (2.3). This V_{DS} is used to find out V_{DS} of subsequent transistor ($i > 2$) using Equation (2.4). I_{sub} of complete stack is found out by using V_{DS_n} of bottom transistor.

$$V_{DS_2} = \frac{nkT}{q(1+2\eta+\gamma')} ln(\frac{A_1}{A_2} e^{\frac{q\eta V_{DD}}{nkT}} + 1) \tag{2.3}$$

$$V_{DS_i} = \frac{nkT}{q(1+\gamma')} ln(1 + \frac{A_{i-1}}{A_i} e^{-\frac{q}{kT} V_{DS_{i-1}}} + 1) \tag{2.4}$$

This model also does not account for the effect of 'ON' transistors in 'OFF' networks and is not capable of providing leakage calculation of complex CMOS gates. Runtime of the model is also very high due to iterative calculation of V_{DS}. Method in [31] first identifies all possible bias conditions of NMOS/PMOS transistors in CMOS gates. Leakage, I_{sub} of NMOS/PMOS transistor with these bias conditions are precharacterized and stored in a table. For a given input vector, state of each transistor is identified in the circuit. I_{sub} of complete CMOS gate for given input vector is calculated by adding the width scalable I_{sub} of each transistor as shown in Equation (2.5).

$$I_{CMOS-Gate} = \sum_{i=1}^{n} W(i) I_g(S(i)) \tag{2.5}$$

Here, $I_g(S(i))$, $W(i)$ are unit width leakage and width of i^{th} transistor respectively. Main disadvantage of this model is the inaccurate assumption of node voltages. This model assumes the drain and source voltages of transistors at either supply or ground voltages only, not accounting the stack effect. This assumption results in large error in leakage estimation of CMOS gates. However, this model accounts for the effect of 'ON' transistors but not accurately. Model proposed in [32] estimates the leakage of a CMOS gate considering I_{sub}, gate tunneling (I_{gate}) and band-to-band tunneling (I_{BTBT}) leakage components. These leakage components have been characterized by accurate BSIM device equations. NMOS/PMOS transistor is first modeled as sum of current sources (SCS), then 'OFF' stack of a CMOS gate is represented as an equivalent SCS model and Kirchhoff Current Law (KCL) is applied at internal nodes to obtain internal node voltages and corresponding leakage components. These leakage components of each transistor have been used to estimate leakage of complete CMOS gate. Large runtime is the main disadvantage of this approach due to the internal node voltage calculation during leakage estimation of CMOS gate.

Lee *et. al.* [6] first characterized the I_{sub} of single unit size transistor ($I_{sub,1}$), then scaled it according to the actual transistor size (S_t) and number of 'OFF' transistors

(k) in the stack. I_{sub} of a 'OFF' stack is given as in Equation (2.6).

$$I_{sub,k} = I_{sub,1} * S_k * S_t \tag{2.6}$$

Both, $I_{sub,1}$ and S_k are precharacterized using SPICE simulation for stacks with transistors having different sizes. I_{gate} is also modeled through a simple analytical model. Interaction of I_{sub} and I_{gate} components is also considered to accurately estimate the leakage of CMOS gates. Effect of 'ON' transistors is still not accurately considered, resulting in large error. Leakage of CMOS gates with parallel transistors is also considered. Two parallel transistors with same input are replaced by a single transistor with effective width as simple summation of both transistor widths. However this is not true due to lithographic variations in width [33]. When one transistor is 'OFF' and another is 'ON', 'OFF' transistor is simply removed without assuming any leakage contribution. Thus, parallel transistor's leakage is not estimated accurately by this model. Yang *et. al.* [7] simplified the model in [6] by replacing stack size and transistor size with a constant factor (A) for each input vector of a gate as in Equation (2.7).

$$I_{CMOS-Gate} = I_{sub,1} * A \tag{2.7}$$

In this model, calculation of scaling factor with respect to input applied is not accurate. For example, I_{sub} of NAND3 gate with input '011' ('0' is applied to bottom transistor) and '101' ('0' is applied to middle transistor) is calculated using single NMOS 'OFF' transistor, assuming zero drop across 'ON' transistors. In fact, 'ON' transistor connected to output node will have one threshold voltage drop across it, reducing the voltage applied to remaining stack. Thus, 'ON' transistors should be taken care of while being removed from the 'OFF' networks.

Models presented in [6, 7, 29, 30, 31, 32] first characterize the leakage of single

NMOS/PMOS ‘OFF’ transistor. Then, these characterized models are used to find out leakage of complete CMOS gate by either scaling or finding the internal node voltages. Characterization time of the models is less because of consideration of only two models. However, these works do not consider process variations. In presence of process variation, scaling factors will itself be a non-linear parameter, which increases the characterization time to develop leakage models. Calculation of internal node voltages for the set of process parameters makes the models very slow. The assumptions about ‘ON’ transistors in presence of process variations is also inaccurate, causing large error in leakage estimation. Mean (μ) and standard deviation (σ) need to be calculated to capture the effect of process variations on leakage of a CMOS gate. For example, effect of length variation ($\triangle L$) on I_{sub} can be captured by solving the expectation integral as in Equation (2.8).

$$E[I_{sub}] = \int_{-\infty}^{+\infty} I_{sub}(\triangle L) \frac{1}{2\pi\sigma_L} e^{\frac{\triangle L^2}{2\sigma_L^2}} \, d(\triangle L) \tag{2.8}$$

Here, $\triangle L$ is the Gaussian random Variable (RV) with zero mean and σ_L^2 variance. Analytical solution of Equation (2.8) depends on the functional form of the leakage representation in terms of random variables. Other parameters such as V_{th}, T_{ox} also affect the leakage of a gate, thus physics based models (BSIM equations) result in a complex equation to relate leakage as a function of these RV’s. This in turn results in a highly complex solution in Equation (2.8), making these models unsuitable for CAD applications. To overcome this problem, empirical models have been proposed. Empirical models captures the leakage dependency on process parameters by fitting SPICE simulated data in predefined form such as exponential linear (EL) or exponential quadratic (EQ) by minimizing the least square error between results of SPICE and model simulation data. A commonly used EL or EQ model is shown in Equation (2.9).

$$I_{sub} = I_0 e^{-(c_1 \triangle L + c_2 \triangle L^2)} \tag{2.9}$$

Here, I_0, c_1 and c_2 are constants, obtained by fitting SPICE simulated data. The variation in length ($\triangle L$) can be expanded to account the effect of inter-die and intra-die process variations. Other process parameters like V_{th}, T_{ox} can also be included in the above model by adding more number of terms corresponds to these parameters. EQ model has been used for full chip leakage estimation using sum of random variables [21], sum of orthogonal polynomials [34], Random gate based method [35] and Karhunen–Loeve expansion (KLE) based method [36]. If there are M gates and i^{th} gate has k_i inputs, then a total of $\sum_{i=1}^{M} 2^{k_i}$ leakage models are required. The important issue here is to reduce the number of leakage models required to reduce characterization efforts without sacrificing accuracy. Another advantage of reducing the number of models is to reduce the runtime complexity due to the use of reduced number of leakage components to be added for full chip leakage estimation. For example, method in [21] uses the dominant I_{sub} and I_{gate} concept among various input vector combinations of CMOS gates, which only adds the leakage of only dominant components. Due to large variation space, this assumption may provide large error in leakage estimation. Equivalence concept between different input vectors of CMOS gates is also used. For example, I_{sub} of NAND4 gate with '1110' input, NAND3 with '110', NAND2 with '10' is assumed to be same as INV with '0' input. This assumption also provides inaccurate results in leakage estimation as already illustrated in Figure 2.1. Thus, reduction in the number of characterized models is very important for characterization time as well as model evaluation time.

EQ models can provide accuracy for limited number of process parameters with only Gaussian assumption. As in advanced technology nodes, due to short channel effects and lithographic limitations, the number of process parameters to be considered are increasing and also, these process parameters may assume any distribution other than Gaussian distribution. Authors in [37] evaluate the accuracy of previously developed leakage models in [21] [38] [39] using BSIM4 transistor model [33]. State-of-the art methods denotes the leakage in the EQ and 3^{rd} order polynomial (poly3) form as function of different varying parameters. These forms of the models have been chosen to simplify the analysis to find out analytical solutions for full chip leakage power. However, these analytical models were characterized using BSIM3 model [40]. More advanced BSIM4 model accounts several short channel effects (SCE) and other secondary effects, which were not considered in BSIM3 models. These physical effects impose high non-linearity into the model, which result in non-exponential linear behavior of leakage current. The average error in mean and standard deviation can go up to ∼20% and ∼40% respectively. Recently, BSIM6 model has also been developed for more advanced technologies [41], which will further increase the non-linearity into leakage model due to various secondary effects. Hence, traditional EQ and poly3 models can not be used in leakage modeling of CMOS gates.

Quadratic models have also been proposed to analyze the effect of voltage and temperature on leakage [42] but without considering process variations. Equation (2.10) represents the generalized way of considering voltage and temperature effect as used in [42]. Authors in [43] simultaneously consider process variations and temperature effect but do not consider the variations in supply voltage. coefficients related to temperature variations are assumed as zero.

$$I_{leak}(\triangle V,\triangle T)=I_{leak}(0,0)*(1+b_1\triangle T+b_2\triangle T^2+a_1\triangle V+a_2\triangle V^2+c_1\triangle T\triangle V) \quad (2.10)$$

These quadratic models are second order Taylor approximations around mean values of voltage and temperature variations. Previous work in [42, 43] does not simultaneously consider variation in the process parameters, voltage and temperature.

Quadratic models are only valid in sub-space of PVT space. These works also do not discuss about the reduction in number of models and hence, characterization time can be be very large to generate models for different input vectors in PVT space.

Recently in [44], a voltage based leakage current estimation method is proposed which identifies the intermediate node voltages of transistor stacks and scales linearly with respect to supply voltage. A linear model has been developed for intermediate node voltage per input vector. Two main disadvantages of this work are as follows - (i) Intermediate node voltage itself can be highly non-linear function of process parameters and temperature. Due to short channel effects, scaling models will not be linear also. Thus, this method does not scales well in presence of process variations. (ii) Total number of models including intermediate node voltage models and scaling models can be very large. For example, in this thesis we consider 20 gate standard cell library for which 176 input vector combinations are possible. Let us assume that if stack in each gate has average 2 intermediate node voltages then 2 scaling models will also be required, one for each intermediate node voltage. Total of 176*2 = 352 models will be required to estimate leakage of complete standard cell library, which would increase the model characterization time significantly.

On the other side, sampling based methods for full chip leakage estimation techniques do not rely on precharacterized leakage models, instead use the SPICE simulation to find out leakage of a gate for a sample generated by any sample generation methodology such as Monte Carlo Technique [45]. One of the disadvantages is the requirement of large number of samples to find out converged μ and σ of full chip leakage, resulting into large runtime for leakage analysis. Various variance reduction techniques have been proposed to reduce sample size [46]. Veetil *et. al.* [47] proposed an efficient sampling method to reduce the number of samples for full chip leakage estimation which further reduces the runtime without sacrificing accuracy. The disadvantage of their approach is that they do not provide any means of reducing the number of models to be characterized. Also, it is very difficult to scale the model in PVTW space. Generally, sampling based methods use SPICE simulation to evaluate samples which may increase the runtime for full chip leakage estimation.

However, CMOS gate characterization is a one time effort, stack modeling methodology helps in increasing the accuracy at the cost of increased characterization time. For statistical leakage characterization of gates, it is more efficient to characterize different kinds of stacks instead of characterizing every gate for each input vector. Due to large process variations in nano-scale technologies and presence of high non-linearity due to temperature and supply voltage variations, surrogate modeling techniques such as Neural Network, Support Vector Machine assume greater significance while modeling non-linear performance parameters. This type of technique requires extensive samples in order to train the models. However, once the models are trained, these models provide more accurate results than models based on BSIM and empirical equations. One of the disadvantages is that these models are slower than empirical equation based approaches. For fast full chip leakage estimation, same stack model can be used multiple times, which helps to reduce leakage estimation time [21].

Table 2.1: Comparison of leakage estimation techniques

Reference	*Scalable (PVTW space)*	T_{model}*	*Accuracy*$	*Runtime*#
[29], [30], [48], [31]	No	Very Low	Very Low	Very High
[32]	No	Very Low	Very High	Very High
[6], [7]	No	Very Low	Very Low	Very High
[21], [34],[49]	Yes	Very High	High	Very Low
[47]	No	Very High	Very High	Very Low
[2]	Yes	Low	Low	Low
[44]	No	Very High	Very High	Low

T_{model}→ Model Characterization time,
* Computation Based on total number of models Characterized.
$ Accuracy under PVTW space
# Runtime for benchmark circuits under PVTW space

In this context, Viraraghavan *et. al.* [2] have used the neural network to model leakage through a stack. The main drawback of this method is in the computation of scaling factors - only one scaling factor per input vector per gate is used for all PVT space. This assumption is not valid due to non-linear dependence of leakage power on PVT parameters. Maximum error in μ and σ reported is 20% for a gate in $130nm$ technology [2]. The error would increase for circuits implemented in latest technology nodes. Another limitation of model presented in [2] is that it does not use width of transistors on a stack as a parameter for modeling stacks, allowing leakage calculation for transistors with fixed set of widths. Neural network suffers from the condition of being trapped in local minima due to the use of gradient descent algorithm to calculate weights in the trained model. These models also suffer from over learning i.e. high error for unseen data [27]. Table 2.1 shows comparison of previous methods in literature with our method w.r.t. scalability of the model, model characterization time, accuracy and runtime.

2.2 VLSI Surrogate Modeling

The standard transistor-level simulation based techniques consume large time to optimize the circuit. From the perspective of highly competitive market, the time to market for a VLSI product design must be very short. Thus it is not infeasible to optimize the circuit using these kinds of simulators. Various macromodeling techniques have been developed to replace highly complex models (e.g. SPICE) with less complex models. But the design process is still dependent on the circuit simulator (e.g., SPICE) and hence MC simulation is still very time consuming [50]. Metamodeling, also known as Surrogate modeling, is an alternative to macromodeling that is more flexible, easier to simulate and optimize than a macromodel. The types of metamodel include polynomials, Splines, artificial neural networks (ANN), support vector machines (SVM), genetic programming, Kriging methods, and Gaussian processes [51].

To reduce the optimization time, authors in [51] developed a polynomial surrogate modeling approach for accurate and fast optimization of oscillator circuits. While in [52], technology independent polynomial surrogate modeling flow is presented to optimize different components of a PLL for power. In [53], design parameter space of OP-AMP circuit is first partitioned into different sub-regions and then low-order polynomials used to fit each region.

It has been shown that regression based non-linear modeling methods such as Artificial Neural Networks (ANN) [2] and Support Vector Machine (SVM) [54] are more accurate than linear regression based techniques e.g. Linear regression with regularization, Logistic regression etc. These linear regression based techniques can not model non-linear performance parameters such as leakage in PVTW space.

ANN surrogate models used in [55] avoid such creation of separate model for each sub-region as used in [53]. In [56], Support Vector Machine (SVM) based surrogate modeling approach is presented to replace expensive circuit simulations. In [57], microwave components are modeled by a surrogate multivariate mathematical model. Authors in [58] developed a surrogate modeling approach for statistical wire-length estimation. In [59], expensive electromagnetic (EM) simulations are replaced a surrogate model developed using ANN and Gaussian processes. Gaussian process is also used with memetic optimization technique for performance analysis and optimization of differential amplifiers [60].

Failure probability analysis of robust circuits needs statistical rate event or high-sigma analysis, which poses many challenges to the statistical analysis based on Monte Carlo (MC) simulation. Because it may require very large number of transistor-level simulations to be performed. Statistical blockade method [61], first develops liner SVM classifier from some SPICE simulated samples. Later this classifier selects most likely-to-fail samples out of the samples generated using joint PDF of process parameters. The samples which have high probability of failure, are only simulated though SPICE. However, this linear classifier results in high error in case of discontinuous and non-linear failure regions [62]. Also the efficiency of this approach highly depends on the classifier accuracy. In case of high-dimensional variation space, a large number of SPICE simulations are required to develop an accurate classifier, which makes this method less efficient [63]. In this context, our proposed method in Algorithm 4.1 can be used to optimally select kernel parameters for classifier development. While Algorithm 4.2 can be used for active selection of samples from high error regions. In contrast to the work in [61], our aim is to develop accurate model in complete process variation space, not in tail region only.

In [64, 55], authors have suggested use of artificial neural network (ANN) to model performance parameters of several different op amp topologies. A standard two layer feed forward neural network is used to model each of several op amp performance parameters. In order to obtain good generalization and accuracy on training and validation data sets, number of hidden layer neurons was manually adjusted. A hyperbolic tangent Sigmoid function [64] and Bayesian regular training [55] have been used as the transfer function for all hidden layer neurons and a linear transfer function was used for all output layer neurons. The training pattern set has been used to train the networks using back propagation algorithm. The generated models are able to capture non-linear behavior of performance characteristics of op amp with good accuracy. They are quite efficient and provide a substantial saving of time in situations. However, the methodology has a drawback that a large number of sample points are required to accurately map the behavior of a circuit. As the dimension of circuit increases, due to grid sampling, number of sampling point increase exponentially. In our work, we have developed an algorithm to efficiently select training samples from process variation space, which highly reduces the model development time.

Also, traditional perceptron based ANN models suffer from limited model generalization ability, generating models that can easily lead to over fitting of data [65]. In contrast, Support Vector Machine (SVM) is based on structural risk minimization (SRM), which is superior than Empirical Risk Minimization (ERM) employed by ANN because SRM minimizes the maximum error in predicting the output, whereas ERM minimizes the maximum error on training data. Thus, SVM has better ability to generalize with larger data size [66]. SVM maps input data into high dimensional feature space to create a optimal separating hyperplane using kernel functions. SVM uses quadratic programming to solve this problem, instead of gradient based optimization used by traditional Neural Network approaches [67]. Neural Network suffers with the problem of being trapped in multiple local minima. SVM can be used for two types of problems i.e. Support Vector Classification (SVC) and Support Vector Regression (SVR). SVM is the class of kernel-based learning methods which maps the input data into high dimensional feature space, where each point denotes one feature of data points. Kernel functions are then operated in feature space by simply evaluating the inner products between all data pairs instead of computing the coordinates of data points which makes it computationally cheaper than the computing coordinates of data points. Kernel trick makes SVM to model highly non-linear relation between input and output in high dimensional space.

Use of kernels with optimum tuning parameters is very important in building accurate and efficient SVM models. The methods for finding the optimal SVM kernel tuning parameters can be classified as heuristic method, grid search method, numerical gradient based optimization and evolutionary search method. Heuristic methods have been applied for finding optimal tuning parameters in [68]. The solution obtained using this method can be suboptimal [69]. Another disadvantage of this method is that efforts required to formulate the equations related to SVM are high and some approximations are also made to frame equations in the particular format which can reduce the accuracy of the solution obtained. Gradient based optimization techniques such as gradient descent and quasi-newton methods which require the gradient of the function being optimized and hence, can be trapped in local minimum without converging to the optimal solution. This leads to the use of search based techniques for kernel parameter optimization. In this direction, Grid search based technique [70] is simple because it only require a grid in tuning parameter space. At each value of the grid, model is trained and tested and the values with least error are selected as optimal tuning parameters. Time and accuracy trade-off depends on the grid density. The solution obtained using this method can also be suboptimal because derived values of optimal tuning parameters may not reside in the grid points. The main disadvantage of this method is to define the grid density so that optimal tuning parameters can be found. There can be multiple local minima in the tuning parameter range [66]. Evolutionary search methods such as Genetic Algorithm (GA) and Differential Evolution (DE) assume the optimization problem as a black-box function and optimize it by iteratively improving the solution without making any assumption about the problem under consideration. These methods do not require the gradient of fitness function i.e. problem need not to be differentiable and are best suited for noisy, multi-modal and multi-dimensional problems. Pandit *et. al.* [71] have used the GA for finding the optimal tuning parameters. Differential Evolution has advantages over GA with respect to solution convergence as it uses advanced mutation and cross-over strategies [72]. One of the important feature of DE is self-organizing scheme, in which any existing population vector is changed by calculating the difference of two randomly chosen population vectors from current population while in traditional methods presumed probability distribution function decides the change in population vectors. However, other search based global optimization techniques can be used. GA and DE methods attempt to search the entire space randomly which may take more time to reach the optimal solution.

Active learning is a supervised learning method for selection of new samples in the input space which are combined with the previous training data-set to further improve accuracy of the model. In Active learning [73], learner has a control over supplied training data set, which reduces the sample size and increases accuracy. Learning starts with fewer training samples, further samples are added according to the requested query to achieve a goal of low error rate with minimum training samples. Authors in [74] have used active learning with SVM, based on version space reduction for text classification. While in [75], active learning is applied to optimize expected future error. In our methodology, we use active learning process to generate new training samples around maximum error sample to achieve desired accuracy. Maricau *et. al.* [76] uses uncertainty predictor $D(.)$ based on the distance between input to input, output to output and model to model. Vaylon *et. al.* [77] uses the dominant $'alpha'$ (α, SVM hyperparameter) of previous model to generate the new samples. Maricau *et. al.* [76] uses uncertainty predictor $D(.)$ based on the distance between input to input, output to output and model to model. Vaylon *et. al.* [77] uses the dominant $'alpha'$ (α, SVM hyperparameter) of previous model to generate new samples.

These SVM models should be developed using a sufficient number of training samples, generated from transistor to system level models. These kind of models have already been adopted in various fields of VLSI research such as Power and delay estimation of using Wavelet Neural Network [78] (2500 training samples are used to prepare models), Optimal memory technology estimation at each non-volatile memory hierarchy level through design space exploration [79] (require 3000 system simulation samples for accurate and efficient models), Board-level functional diagnosis and repair [80, 81] (require 1000 board level simulations for efficient and accurate models), Area, power and performance modeling of Network on chip (NOC) router [82](Adaptive sampling method is adopted instead of brute-force Latin Hypercube Sampling to reduce model's training sample size and 1024 samples are used to develop accurate and efficient clock tree synthesis model using SoC Encounter). In recent years, post-CMOS devices in hybrid technologies such as Carbon Nanotube FET (CNFET) [83] and Nano-Electro-Mechanical-System (NEMS) [84] have been used to design the power gated switch to remove drawbacks of CMOS based power gating methodology. SVM based methodology can be a generalized methodology for all hybrid technologies where non-linearity of the model in terms of the power gating switch parameters can be very high. Thus, we can say that developing SVM regression based black box models is an important and emerging area in the VLSI field. In comparison to analytical regression based models and other non-linear models, SVM model's accuracy and efficiency also depends on efficient sampling techniques. Authors in [85] argued that ANN and SVM based models can outperform regression based exponential quadratic leakage models in terms of the number of training samples with less characterization time and higher accuracy. However, above mentioned previous work does not focus on reduction of number of training samples to prepare the model. The number of training samples directly affect the runtime of models. Thus, we need to develop models with minimum number of training samples for a given accuracy level. In this direction, authors in [86] show that the higher number of samples are required to accurately fit a highly non-linear model. Various sampling methods such brute force sampling, random sampling, Latin-Hypercube sampling etc. have been used to generate a fixed number of training samples. These training samples are then used to develop regression model. LHS sampling method is shown to be better than other sampling methods in terms of accuracy.

2.3 Models for Virtual Ground Voltage Estimation

With the rapid scaling of MOSFET technologies, the contribution of leakage power to the total power is increasing. Power gating technique has been applied for reducing leakage power. Power gating is coarse-grained generalization of MTCMOS technique in which high threshold transistor is inserted in pull-down and/or pull-up network as a footer in ground gating case or header in supply gating case or combined gating case. In ground gating, when the sleep transistor is 'OFF' during the standby mode, the leakage current flowing through logic cluster (I_{LC}) charges the virtual ground node and reduces the effective V_{dd} to ground voltage across logic cluster. Various issues such as - performance degradation, ground bounce noise, wake-up energy consumption, data retention, virtual ground (V_{gnd}) or virtual supply voltage etc. are needed to be considered before applying it to the logic circuits.

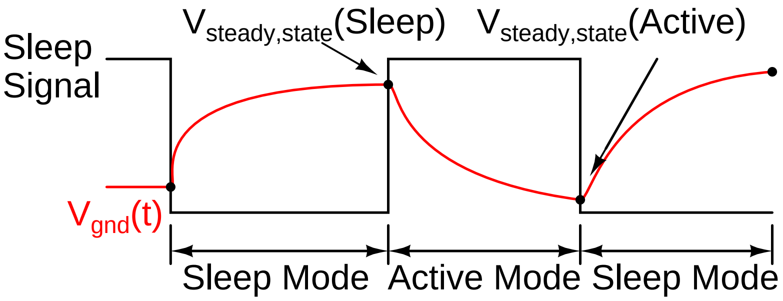


Figure 2.2: Typical sleep and active mode cycles in ground-gated circuits

Recently, for many variants of power gating technique, V_{gnd} has been the basic parameter to be analyzed. Singh *et. al.* [87] have represented the leakage current of logic cluster and footer transistor as a function of V_{gnd}, then both currents are made equal to find the exponential linear model of V_{gnd} as a function of design parameters of logic cluster and footer transistor. Kim *et. al.* [88] have proposed the intermediate strength power gating which provides the option of selecting a particular value of V_{gnd} from the list of many values depending on the power and performance constraints. Sinkar *et. al.* [89] have proposed to clamp the V_{gnd} at a specific value under temperature variations and reliability constraints for a fixed percentage of leakage reduction with enhancing the reliability of the circuit. The V_{gnd} calculated in [87, 88, 89] is a static voltage but it is a dynamic characteristic whose value is increased from lower to higher steady state value after the circuit is gated. Xu *et. al.* [4] have estimated the dynamic V_{gnd} during the mode transition which is further used to estimate energy consumed due to the transition of V_{gnd} value, allowing the fine grained optimization of power gated circuits. Tovinakere *et. al.* [5, 90, 91] have derived the semiempirical model for dynamic V_{gnd} estimation. Xu *et. al.* [4] have used dynamic V_{gnd} characteristics to obtain a trade-off between energy consumed in wake-up to sleep mode transition and leakage saving in power gating mode. The consumed energy is highly dependent on time dependent V_{gnd} characteristics during mode transition. Dynamic characteristics of V_{gnd} will also vary according to the input vector applied to logic cluster and design parameters of the footer transistor. Thus, we need a fast and accurate methodology that can estimate V_{gnd} with respect to the variation in these parameters.

In our work, initially we concentrate on static virtual ground voltage estimation, then model is extended for dynamic V_{gnd} estimation. The static V_{gnd} estimation is an important parameter in the case of standby mode of operation of CMOS circuits in following cases.

1) static V_{gnd} model can be used for minimum leakage vector (MLV) estimation to reduce the leakage power during standby mode [92, 93, 94]. The same problem should also be solved for power gated circuits because MLV can be different for non-power gated circuits. Since, leakage will vary according to the design parameters of footer transistor, so we need a quick exploration of design parameter space for each input vector.

2) Variation of V_{gnd} w.r.t. time in sleep and active mode is shown in Figure 2.2. Accurate calculation of energy during mode transition require steady state values in sleep and active mode as starting points. Previous work in [4, 5] generally assumes

the starting point as V_{dd} or $0\,V$ in active and sleep mode respectively. Our results show that the steady state values are a strong function of input vector, number of gates in the circuit as well as footer transistor parameters. Hence, V_{gnd} value can be largely different than assumed values.

3) In leakage - delay trade-off optimization of power gated circuits [95], V_{gnd} value should be high to achieve high saving in leakage during sleep mode, which requires minimum width and high threshold footer transistors. But these set of parameters result in high delay degradation due to increased resistance i.e. drop across footer transistors. Static V_{gnd} models can be used as input to these optimization methods.

4) To mitigate the effect of Negative Bias Temperature Instability (NBTI), ground gating has been used in sleep mode [95]. When footer transistor goes to 'OFF' condition, V_{gnd} and internal node voltages of CMOS gates increased up to higher value ($V_{steady,state}(sleep)$) as shown in Figure 2.2, thus changes the input voltages of PMOS transistors to V_{gnd} value. It reduces the gate to source voltage (V_{gs}) and thus drives the PMOS transistors in recovery mode. Any error in V_{gnd} estimation may result in wrong prediction of trade-off calculation between leakage saving and amount of NBTI recovery in sleep mode.

Thus, there is clearly a need of developing static V_{gnd} model efficiently and model should also be highly accurate and fast. One another use of static V_{gnd} model is that it can be used in the derivation of dynamic V_{gnd} model for enhancing the accuracy and efficiency of the model. Previous models provides large error in V_{gnd} estimation because of inaccurate leakage models for power gated circuits and assuming inaccurate voltage conditions at the input of CMOS gates in power gated circuits.

2.3.1 Previous V_{gnd} models incorporating inaccurate leakage models

To calculate static V_{gnd}, authors in [4, 87] have represented the leakage current of logic cluster (I_{LC}) as an Exponential Linear (EL) function of V_{gnd} which assumes the same voltage at the input of equivalent transistor and the virtual ground as shown in Figure 2.3. Since, the EL behavior of I_{LC} with respect to V_{gnd} is not valid rather it depends upon the type of circuits and their corresponding input vectors which may result in higher error for leakage modeling of logic cluster and consequently in static V_{gnd} estimation.

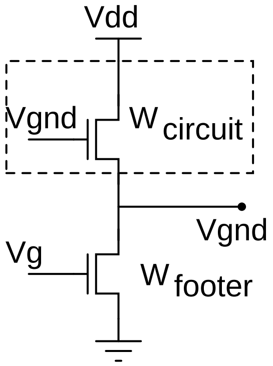


Figure 2.3: Equivalent circuit for virtual ground (V_{gnd}) model

From Figure 2.3, static V_{gnd} can be estimated by equalizing the leakage current of logic cluster and footer transistor using Equation (2.11).

$$I_{LC}(logic - cluster) = I_{footer}(footer - transistor) \tag{2.11}$$

Replacing $I_{LC}(logic-cluster)$ and $I_{footer}(footer-transistor)$ in the above equation as a function of V_{gnd} gives a equation of V_{gnd} as a function of design parameters of logic cluster and footer transistor. From [96], sub-threshold leakage current of a single

‘OFF’ transistor can be represented as in Equation (2.12).

$$I = A.e^{1/mV_T(V_g - V_s - V_{th0} - \gamma^{'} V_s + \eta V_{ds})}.(1 - e^{-V_{ds}/V_T}) \tag{2.12}$$

$$\text{with } A = \mu_0 C_{ox}^{'} \frac{W}{L_{eff}} (V_T)^2 c^{1.8} c^{-\triangle V_{th}/\eta V_T}$$

Here, V_{th0} is the threshold voltage at zero body bias, V_T is the thermal voltage, $\gamma^{'}$ is body bias coefficient and η is the DIBL coefficient. In [87, 4], for $V_{ds} \gg V_T$, the term $(1-e^{-V_{ds}/V_T})$ in Equation (2.12) is neglected. Authors in [87], represented above equation as a EL function of V_{gnd}, logic cluster and footer transistor design parameters while in [4], it is denoted only in terms of virtual ground voltage (V_{gnd}) as shown in Equation (2.13) and Equation (2.14) respectively.

$$I_{leak} = I_0 \frac{W}{L} 10^{(-V_{th} - (\eta V_{gnd})/S_S)} \tag{2.13}$$

$$I_{leak} = \hat{I_N}.e^{-K_N V_{gnd}} \tag{2.14}$$

Neglecting the term $(1-e^{-V_{ds}/V_T})$ in Equation (2.12) for higher values of V_{ds} i.e. low source voltage for fixed drain voltage (V_{dd} in case of logic cluster) causes error in estimating sub-threshold leakage current of a transistor for comparable values of V_{ds} and V_T.

The authors in [90] [5] represent leakage current as a polynomial function of degree N in terms of V_{gnd} as shown in Equation (2.15).

$$I_{leak} = \sum_{j=0}^{N} p_j V_{gnd}^{j} \tag{2.15}$$

These type of models can result in very large error in estimating the leakage current of logic cluster. However, for higher values of N, accuracy is improved but the complexity will be increased for obtaining the virtual ground voltage equation. Hence, traditional EL and poly3 models can not be used in leakage modeling of CMOS gates.

2.3.2 Previous V_{gnd} models based on inaccurate assumption of input voltages

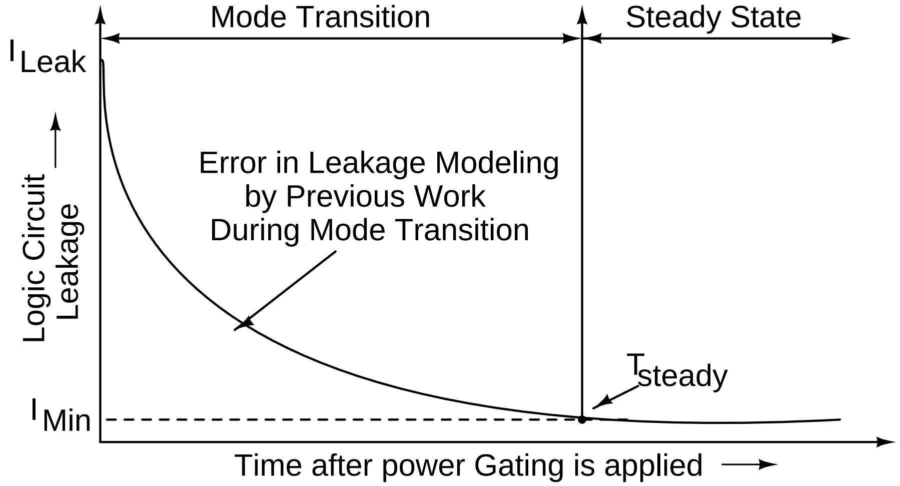


Figure 2.4: Error of previously developed leakage models during mode transition

Figure 2.4 shows the characteristics of logic circuit leakage after power gating is applied. Previous model in [97] has only considered steady state leakage to calculate energy saved in power gating. This assumption is only true if idle time is very high compared to mode transition. Authors in [98] have reduced this error by multiplying the energy saving in standby mode with a constant factor. But this assumption is not true due to change in the characteristics of leakage during mode transition depending on the conditions of circuit. Other models [99, 100] are high level models and do not consider the effect of circuit topologies and input states. In high performance circuits, idle time may not be very high compared to number of mode transitions. More accurate models are required to calculate the overall energy saved by the circuit in idle time considering the energy consumed by mode transitions. Authors in [4, 5] model the logic circuit leakage during mode transition considering the effect of input states and circuit topologies. But these models do not consider the accurate input voltage conditions which depends upon the virtual ground voltage. They consider the input voltage of the circuit as a V_{gnd} for the whole voltage range varying from $0\,V$ to V_{dd}, however, it is only true for the lower values of V_{gnd} in ground gating case. for higher values of V_{gnd}, both pull down network (PDN) and pull up network (PUN) are 'OFF', resulting in the output voltage of that gate settled in between V_{dd} and V_{gnd}. This makes the input voltage of other gates different than the V_{gnd}, thus resulting in large error V_{gnd} estimation.

Chapter 3

Process Variation Aware Leakage Models: A Transistor Stack Based Approach

CMOS gate characterization is a one time effort, wherein transistor stack modeling methodology helps in increasing the accuracy but at the cost of increased characterization time. While characterizing statistical leakage of gates, it is more efficient to characterize different kinds of stacks instead of characterizing every gate for each input vector. Due to significant process variations in UDSM technologies and presence of high non-linearity due to temperature and supply voltage variations, surrogate modeling techniques such as Neural Network and Support Vector Machine assume greater significance, especially for modeling non-linear performance parameters. A large number of samples are required in order to train such models. However, these trained models provide more accurate results than models based on BSIM device equations and empirical equations. One of the disadvantages is that these models are slower than look up table based approaches. For full chip leakage estimation, a characterized stack model can be used multiple times, which helps to significantly reduce leakage estimation time. In this context. We have extended the leakage model to include transistor width as an additional parameter. Thus, leakage power of a gate i with an input vector ϑ applied, can be given as follows.

$$Y_i^{\vartheta} = f_i^{\vartheta}(W_i,\, GP,\, LP_i,\, T,\, V_{dd}) \tag{3.1}$$

Here, $W,\ GP,\ LP,\ T,\ V_{dd}$ are defined as width of transistors on stack, global process parameters, local process parameters, temperature and supply voltage respectively. The advantage of adding width is that it helps to reduce the number of characterized models other than basic stack models. These models need to be developed for the gates having parallel transistors or series of parallel transistors. We propose an approach for efficient and accurate estimation of leakage using transistor stacks as it avoids calculation of scaling factors as reported in [2]. The proposed approach uses a little bit more number of basic stacks, removing dependency on scaling factors while computing leakage current. Thus, design efforts get reduced because of not using scaling factors, and subsequently model characterization time for the stacks is also reduced. Our approach estimates leakage at any given V_{dd}, T and width of transistors on a stack. The proposed methodology is a generalized one, which may be applied to leakage characterization of post-CMOS devices i.e. FINFET, CNTFET and others. The novel contributions of the proposed methodology are as follows.

- More accurate reduced set of stack based Subthreshold (I_{sub}) and gate tunneling leakage (I_{gate}) models are developed. Our methodology requires smaller number of models for leakage characterization of 20 CMOS gate standard cell library across 176 input vector combinations in a complete Process - Voltage - Temperature - Width (PVTW) space.

- We combine the parallel transistors under the influence of same input and replace it with a single transistor of effective width. Thereafter, the precharacterized basic models can be used, thus, requiring smaller number of models for the overall leakage calculation of the standard cells as compared to [2].

- A novel methodology for Effective width calculation is developed for the parallel transistors in CMOS gates with and without considering process variations, resulting into higher accuracy than the previous model [3].

3.1 Stack based Models for Basic CMOS Logic Gates

3.1.1 Subthreshold Leakage Modeling

In this work, we assume maximum stack size of four, as higher order stack increases the delay of a gate due to increased logical effort. We modify conventions used in [2] for labeling stack parameters, which is shown as follows.

{stack type}{stack size}{leakage type}/{input to the stack}

Here, stack type indicates whether it is an NMOS stack (n) or a PMOS stack (p). Stack size represents the number of transistors on a stack. Leakage type denotes the subthreshold (s) or gate tunneling (g) leakage. Least Significant Bit (LSB) of input vector is applied to the transistor, which is closest to the output. The conventions used to model leakage are based on series transistors and input vectors, not parallel transistors. If any set of parallel transistors is found in a gate, it is first converted into a stack by combining parallel transistors and then precharacterized stack is used for this equivalent stack. I_{sub} leakage of a 'OFF' stack can be calculated by estimating the current flowing into the ground terminal through NMOS transistors whose source is connected to the ground [32]. I_{sub} of a gate for a given input vector is calculated from 'OFF' network only because 'ON' network has $\sim 0\,V$ drop across it.

Common stack models for I_{sub} estimation are extracted based on the current flowing from drain to source of transistors in 'OFF' PDN or PUN network. Let us consider

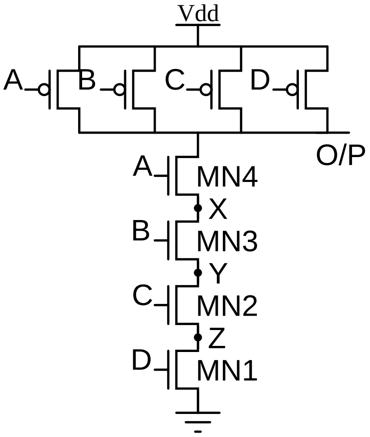


Figure 3.1: 4-input NAND Gate

a four input NAND gate as shown in Figure 3.1. We now illustrate leakage current estimation using basic stacks under the influence of each possible input vector.

Input vector (0000): In this state, all four NMOS transistors are turned 'OFF' while all four PMOS transistors are turned 'ON' and treated as short circuits. Thus to estimate 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 is referred as n4s/0 according to conventions used for stack representations.

Input vectors (1000 / 0100 / 0010): In this case, the top most transistor, MN4 has its gate connected to ground voltage and one of the other 3 NMOS transistors has gate connected to V_{dd}. Three PMOS transistors are fully turned 'ON', thus the I_{sub} is to be calculated from NMOS 'OFF' network. Similarly, one NMOS transistor that has its gate connected to V_{dd} behaves as a short circuit and can be removed from the NMOS stack. Thus, to predict the leakage for this set of input combinations, 3 transistor NMOS stack is modeled with all inputs grounded i.e. we need to model an n3s/0 stack. Note that the same n3s/0 model will be used to predict the leakage of a NAND3 gate with input '000'. 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 need to scale the currents by an appropriate factor to account for this width difference. But our experiments in Section 3.2.1 show that we do not need to calculate the scaling factors because the mean (μ) and standard deviation (σ) calculated from the n3s/0 stack is almost same as μ and σ of n4s/2, n4s/4, n4s/8 stack in complete PVTW space.

In 'ON' PUN network for all three input vectors, one PMOS transistor is 'OFF' and appears in parallel with three 'ON' PMOS transistors. These 'ON' transistors make drain and source voltage equal, resulting in drain to source voltage (V_{ds}) to zero. However, PUN consists of 'OFF' transistor but zero V_{ds} results in zero I_{sub} across 'OFF' transistors also. Same analysis is applicable for other input vectors also except '1111'.

Input vector (0001): Here, again I_{sub} is to be calculated from NMOS 'OFF' network. As the gate of the top most transistor MN4 is connected to V_{dd}, the drop across this transistor is high. MN4 transistor will have one threshold voltage (V_{th}) drop across it, making $V_X = V_{dd} - V_{th}$. This transistor can not be treated as a short circuit and voltage drop across it must be accounted in I_{sub} estimation for this input vector. Hence, we need to model a four transistor NMOS stack with the gate of top most transistor connected to V_{dd} which is modeled as n4s/1 stack.

Input vectors (1100 / 1010 / 0110): Here also, the I_{sub} is calculated from NMOS 'OFF' network. In the 'OFF' network two transistors with the gates connected to V_{dd} can be treated as short circuit which reduces the NMOS stack to 2 transistors and can be modeled as n2s/0 stack to predict the NAND4 leakage for these input vector combinations. Our simulation results show that the error is under tolerable limits, which does not effect much average error in μ and σ if we do not scale the I_{sub} of n2s/0 stack for accounting for the width difference of the 2 transistor NMOS stack and the NAND4 gate. Here again note that the same n2s/0 model will be used to predict the leakage for the NAND3 gate with input '100' and NAND2 gate with input '00'.

Input vectors (1001 / 0101 / 0011): As in the '0001' case, the top most transistor can not be removed from the stack to find equivalent stack, while other 'ON' transistor can be treated as a short circuit and can be removed from the stack.We need to develop n3s/1 model. This model is also used for I_{sub} modeling of NAND3 gate with '001' input.

Input vectors (0111 / 1011 / 1101): For I_{sub} calculation of these input vector combinations n2s/1 model can be used but the DIBL effect due to change in drain to source voltage of 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 n2s/1 model. In [2], n2s/1 model is used and scaling factors are also calculated for accounting the width difference in lower and higher order stack. The error in predicting the μ and σ using this model is very high. Thus, We need a separate model for each of these three input vectors.

Input vectors (1110): In this case, the bottom three transistors MN1, MN2 and MN3 can not be considered as short circuits due to the voltage drop across these transistors and hence, the I_{sub} will be predicted by a separate model i.e. n4s/14 with less error in our case compared to I_{sub} calculation using model n1s/0 with scaling factor in [2].

Input vectors (1111): In this case, MN1-MN4 are 'ON'. I_{sub} is determined by 'OFF' network i.e. parallel PMOS transistors for which we only need one model i.e. p1s/1 model, which can be reused for all 4 PMOS transistors.

To analyze the inaccuracy of models in [2], we calculate I_{sub} of NAND4, NAND3 and NAND2 gate for the inputs '1110', '110', and '10' by simulating 20000 Monte Carlo samples taken considering 10% variation in process parameters with $V_{dd} = 1V$, $T = 27°C$ and $W = 28nm$. In our framework, we represent these models as n4s/14, n3s/6 and n2s/2. I_{sub} of these models is compared with leakage of single NMOS transistor with '0' input i.e. n1s/0 model. Figure 3.2 shows the correlation curve between the SPICE leakage i.e. n4s/14, n3s/6 and n2s/2 and modeled leakage i.e. n1s/0. The error of modeling n4s/14, n3s/6, n2s/2 by n1s/0 can vary from minimum 0.01% to maximum 59.97%, 51.22% and 34.78% respectively. This error is due to the drop across the 'ON' transistors, which affects I_{sub} in two ways: 1) n1s/0 model over-estimate the drain-to-source voltage of the 'OFF' transistor, 2) Drain Induced Barrier Lowering (DIBL) effect also changes the V_{th} of the 'OFF' transistor. One of the important observations can be made here is that the error is higher for higher leakage samples. In our methodology, we remove the 'ON' transistors from the 'OFF' network if and only if I_{sub} of the stack falls into low leakage region. Our Proposed methodology gives less than $<0.5\%$ error in μ and σ of the I_{sub} of basic gates i.e. NAND4, NAND3, NAND2, NOR4, NOR3, NOR2 obtained using our extracted models among basic gates of the considered 20 CMOS gates standard cell library.

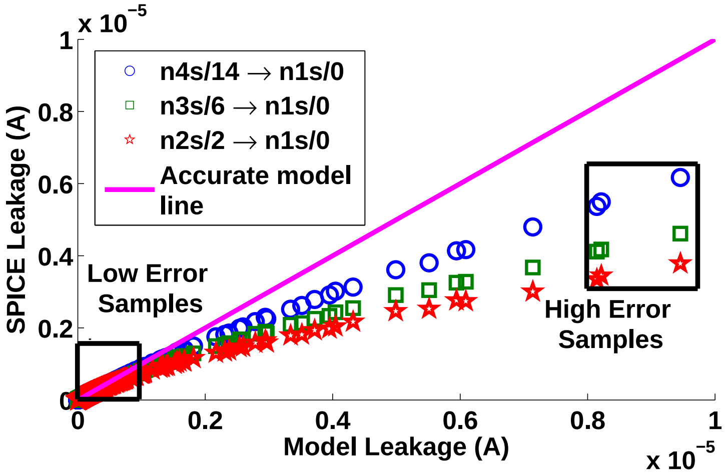


Figure 3.2: Correlation between SPICE and model subthreshold leakage using methodology in [2]

3.1.1.1 Gate Tunneling Leakage Modeling

Authors in [32, 33, 101] model the direct I_{gate} flowing from bulk and source/drain overlap region to gate of MOSFET. The direct I_{gate} can be represented as follows.

$$I_{gate} = W.L.A_g \left(\frac{V_{ox}}{T_{ox}}\right)^2 exp\left(\frac{-B_g\left(1-\left(1-\frac{V_{ox}}{\phi_{ox}}\right)^{3/2}\right)}{\frac{V_{ox}}{T_{ox}}}\right) \tag{3.2}$$

Here, $A_g = \frac{q^3}{16\pi^2\hbar\phi_{ox}}$, $B_g = \frac{4\sqrt{2m^*}\phi_{ox}^{3/2}}{3\hbar q}$, J_{DT} is the direct tunneling current density, V_{ox} is the potential drop across the gate oxide, ϕ_{ox} is the barrier height of the tunneling electron, m^* is the effective mass of an electron in the conduction band of silicon and T_{ox} is the oxide thickness. Three important components of I_{gate} in a scaled MOSFET device are: 1) gate to source/drain overlap region current ($I_{gd/gs}$) due to potential difference ($V_{gd/gs}$) across overlapped region; 2) Gate to channel current (I_{gc}), part of which goes to source (I_{gcs}) and rest goes to drain (I_{gcd}); and 3) Gate to substrate leakage current (I_{gb}). Out of the three components, contribution of $I_{gd/gs}$ is highest to the total I_{gate} while I_{gb} has the lowest contribution. I_{gb} is found to be several orders lower than other components, thus neglected in I_{gate} estimation of CMOS gates.

To calculate gate tunneling leakage, we generate the gate tunneling leakage values from SPICE tool using the parameters *IGCMOD* and *IGBMOD* in the device model. Since, the contribution of I_{gb} to the total I_{gate} is very less thus I_{gb} is neglected by simply turning 'OFF' *IGBMOD* model parameter in the device model. Gate-to-source and gate-to-drain tunneling current is modeled for each basic model shown in Figure 3.3. These currents are modeled as voltage dependent current sources depending on the potential difference across the the terminals.

Figure 3.3 shows the basic I_{gate} conditions which can exist in CMOS gates. However, depending on the location of the transistor on the stack and process parameters, their source and drain voltages can be changed. To accurately capture I_{gate} of a CMOS gate, a highly non-linear model is required to scale I_{gate} of a transistor shown in Figure 3.3. Our method is based on the identification of dominant I_{gate} transistors and breaking a higher order stack into smaller stacks across all gates with their input vectors and using a common set of I_{gate} models. I_{gate} of each transistor is calculated based on the current flowing across gate terminal from source and drain terminals in PDN and PUN network. Both 'ON' and 'OFF' networks contribute to the I_{gate}.

Authors in [102] proposed dual dielectric - dual thickness (DKDT) approach to reduce the I_{gate} in combinational circuits. This work relies on I_{gate} characterization of CMOS gates for different dielectric materials having different T_{ox}, which requires very large characterization time. Linear change in I_{gate} is also observed for different values of thickness and dielectric constant (K). This linear change can be easily incorporated in our framework of reduced set of models trough surrogate modeling by simply including K and T_{ox} during model characterization, without requiring much extra training samples.

Now, we explain our I_{gate} models extraction methodology using NAND4 gate shown in Figure 3.1. LSB of a input vector is always applied to the transistor nearest with output node. Same conventions are used here to represent models as for I_{sub} models.

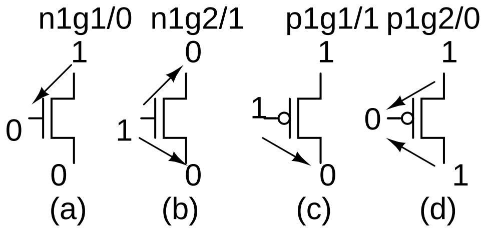


Figure 3.3: Basic I_{gate} models for single transistor

Both PDN and PUN network contribute to I_{gate} of a gate. Thus, voltage conditions at different terminals of all transistors need to be analyzed. In all input vectors except '1111' of NAND4 gate, only 'ON' PMOS transistors will contribute to the I_{gate} because 'OFF' transistors i.e. input $=$ logic '1' will have the same voltage i.e. logic '1' at all terminals. The 'ON' transistors have same terminal conditions as in Figure 3.3.(d) and thus, do not require any new model.

Input vector (0000): In PDN, Only MN4 transistor has the significant leakage

Table 3.1: Models for estimation of I_{sub} and I_{gate} of AND type basic CMOS gates with respect to each input vector

NAND4				**NAND3**			
Input (DCBA)	I^*_{sub}	I_{gate}		*Input (DCBA)*	I^*_{sub}	I_{gate}	
		PDN	*PUN*			*PDN*	*PUN*
0000	n4s/0	n1g1/0	p1g2/0	000	n3s/0	n1g1/0	p1g2/0
0001	n4s/1	n4g/1	p1g2/0	001	n3s/1	n3g/1	p1g2/0
0010	n3s/0	n4g/2	p1g2/0	010	n2s/0	n3g/2	p1g2/0
0011	n3s/1	n4g/3	p1g2/0	011	n3s/3	n3g/3	p1g2/0
0100	n3s/0	n4g/4	p1g2/0	100	n2s/0	n1g1/0, n1g2/1	p1g2/0
0101	n3s/1	n4g/5	p1g2/0	101	n3s/5	n2g/1, n1g2/1	p1g2/0
0110	n2s/0	n4g/6	p1g2/0	110	n3s/4	n1g1/0, n1g2/1	p1g2/0
0111	n4s/7	n4g/7	p1g2/0	111	p1s/1	n1g2/1	p1g1/1
1000	n3s/0	n1g1/0, n1g2/1	p1g2/0	**NAND2**			
1001	n3s/1	n3g/1, n1g2/1	p1g2/0	00	n2s/0	n1g1/0	p1g2/0
1010	n2s/0	n3g/2, n1g2/1	p1g2/0	01	n2s/1	n2g/1	p1g2/0
1011	n4s/11	n3g/3, n1g2/1	p1g2/0	10	n2s/2	n1g1/0, n1g2/1	p1g2/0
1100	ns2/0	n1g1/0, n1g2/1	p1g2/0	11	p1s/1	n1g2/1	p1g1/1
1101	ns4/13	n2g/1, n1g2/1	p1g2/0	**INV**			
1110	n4/14	n1g1/0, n1g2/1	p1g2/0	0	n1/0	n1g1/0	p1g2/0
1111	p1/1	n1g2/1	p1g1/1	1	p1/1	n1g2/1	p1g1/1

*We do not use separate columns for PDN and PUN under I_{sub} column because either PUN or PDN is used for I_{sub} estimation

flowing from V_{dd} (output node) to input which is same as I_{gate} of the transistor shown in Figure 3.3.(a). Drain voltage of other transistors is very less, gives almost negligible current across drain and gate terminal. Thus, we do not need any I_{gate} model other than basic models for this input vector and n1g1/0 model is enough to calculate I_{gate} for NMOS stack applied with this input vector. Here, we conclude that the in any NMOS stack with all 'OFF' transistors, I_{gate} will be due to the only transistor, which is connected to the output node. Same is true for PMOS stack with all 'OFF' transistors.

Input vectors (0001/0011/0111): In 'OFF' PDN for '0001' input, MN3 transistor has the significant leakage whose drain and source voltages are significantly different than in Figure 3.3.(a). We denote this new leakage model as n4g/1. Similarly, new models n4g/3 and n4g/7 are required for '0011' and '0111' inputs respectively. This is due to the different threshold voltage drop across 'ON' transistors in presence of process variations. Previous work in [6] and [7] neglects the I_{gate} of 'ON' transistors in these input vectors. We observe that this assumption results in high error for I_{gate} estimation.

Input vectors (0010/0100): In this case, same leakage is observed across MN4 transistor as in the case of '0000' input. New leakage models are required to calculate gate leakage of MN3 and MN2 transistor for '0010' and '0100' inputs and models are represented as n4g/2 and n4g/4 respectively. Here also the methodology in [6] and [7] neglect the I_{gate} of 'ON' transistor resulting in high error.

Input vector (1000/1100/1110): For '1000' input, transistor MN1 is 'ON' which makes it's drain and source terminal connected to ground. It's I_{gate} will be same as transistor in Figure 3.3.(b). I_{gate} of top three transistors in PDN is same as NAND3 gate with '000' input. In NAND3 gate with '000' only top transistor will have significant I_{gate} as in the case of NAND4 gate with '0000' input. Same conditions are also applicable to '1100' and '1110' inputs. We can remove 'ON' transistors starting from the bottom until an 'OFF' transistor comes. Drop across removed 'ON' transistors in 'OFF' PDN do not affect gate leakage very much. Thus, No extra model is required in all three cases.

Input vectors (1001/1010/1011): As in previous case, the bottom transistor has significant I_{gate} for all three input vectors. But the top three transistors will have different I_{gate} and require different models named as n3g/1, n3g/2, n3g/3 respectively. n3g/1, n3g/2, n3g/3 models are also used for I_{gate} estimation of NMOS stack of NAND3 gate with '001', '010' and '011' inputs respectively.

Input vector (1101): Bottom two transistors have same conditions as in the case of '1100' input but requires one extra model due to top two transistors as n2g/1. n2g/1 model is also used for I_{gate} estimation of NMOS stack of NAND2 gate with '01' input.

Input vector (0101/0110): In each case, two transistors contribute to the I_{gate} and impose significant change in drain and source voltage in comparison to basic gate leakage model conditions. We jointly model I_{gate} of two transistors into single model for each input vector. Thus, two new models are required and named as n4g/5, n4g/6 for input '0101' and '0110' respectively.

Input vectors (1111): In this case, all 4 PMOS transistors are 'OFF' and contribute to I_{gate} using model of transistor as in Figure 3.3.(c). The 'ON' PDN's I_{gate} can be estimated using model n1g2/1 because inputs are applied with logic '1' and source, drain terminals are at logic '0'.

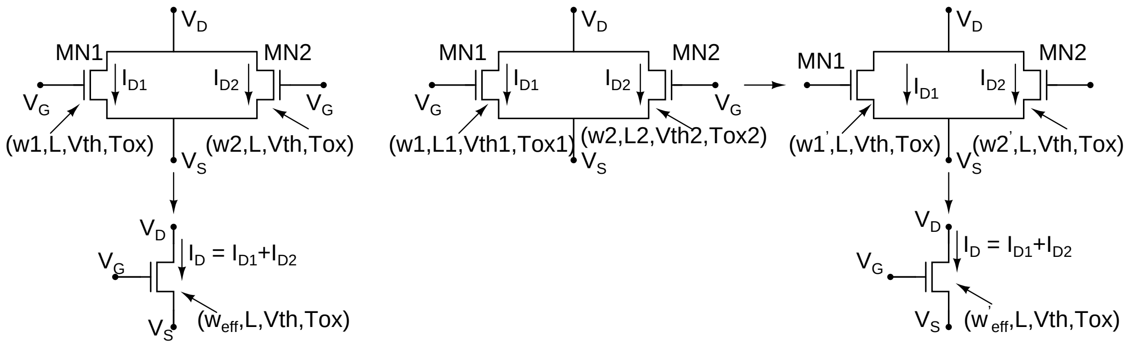
Similarly, I_{gate} for other CMOS gates can be modeled. Table 3.1 shows the list of models used in I_{sub} and I_{gate} modeling of AND type basic CMOS gates.

3.1.2 Complex Logic Gates consisting parallel or Stack of Parallel transistors

For these type of complex logic gates, we replace the parallel transistors of different widths with a single transistor of equivalent width. Since our models are function of width also, there is no need to characterize any new model. Next, we develop the analytical equations to calculate effective width of the transistor with and without process variation.

3.1.2.1 Effective width calculation of parallel transistors of same input without considering process variations

The possible node voltages and process parameter conditions in the absence of process variations is shown in Figure 3.4.(a). The I_{sub} of a transistor can be represented as follows.



(a) w/o Process Variation

(b) with Process Variations

Figure 3.4: Effective width estimation of parallel transistors with same inputs

$$I_{sub} = \mu_N C_{ox} \frac{W}{L} V_t^2 exp\left[\frac{V_{gs} - V_{th}}{nV_t}\right]\left[1 - exp\left[-\frac{V_{ds}}{V_t}\right]\right] \tag{3.3}$$

From (3.3), it can be observed that I_{sub} is directly proportional to the width of transistor. But in actual, this width is effective width which is calculated according to BSIM4 device equations. The effective width can be represented in terms of drawn width W_{drawn} as follows.

$$W_{eff} = \frac{W_{drawn}}{NF} + XW - 2dW \tag{3.4}$$

$$dW = dW' + DWG.V_{gsteff} + DWB\left(\sqrt{\phi_s - V_{bseff}} - \sqrt{\phi_s}\right) \tag{3.5}$$

$$dW' = WINT + \frac{WL}{L^{WLN}} + \frac{WW}{W^{WWN}} + \frac{WWL}{L^{WLN}W^{WWN}} \tag{3.6}$$

Here,, NF =Number of device fingers, XW =Parameter to account channel width offset due to mask/etch effect, DWG, DWB =To account for the contribution of both gate and substrate effect. $WINT$, WL, WW, WWL, WLN, WWN = Model parameters to describe the dependence of dW on device geometry. $WINT$ is calculated in the traditional manner from which 'delta W' is extracted. (from the intercept of straight lines on a $1/R_{ds}$∼W_{drawn} plot)

In general,

$$WL = WW = WWL = 0,\ WLN = WWN = 1 \tag{3.7}$$

putting values from Equation (3.7) in Equation (3.6), we get

$$dW' = WINT \tag{3.8}$$

$$dW = WINT + DWG.V_{gsteff} + DWB\left(\sqrt{\phi_s - V_{bseff}} - \sqrt{\phi_s}\right) \tag{3.9}$$

Since, in our model we are not considering the gate and substrate effects, the parameters related to gate and substrate effects can be assumed to be zero, i.e.

$$DWG = DWB = 0 \tag{3.10}$$

putting values from Equation (3.10) to Equation (3.9),

$$dW = WINT \tag{3.11}$$

putting Equation (3.11) into Equation (3.4), we get,

$$W_{eff} = \frac{W_{drawn}}{NF} + XW - 2WINT \tag{3.12}$$

In our case, the number of fingers, NF =1 and XW =0, final value of W_{eff} can be represented as:

$$W_{eff} = W_{drawn} - 2WINT \tag{3.13}$$

Now consider the two transistors MN1 and MN2 as shown in Figure 3.4 having width $W_1$and W_2 with same potential at all terminals and being same other parameters i.e. L, V_{th}, T_{ox} are also equal. These transistors can be replaced by a single transistor with equivalent width as $w_{equi.,drawn}$,which can be calculated as follows.

$$W_{equi.,eff} = W_{1,eff} + W_{2,eff} \tag{3.14}$$

$$\begin{aligned} W_{equi.,drawn} - 2.WINT = \; & W_{1,drawn} - 2.WINT \\ & +W_{2,drawn} - 2.WINT \end{aligned} \tag{3.15}$$

$$W_{equi.,drawn} = W_{1,drawn} + W_{2,drawn} - 2.WINT \tag{3.16}$$

which can be generalized for N parallel transistors as follows:

$$\begin{gathered} W_{equi.,drawn} = W_{1,drawn} + W_{2,drawn} + ... - 2.N.WINT \\ +2.WINT \qquad N = 2, 3, ...n \end{gathered} \tag{3.17}$$

$$\begin{gathered} = W_{1,drawn} + W_{2,drawn} + ... - 2.(N-1).WINT \\ N = 2, 3, ...n \end{gathered} \tag{3.18}$$

Equation (3.18) represents the final equivalent drawn width that is to be used in simulations for the case of N parallel transistors. The factor $2.(N - 1).WINT$ must be subtracted from the sum of drawn widths. In our simulations $WINT = 5nm$. For example, if there are two transistors having widths $W_1 = W_2 = 50nm$, then these transistors can be replaced by a single transistor of width $90nm$ ($50nm + 50nm - 2 * 5nm$). I_{gate} is also directly proportional to the effective width of transistor. Same effective width formula is also applicable for I_{gate} calculation of parallel transistors applied with same inputs.

3.1.2.2 Effective width calculation of parallel transistors of same input in presence of process variations

In presence of process variations, we can not use effective width formula directly because both parallel transistors can have different process parameters as shown in

Figure 3.4.(b). Instead, we first compute the change in width required to change the current of transistor by same amount as process parameter changes. Then, effective width is calculated using the modified widths of the parallel transistors. This change in width is calculated for both I_{sub} and I_{gate}. Let us assume the tuples (L_{mean}, $V_{th,mean}$, $T_{ox,mean}$), ($L_{mean}+\triangle L$, $V_{th,mean}+\triangle V_{th}$, $T_{ox,mean}+\triangle T_{ox}$) represent the nominal values and a sample of process parameters of transistor respectively. W_{act} and ($W_{act}+\triangle W$) denote the actual width and modified width (W_{mod}). The change in the width ($\triangle W$) can be calculated by equating the current flowing due to the parameters (W_{act}, $L_{mean}+\triangle L$, $V_{th,mean}+\triangle V_{th}$, $T_{ox,mean}+\triangle T_{ox}$) and ($W_{act}+\triangle W$, L_{mean}, $V_{th,mean}$, $T_{ox,mean}$). We represent these currents as $I_{sub/gate,act}$ and $I_{sub/gate,mod}$. For I_{sub}, $\triangle W$ can be calculated as:

$$I_{sub,act} = I_{sub,mod} \tag{3.19}$$

$$\begin{aligned} A.\tfrac{W_{act}}{(L_{mean}+\triangle L)}.exp\left[\tfrac{V_{gs}-(V_{th,mean}+\triangle V_{th})}{\eta V_t}\right]\left[1-exp\left[-\tfrac{V_{ds}}{V_t}\right]\right] = \\ A.\tfrac{(W_{act}+\triangle W)}{L_{mean}}.exp\left[\tfrac{V_{gs}-V_{th,mean}}{\eta V_t}\right]\left[1-exp\left[-\tfrac{V_{ds}}{V_t}\right]\right] \end{aligned} \tag{3.20}$$

$$\text{Here } A = \mu_N C_{ox} V_t^2$$

$$\begin{aligned} A.\tfrac{W_{act}}{(L_{mean}+\triangle L)}.exp\left[\tfrac{V_{gs}-V_{th,mean}}{\eta V_t}\right].exp\left[\tfrac{-\triangle V_{th}}{\eta V_t}\right]\left[1-exp\left[-\tfrac{V_{ds}}{V_t}\right]\right] = \\ A.\tfrac{(W_{act}+\triangle W)}{L_{mean}}V_t^2.exp\left[\tfrac{V_{gs}-V_{th,mean}}{\eta V_t}\right]\left[1-exp\left[-\tfrac{V_{ds}}{V_t}\right]\right] \end{aligned} \tag{3.21}$$

$$\frac{(W_{act}+\triangle W)}{L_{mean}} = \frac{W_{act}}{(L_{mean}+\triangle L)}.exp\left[\frac{-\triangle V_{th}}{\eta V_t}\right] \tag{3.22}$$

$$1+\frac{\triangle W}{W_{act}} = \frac{L_{mean}}{(L_{mean}+\triangle L)}.exp\left[\frac{-\triangle V_{th}}{\eta n V_t}\right] \tag{3.23}$$

$$\triangle W = W_{act}\left[\frac{L_{mean}}{(L_{mean}+\triangle L)}.exp\left[\frac{-\triangle V_{th}}{\eta V_t}\right]-1\right] \tag{3.24}$$

Here, η is subthreshold swing factor and η=1 is used in Equation (3.24), V_t is the thermal voltage and is given by kT/q whose value at room temperature is $26mV$. Equation (3.24) denotes the change in width required due to change in V_{th} and length of the transistor. $\triangle V_{th}$ is the change in the width due to multiple effects such as Channel Dopants (N_{ch}), L and T_{ox}. Across three effects, V_{th}change ($\triangle V_{th,L/T_{ox}}$) due to L and T_{ox} also depends upon the drain-to-source voltage (V_{ds}) of the transistor as shown in Figure 3.5.

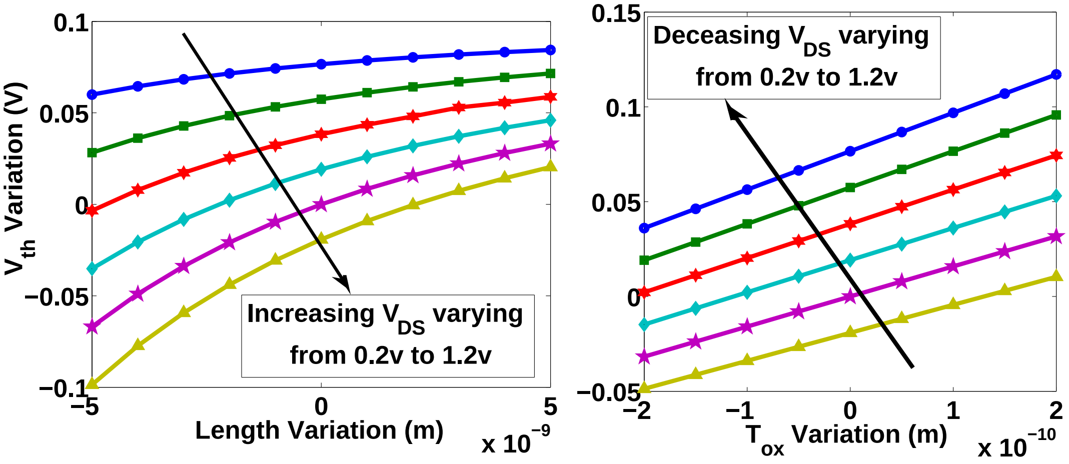


Figure 3.5: Effect of variation in length and oxide thickness on threshold voltage (different drain-to-source voltage)

The variation in process parameters changes V_{ds} of the corresponding transistor and can not be predicted. Thus, $\triangle V_{th,L/T_{ox}}$ can not be accurately calculated and gives error in $\triangle W$ estimation. This problem can be somewhat solved by assuming the mean value of V_{ds} for all samples of process parameters. However, this problem will occur for a very less number of input vectors and does not affect the average error across all input vectors of a CMOS gate. Function $V_{th,L/T_{ox}}$ is captured by fitting the data samples by Latin Hypercube Sampling method and curve fitting toolbox in MATLAB. The fitting function is given as follows.

$$V_{th,T_{ox}} = f(T_{ox}, V_{DS}) = A_1 + B_1.V_{DS} + C_1.T_{ox} \tag{3.25}$$

$$V_{th,L} = f(L, V_{DS}) = A_2 + B_2.V_{DS} + C_2.L + D_2.V_{DS}^2 + E_2.V_{DS}.L \qquad (3.26)$$

Both fitted models in Equation (3.25) and Equation (3.26) give sufficient accuracy and greater than 0.999 correlation coefficient is obtained with respect to actual curve. Figure 3.6 shows the regression models to represent the effects of L, V_{th} and V_{DS} on V_{th} of the transistor. The change in threshold voltage, $\triangle V_{th,L/T_{ox}}$, can be calculated by subtracting V_{th} at nominal process parameters from the modified V_{th}.

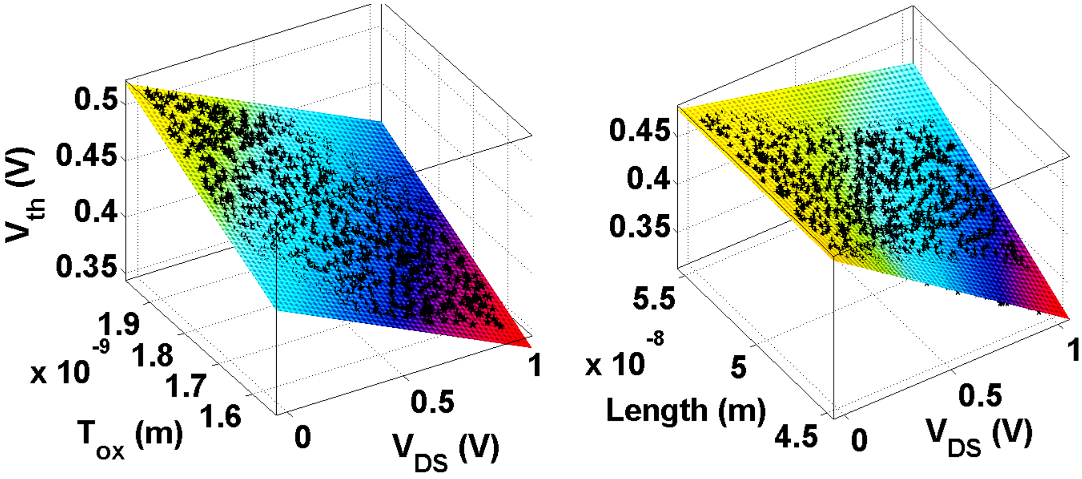


Figure 3.6: Regression models obtained through curve fitting showing the effect of T_{ox}, L and V_{ds} on V_{th}.

The I_{gate} model in (3.2) is very complex and does not suit for our framework for effective width estimation. First, we develop an accurate and simple regression based analytical model of I_{gate} for all major components in terms of transistor terminal voltages and process parameters as shown in Equation (3.27).

$$I_{gate} = \alpha.W.L.V_{gs/ds}.exp\left[\beta.V_{gd/gs} + \gamma.T_{ox}\right] \qquad (3.27)$$

Here, α, β and γ are fitting coefficients and are obtained through curve fitting, $V_{gd/gs}$ is the voltage across oxide depending upon gate to source or gate to drain voltage. V_{th} is not included in the model because it does not affect the I_{gate}. T and V_{ds} also do not affect the I_{gate} and hence, not included in the model. Equation (3.2) also indicate that I_{gate} is the function of W, L, V_{ox} and T_{ox}. Our model is more accurate than the model developed in [6] for complete process variation space and V_{gs} ranging from $0\,V$ to V_{dd} for fixed V_{ds}. Figure 3.7 shows the regression model obtained through curve fitting in MATLAB for fixed width, length and correlation of model simulated data with the SPICE data.

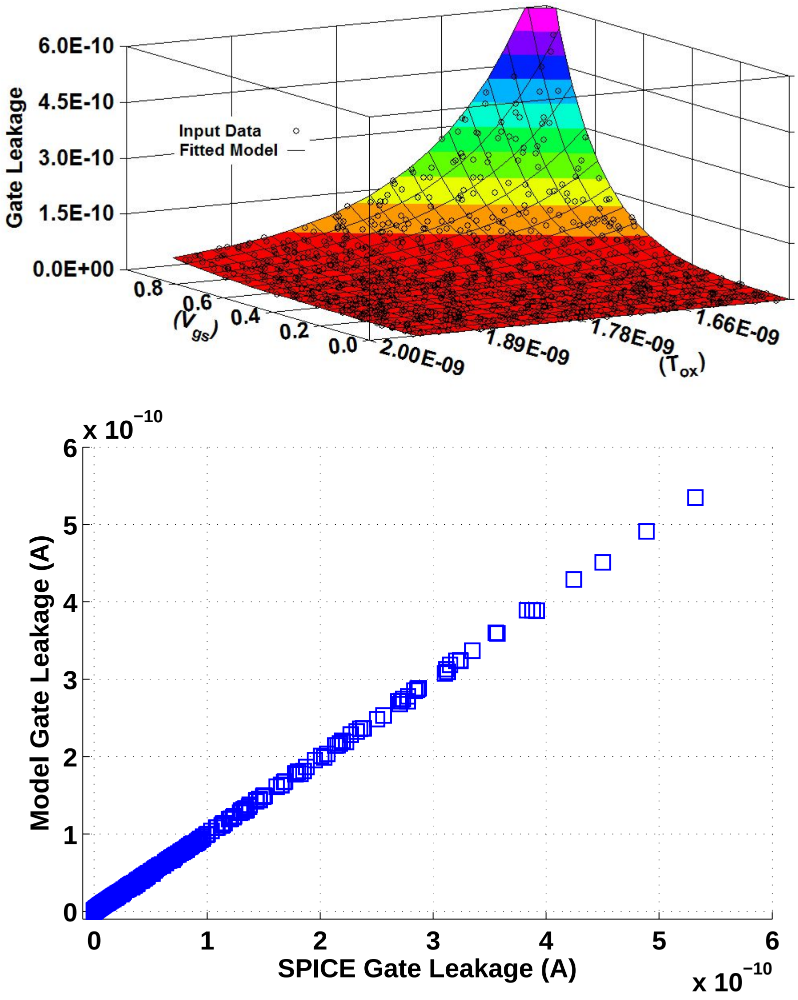


Figure 3.7: (Top) Regression model showing the proposed I_{gate} model, (Bottom) Correlation curve between I_{gate} obtained from SPICE and fitting model

Authors in [6] model the I_{gate} as shown in Equation (3.28), which is only accurate for higher values of $V_{gs/ds}$.

$$I_{gate} = \alpha.W.L.exp\left[\beta.V_{gd/gs} + \gamma.T_{ox}\right] \tag{3.28}$$

The error in previous model for low $V_{gs/ds}$ can be easily observed. For example, if $V_{gd/gs}$ is zero, then leakage should be zero due to zero potential difference across oxide but Equation (3.28) will give some leakage. I_{gate} for lower $V_{gd/gs}$ values is very important for the case of transistors with higher width values and may give comparable leakage as the case of minimum width transistors with higher $V_{gd/gs}$ values.

Authors in [103] proposed a methodology to optimize I_{gate} in datapath circuits. A bottom-up approach is developed for the Characterization of functional units such as adder, multiplier, shifter, register etc. First I_{gate} of the NAND gate is characterized using analog simulation and then this characterized data is used to develop analytical model for each functional unit as shown in Equation (3.29).

$$I_{gate} = A.exp(-T_{ox}/\alpha) + B \tag{3.29}$$

This analytical model includes only T_{ox} as a variable parameter and did not consider the dependency of I_{gate} on L and W of transistors. This models also suffers from the large error in I_{gate} estimation at lower $V_{gd/gs}$ values. This work only considered the NAND as a component in each functional unit, thus did not consider realistic characterization scenario. The proposed methodology relies on characterization of a gate for each input vector, thus requires very large time to characterize a complete library.

Now, expression to calculate the change in the width ($\triangle W$) for I_{gate} can be derived as similar to process adopted for I_{sub}.

$$I_{gate,act} = I_{gate,mod} \tag{3.30}$$

$$\begin{aligned}\alpha.W_{act}.(L_{mean}+\triangle L).exp\left[\beta.V_{gd/gs}+\gamma.(T_{ox,mean}+\triangle T_{ox})\right] = \\ \alpha.(W_{act}+\triangle W).L_{mean}.exp\left[\beta.V_{gd/gs}+\gamma.T_{ox,mean}\right]\end{aligned} \tag{3.31}$$

$$\begin{aligned}\alpha.W_{act}.(L_{mean}+\triangle L).exp\left[\beta.V_{gd/gs}+\gamma.T_{ox,mean}\right].exp\left[\gamma.\triangle T_{ox}\right] = \\ \alpha.(W_{act}+\triangle W).L_{mean}.exp\left[\beta.V_{gd/gs}+\gamma.T_{ox,mean}\right]\end{aligned} \tag{3.32}$$

$$\frac{.(L_{mean}+\triangle L)}{L_{mean}}.exp\left[\gamma.\triangle T_{ox}\right] = 1+\frac{\triangle W}{W_{act}} \tag{3.33}$$

$$\triangle W = W_{act}\left[\frac{.(L_{mean}+\triangle L)}{L_{mean}}.exp\left[\gamma.\triangle T_{ox}\right]-1\right] \tag{3.34}$$

Table 3.2: I_{sub} and I_{gate} Models for 2-input XOR gate with respect to all input vectors

Input	I^{*}_{sub}		
(BA)	*G1*	*G2*	*G3*
00	n1s/0	p2s/2	n1s/0
01	p1s/1	n2s/2, n2s/1	n1s/0
10	n1s/0	n2s/2, n2s/1	p1s/1
11	p1s/1	p2s/1	p1s/1

Input	I_{gate}					
(BA)	*G1*		*G2*		*G3*	
	PDN	*PUN*	*PDN*	*PUN*	*PDN*	*PUN*
00	n1g1/0	p1g2/0	n1g1/0, n1g2/1	p2g/2	n1g1/0	p1g2/0
01	n1g2/1	p1g1/1	n2g/1, n1g1/0, n1g2/1	p1g2/0	n1g1/0	p1g2/0
10	n1g1/0	p1g2/0	n2g/1, n1g1/0, n1g2/1	p1g2/0	n1g2/1	p1g1/1
11	n1g2/1	p1g1/1	n1g1/0, n1g2/1	p1g2/0, p1g1/1	n1g2/1	p1g1/1

*We do not use separate columns for PDN and PUN under I_{sub} column because either PUN or PDN is used for I_{sub} estimation

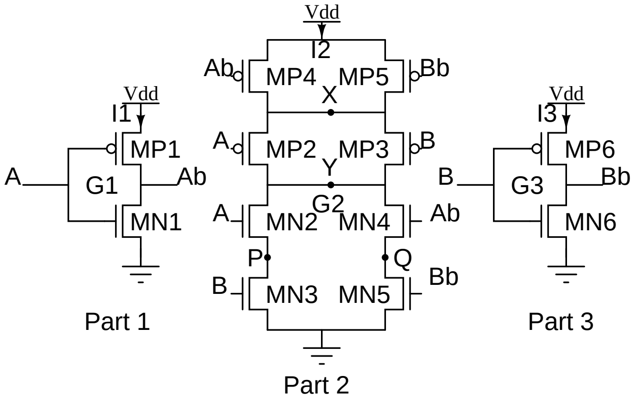


Figure 3.8: 2-input XOR gate

3.1.2.3 Subthreshold Leakage Modeling of Complex logic gates consisting parallel or stack of parallel transistors.

Consider a 2-input XOR gate as shown in Figure 3.8. Let us examine the input vectors to check whether the new stack models are required or not and how effective width can be used in model reduction for stacks consisting of parallel transistors. It should be noted that we do not use any conventions to label parallel transistor stack because each parallel transistor stack is converted to its equivalent stack for each input vector.

Input Vectors (00/11) : When the input vectors ‘00’/‘11’ are applied, NMOS network of part 2 is ‘ON’ due to either {MN2, MN3} or {MN4, MN5} are ‘ON’. For input ‘00’, {MP4, MP5} are ‘OFF’ and {MP2, MP3} are ‘ON’. Based on the explanation given in Section 3.1.2.1 and 3.1.2.2, parallel transistors with same inputs are merged into single transistor and equivalent stack is formed. Now, I_2 can be estimated using basic stack model i.e. p2s/2. Since in presence of process variations, we do not know the values of intermediate node voltage ‘X’ for different values of process parameters. Thus, accurate V_{th} change can not be evaluated. However, the effect of unknown values at ‘X’ can be suppressed by assuming the mean node voltage calculated at mean values of process parameters. I_1 and I_3 both can be estimated using n1s/0 model due to turning 'ON' of PDN in both inverters. In case of input '11', {MP4, MP5} are 'ON' and {MP2, MP3} are 'OFF'. I_2 is estimated using stack model p2s/1. Node voltage at 'X' can be assumed to be V_{dd} because 'ON' MP4 and MP5 transistors connect the 'X' at approximately equal to V_{dd}. I_1, I_3 can be estimated using p1s/1 model. So, no extra I_{sub} model needed to be characterized for these input vectors in comparison of the methodology in [2].

Input Vectors (01/10) : When 'BA = 01', PUN will be 'ON' because MP2 and MP5 transistors are in 'ON' condition. Output node 'Y' is connected to the V_{dd}. I_{sub} of PDN (I_1) can be calculated by summing leakage of two separate stacks: 1) n2s/2 model formed by MN2 and MN3 transistors, 2) n2s/1 model formed by MN4 and MN5 transistors. I_1 and I_3 can be estimated using p1s/1 (MP1 transistor is 'OFF') and n1s/0 (MN6 transistor is 'OFF') models respectively. I_{sub} for input 'BA = 10' will be same as '01' input case due to the symmetry in terms of stacks used for I_{sub} estimation.

3.1.2.4 Gate tunneling Leakage Modeling of Complex logic gates containing parallel or stack of parallel transistors.

Consider an example of 2-input XOR gate as shown in Figure 3.8.(b) to analyze I_{gate} estimation of CMOS gates having parallel transistors for each input vector. I_{gate} flows across both PDN and PUN.

Input Vector (00) : For this input, PDN of gate G2 consist of two stacks where each stack is similar as the PDN of 2-input NAND with inputs '00' and '11'. Corresponding I_{gate} models are provided in Table 3.1. Inputs 'BA = 00' and 'BbAb = 11' form the condition of parallel transistor with same inputs in PUN of gate G2. To calculate I_{gate} of PUN of G2, first Effective width needs to be calculated using Equation (3.34) then basic model p2g/2 can be used for equivalent stack formed after combining parallel transistors. This p2g/2 model is characterized for 2-input NOR gate as similar to n2g/1 model for 2-input NAND gate. I_{gate} of PDN for gate G1/G2 is calculated using n1g1/0 model due to MN1/MN6 transistor while p1g2/0 model is used for PUN due to MP1/MP6 transistor.

Input Vector (11) : I_{gate} for PDN of gate G2 will be same as for input '00' because of the same stacks. Inputs 'BbAb = 00' turn on the PMOS transistors MP4 and MP5 in PUN of G2, which connects the node 'X' to V_{dd}. Terminal voltages of all four transistors in PUN (MP2, MP3, MP4, MP5) are perfectly defined and will be same for all process parameters i.e. $V_X = \mathit{1}$, $V_Y = 0$. Parallel transistors share same model due to same terminal voltage conditions. I_{gate} of MP4(MP5) and MP2(MP3) transistors can be estimated using p1g2/0 and p1g1/1 model respectively. Models n1g2/1 and p1g1/1 are used for both gates G1 and G2.

Input Vector (01) : For 'BA = 01' input, I_{gate} of the stack in PDN of gate G2 formed by MN2 and MN3 transistors is similar to PDN of 2-input NAND gate with '01' input and can be estimated through n2g/1 model. Two models i.e. n1g1/0 and n1g2/1 for the stack formed by the transistors MN4 and MN5. In PUN, MP3 and MP4 transistors are 'ON' due to 'Ab = B = 0' input, which connects the intermediate node 'X' and output node 'Y' to V_{dd}. Since, transistors MP2 and MP5 have logic high potential at all terminals, thus do not have any I_{gate} across these transistor. MP2 and MP5 transistor can be removed from the circuit, independent of their process parameters. Remaining MP3 and MP4 transistors have same terminal voltages i.e. '0' input at gate, source and drain is connected to '1'. I_{gate} for both transistor can be evaluated from p1g2/0 model. Gate G1 and G2 conditions are same as inverter with '1' and '0' input respectively. I_{gate} of G1 and G2 is estimated using the models as used for inverter.

Input Vector (10) : Input '10' is symmetric with '01' in terms of the stack models used for I_{gate} calculation. Same models are used as for the case of '01' input.

Table 3.2 shows the I_{sub} and I_{gate} models for 2-input XOR gate with respect to all input vectors. Models are separated for PDN and PUN of each gate G1, G2, G3 for each input vector.

Following rules can be derived for the stack models extraction among different gates of a standard cell library for I_{sub} estimation.

1. If the parallel transistors are supplied with same inputs, replace it with single transistor having equivalent effective width; and rules 3 and 4 are applied.

2. If different inputs are supplied to the parallel transistors, then the 'OFF' transistor is removed because the 'ON' transistor makes the drain and source voltages equal, thus, nullifies the effect of 'OFF' transistor; and then rules 3 and 4 are applied.

3. If the number of 'ON' transistors are greater than the number of 'OFF' transistors, then (i) separate stack model is required per stack type per input vector; else (ii) the transistors applied with 1/0 input except transistor closest to the output for NMOS/PMOS stack are removed and then a separate stack with that input vector is built and modeled for this case.

4. I_{sub} due to parallel stacks simply adds up.

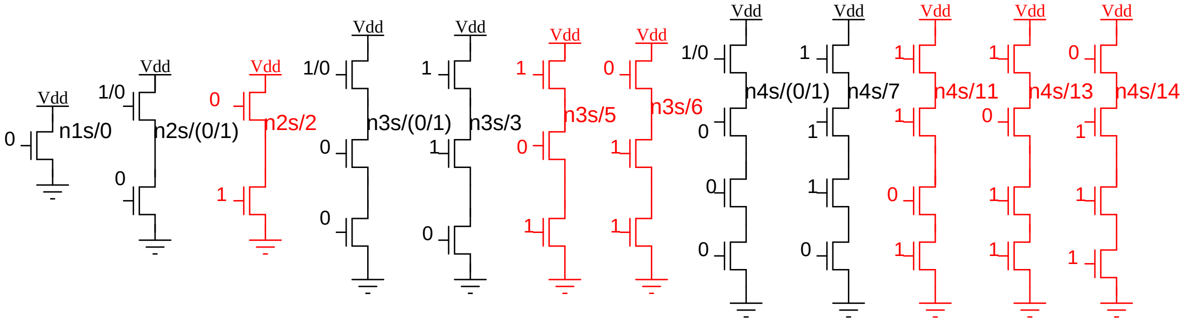


Figure 3.9: Basic NMOS stack models for I_{sub} estimation

We have added n2s/2 and p2s/1 stack models to the list of basic stacks, as the error was observed to be high while estimating the leakage of these stacks using lower order stacks according to the rules. Commonly used stacks are presented in Figure 3.9, stacks in red representing the extra stack models to be characterized than in [2]. In a similar way, the PMOS stacks can also be derived for the OR-type family of gates. However, the number of stack models to be characterized are solely dependent on the type of gates available in the standard cell library. New models are required to characterize for complex gates consisting series-parallel stacks are listed in Figure 3.10. Using our effective width methodology, basic stack models of Figure 3.9 can be used. It should be noted that these stack models can only be used for leakage estimation for the gates having series connection of two parallel transistor pairs. If we do not combine parallel transistors, then 16 new models are required for leakage power estimation of XOR2, Majority, AOI22, OAI22 gates. Total 30 models are required in our methodology compared to 34 models in [2].

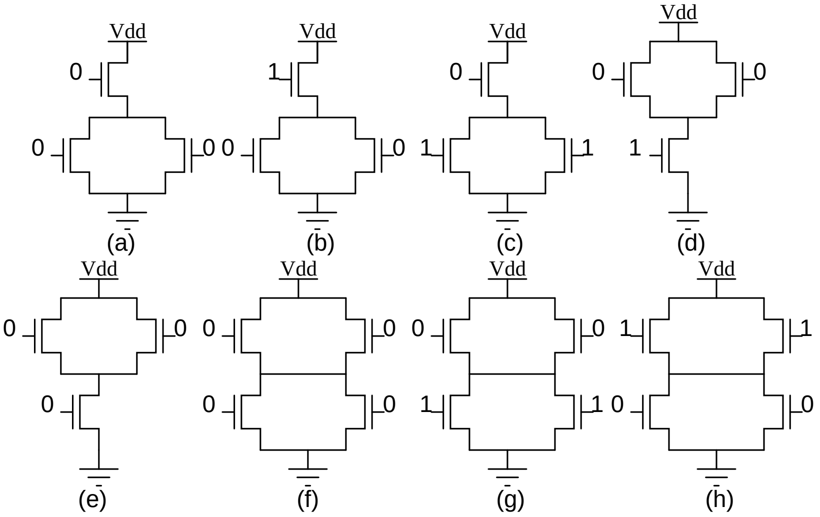


Figure 3.10: Stacks for complex gates consisting parallel transistors - Without combining parallel transistors with same input

Following rules can be derived for the NMOS stack models extraction among different gates in a standard cell library for I_{gate} estimation:-

1. a) If parallel transistors are applied with same inputs, first calculate effective width and replace parallel transistors by a single transistor of effective width.

Rules 2, 3, 4 can be used for equivalent NMOS stack model depending on the input conditions. b) If parallel transistors are applied with different inputs, remove transistors with same voltages at all terminals and then use rules 2, 3, 4 for I_{gate} estimation.

2. If all transistors are in 'OFF' condition, then remove 'OFF' transistors which are not connected to output node. I_{gate} of the NMOS and PMOS stack is estimated using n1g1/0 and p1g1/1 model respectively.

3. If all transistors are not 'OFF' and transistor directly connected to the ground or V_{dd} is not in 'ON' condition. A new stack model is required for I_{gate} estimation of the corresponding NMOS and PMOS stack.

4. If at least one transistor is 'OFF' and only one transistor or series of transistors connected to the ground is 'ON', then remove these 'ON' transistors from the NMOS and PMOS stack. I_{gate} of each removed transistor from NMOS and PMOS stack is calculated by n1g2/1 and p1g2/0 model respectively. Remaining NMOS and PMOS stack can be modeled by rules 2,3.

5. If all transistors are 'ON' in NMOS and PMOS stack. Each transistor in NMOS and PMOS stack is modeled by n1g2/1 and p1g2/0 model respectively.

6. I_{gate} due to parallel stacks simply adds up.

Figure 3.3 shows our I_{gate} models for all possible conditions in single NMOS and PMOS transistor. Our I_{gate} models for NMOS stacks consisting more than one transistor in series are shown in Figure 3.11. Similarly, PMOS stack models can also be derived. Total 26 I_{gate} models are required for I_{gate} estimation of 20 gates with 176 input vectors. Only complex stacks shown in Figure 3.10.(h) need to use effective width equation. Thus, for the considered standard cell library we save in characterization time of two models i.e. one is shown in Figure 3.10.(h) and another is its

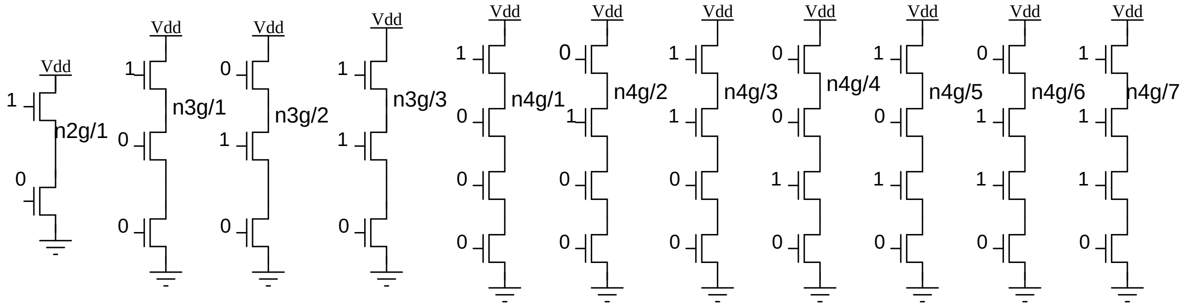


Figure 3.11: I_{gate} estimation-Basic NMOS stack Models consisting series combination of NMOS transistors

PMOS version. But as higher order stacks with parallel transistors are considered, our methodology will result in significant saving of characterization time of parallel transistor stacks. I_{gate} can have greater contribution than I_{sub} to the total power in more advanced technologies and with different input vector applied. We model I_{gate} accurately and efficiently which was not considered in [2].

3.2 Accuracy analysis of proposed stack extraction

We have used $28nm$ Predictive technology model (PTM) model file for all simulations. Inter-die and Intra-die variations have been considered on three process parameters L, V_{th} and T_{ox} with Gaussian distribution (3σ =10%) . supply voltage ($0.6\,V$-$1.2\,V$), temperature ($0^{\circ}C$-100°C) and width ($28nm$-$200nm$) have been sampled considering uniform distribution with same percentage of variation.

3.2.1 Validation of assumptions made in extracting leakage models

We have developed common stack finding methodology across various gates presented in standard cell library for all possible input patterns. In this methodology, basic stacks are used for higher order stacks by neglecting some 'ON' transistors with the loss of some accuracy for I_{sub} estimation while I_{gate} for higher order stacks is calculated by breaking into basic I_{gate} models for single transistor and series connected transistors . In this section, our aim to justify that the error is under tolerable limits in complete PVTW space considering assumptions.

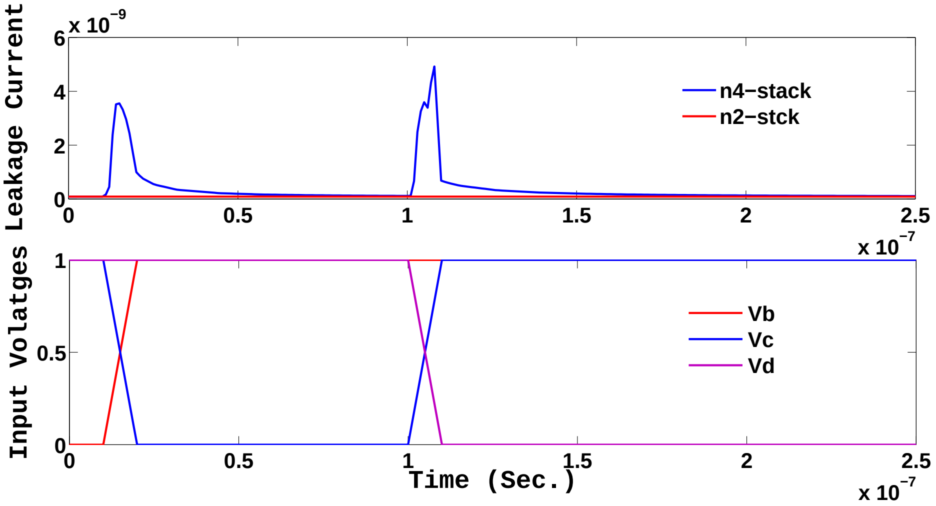


Figure 3.12: Charge Sharing between internal nodes of a 4-input NMOS stack

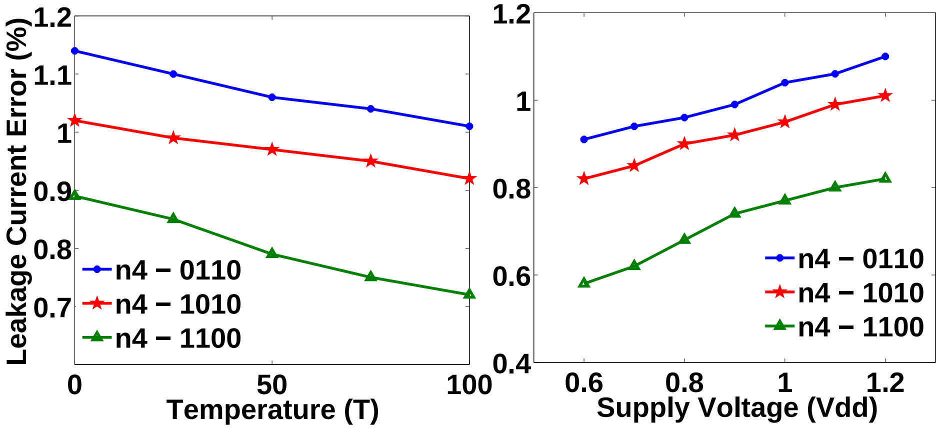


Figure 3.13: Percentage error in leakage current when 4-input NMOS stack with inputs 1100, 1010, 0110 is modeled by 2-input NMOS stack with inputs 00, with respect to (a) Supply voltage (V) (b) Temperature (C)

Whenever there is switching between the inputs, even though same stack is ‘OFF’ but charge sharing occurs between internal nodes in the stack. Since we are using same stack model for ‘1100’, ‘1010’, ‘0110’ inputs of 4-input NAND gate, this charge sharing affects very less in estimating standby I_{sub} for these set of inputs. Consider a 4-input NAND gate as shown in Figure 3.1. First we vary the inputs from '1100' to '1010' and then '1010' to '0110'. Any transistor in 'ON' state can be modeled as a capacitance and resistance. During input switching time interval, due to internal node capacitances, there is a peak in the leakage current as shown in Figure 3.12 but as time passes, the voltage across capacitances saturate and capacitances at internal nodes can be treated as open circuit. In standby mode, only resistance of a transistor will affect the accuracy of model due to voltage drop across them. Our **first assumption** is that the I_{sub} of the 4-input NAND gate with inputs '1100', '1010', '0110' is modeled with n2s/0 model. The 'ON' resistance of a NMOS transistor can be given as follows.

$$R_{ON}(NMOS) = \frac{1}{\mu_n C_{ox}(W/L)(V_{gs} - V_{th})} \tag{3.35}$$

The accuracy of the model is directly governed by the voltage drop across 'ON' transistors. In worse case conditions, maximum resistance will have maximum drop and hence, maximum error in estimating leakage current for higher order stacks using basic stacks. From (3.35), maximum R_{ON} will be for the minimum value of W and maximum value of L, V_{th} and T_{ox}. Now the point is, how this maximum resistance affects the percentage error ($\triangle I_D\%$) in I_{sub} under complete PVTW space. For this purpose, $\triangle I_D\%$ is calculated for all three inputs '1100', '1010' and '0110' for varying V_{dd} and T. We choose these inputs to show the accuracy of I_{sub} stack models due to the fact that the maximum number of 'ON' transistors are removed from these input vectors.

Figure 3.13.(a) shows the error in leakage current for increasing V_{dd} from $0.6\,V$ to $1.2\,V$ with the step size of $0.1\,V$ and T is kept at 25°C. Error is less than 1% across the modeled range of V_{dd} for all three inputs. Maximum error is obtained at $1.2\,V$ due to higher drop across 'ON' transistors. To evaluate the effect of T, we first kept

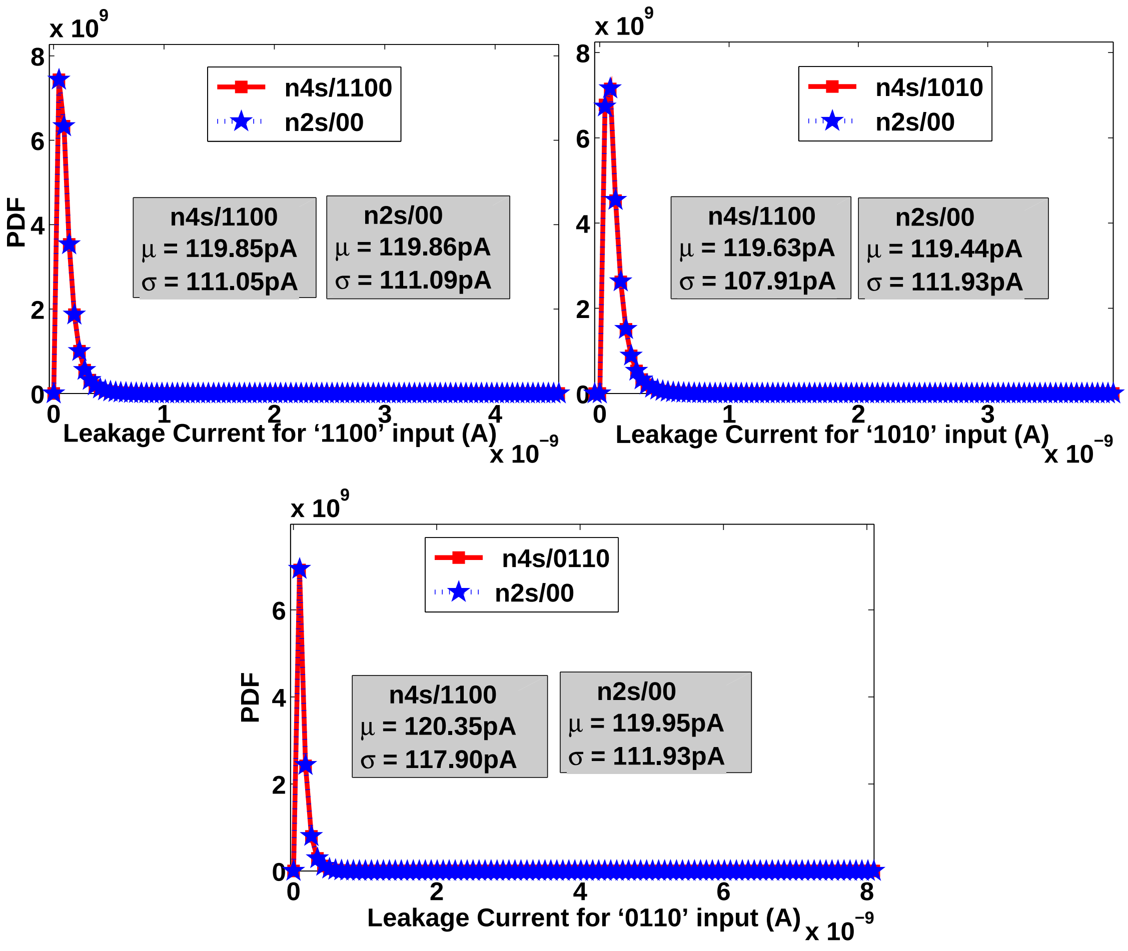


Figure 3.14: PDF curve of actual and model I_{sub} for 4-input gate with '1100', '1010', '0110' input for $W = 28nm$, $T = 27°C$ and V_{dd} =1V and 10000 Monte Carlo Samples

the V_{dd} value at 1.2V and varied the T from 0°C to 100°C with the step size of 25°C. It is important to note that the error characteristics is almost flat in this case i.e. T affects very less to the drop across 'ON' transistors. Increasing the T reduces the drop across the 'ON' transistors and hence improves the accuracy of our models. Out of three inputs, maximum error is for the input which has more number of 'ON' transistors near to the output node. Same analysis can also be performed for stacks consisting of parallel transistors where the error is little bit higher but less than 2% under complete PVTW space.

Figure 3.14 shows the accurate matching of PDF curve of the actual and model I_{sub}.

These PDF curves are generated using 10000 Monte Carlo samples assuming Gaussian Distribution on process parameters using SPICE tool at $W = 28nm$, $T = 27°C$ and V_{dd} =1V. Error in μ and σ is negligible between actual and model output. The order of I_{sub} for these input vectors is $\sim 10^{-9}$, which lies in low error region marked in Figure 3.2. Experiments show high accuracy of our approach in finding the common stack models for I_{sub} across these input patterns of the gates presented in a standard cell library.

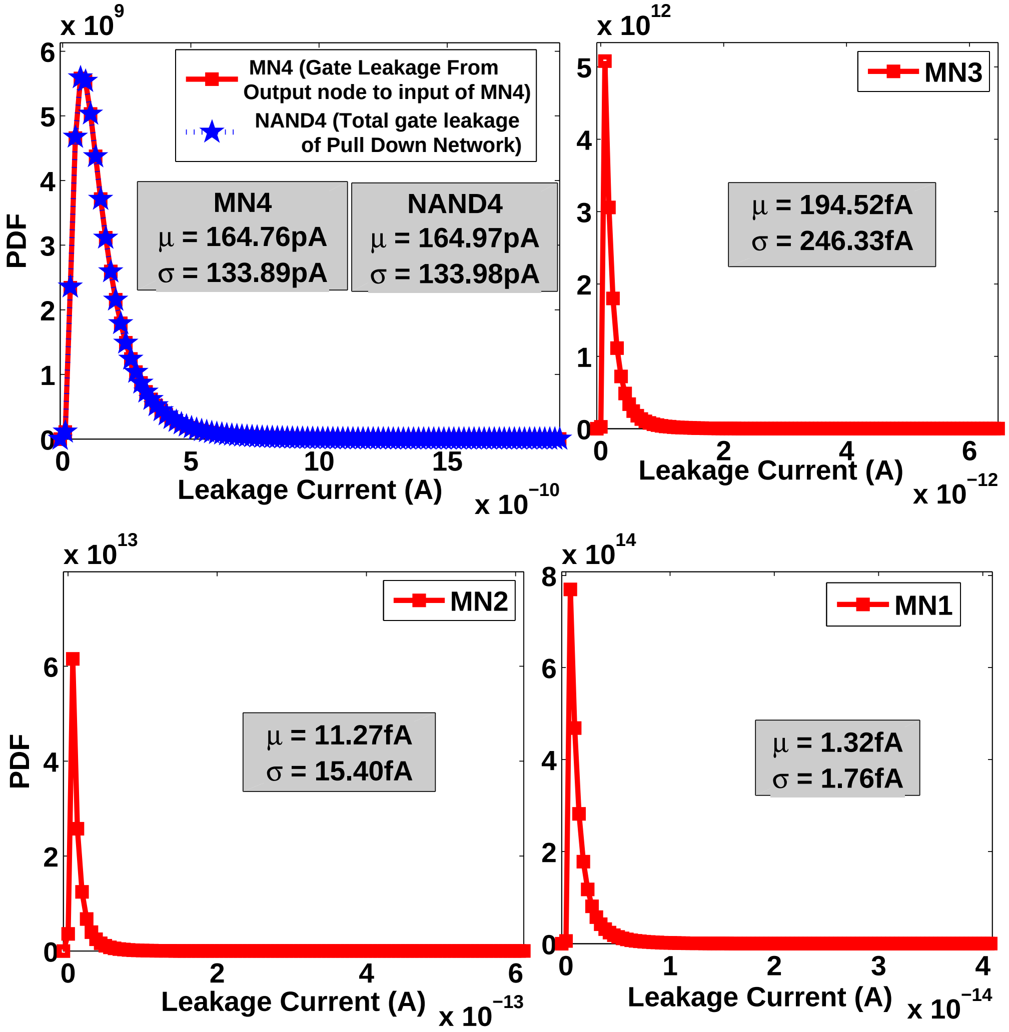


Figure 3.15: PDF curve of Actual and model I_{gate} for PDN of 4-input NAND gate with '0000' input for transistor width MN1-MN4 $= 28nm$ and $T = 27°C$ and V_{dd} =1V and 10000 Monte Carlo Samples.

Second assumption that we made is that I_{gate} for PDN of 4-input NAND gate '0000' input is modeled by n1g1/0 model. All transistors except the one which is connected to the output node, are removed from the stacks. I_{gate} from output node to input of the transistor which is nearest to the output is most significant among all I_{gate} components, thus all other components can be neglected. To verify this assumption, consider PDN of 4-input NAND gate in Figure 3.1 with input '0000' whose PDF curve for every transistor corresponding to each input and total PDN's I_{gate} is plotted in Figure 3.15. In Figure 3.15, width of all transistors is $28nm$, $T = 27°C$ and V_{dd} =1V and PDF curves are generated using 10000 Monte Carlo samples assuming Gaussian Distribution on process parameters using SPICE tool. It shows that the order of I_{gate} for removed transistors is very less and this order increases for the transistors which are nearer to ground voltage. Since, I_{gate} of any transistor is directly proportional to width of that transistor, we now increase the width of removed transistors to $100nm$ and $200nm$ as shown in Figure 3.16 and 3.17 respectively. Still the I_{gate} of the removed transistors is very less, which incurs negligible error in I_{gate} estimation for PDN of 4-input NAND gate with n1g1/0 model.

Second assumption can be generalized for any NMOS and PMOS stack with all transistors in 'OFF' condition as: if all inputs of any NMOS / PMOS stack are applied with '0/1', then all transistors other than the transistor connected to the output node are removed and I_{gate} of this single transistor is estimated using n1g1/0 and p1g1/0 for NMOS and PMOS stack respectively. To verify this assumption, consider a NAND4, NAND3, NAND2 gate with all inputs '0'. With this assumption, I_{gate} of NMOS stacks in three gates must be similar to NMOS transistor of inverter (INV) with '0' input. Figure 3.18 shows the PDF curve of I_{gate} for PDN of NAND4, NAND3, NAND2 and INV. Width of all transistors is $28nm$, $T = 27°C$ and V_{dd} =1V and PDF curves are generated using 10000 Monte Carlo samples assuming Gaussian Distribution on process parameters using SPICE tool. The PDF curve obtained through n1g1/0 model accurately matches with actual PDF curves. The error in μ and σ is also negligible,

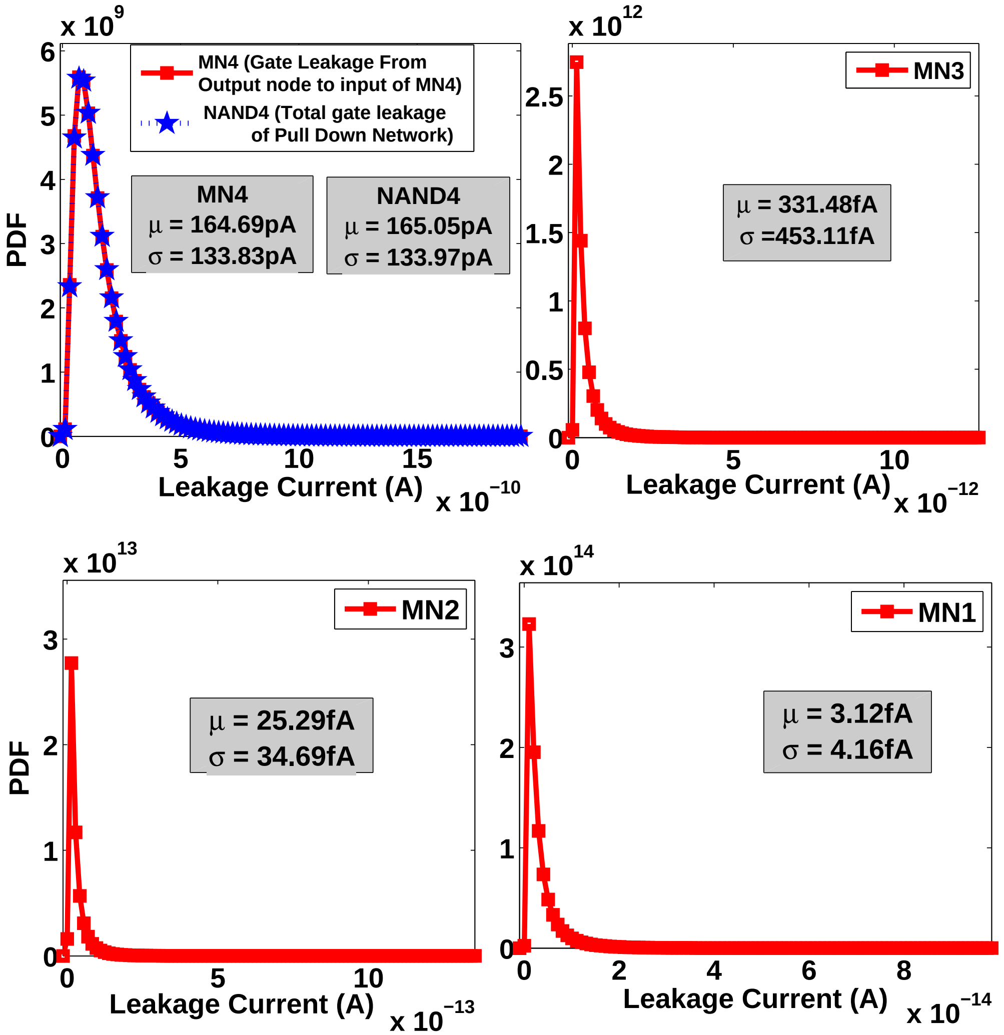


Figure 3.16: PDF curve of Actual and model I_{gate} for PDN of 4-input NAND gate with '0000' input for transistor width MN1 $= 28nm$, MN2-MN4 $= 100nm$ and $T = 27°C$ and V_{dd} =1V and 10000 Monte Carlo Samples.

which justifies the validity of this generalized assumption.

Our **third assumption** is that if any NMOS or PMOS stack is consisting a series of 'ON' transistors, starting from ground or V_{dd}, remove these transistors from stack and each transistor's I_{gate} is estimated using n1g2/1 or p1g2/0 model. Remaining stack is modeled by other I_{gate} estimation rules. Total I_{gate} can be calculated by summing I_{gate} from all models. To verify this assumption, consider a 4-input NMOS stack of NAND4 gate shown in Figure 3.1 with input '1110'. Total I_{gate} can be obtained by using n1g2/1 model for each 'ON' and n1g1/0 for 'OFF' transistor. Figure 3.19

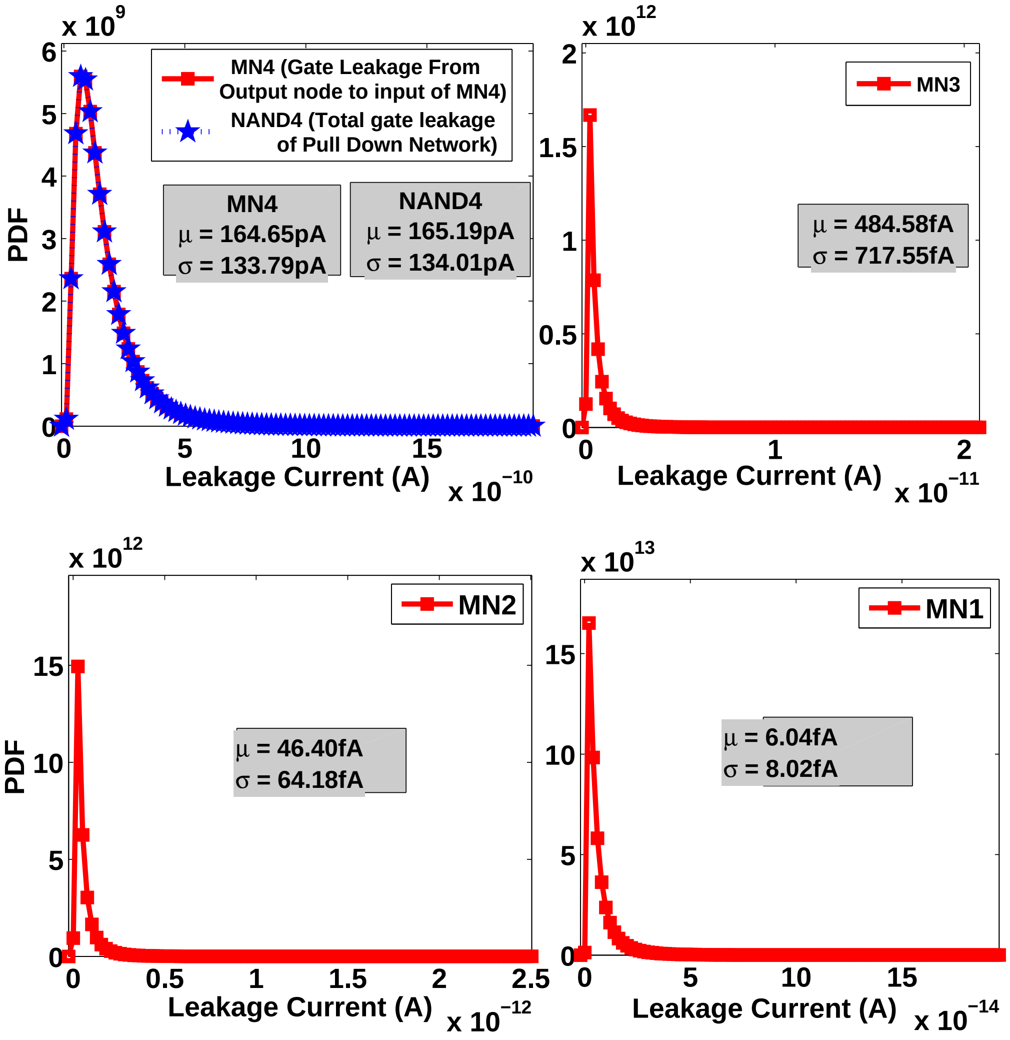


Figure 3.17: PDF curve of Actual and model I_{gate} for PDN of 4-input NAND gate with '0000' input for transistor width MN1 $= 28nm$, MN2-MN4 $= 200nm$ and $T = 27°C$ and $V_{dd} = 1V$ and 10000 Monte Carlo Samples.

shows accurate matching of PDF curve of actual I_{gate} for each transistor with its corresponding model. PDF curve of total I_{gate} calculated by summing I_{gate} from all models also matches with the actual I_{gate} with less than 0.01% error in μ and σ.

It should be noted that we are developing the framework for leakage estimation in idle-time mode excluding leakage during circuit operation. Runtime leakage is also vector dependent and changes whenever input vector changes [104]. Authors in [104] show that the runtime leakage is highly dependent on the switching frequency of inputs. If switching frequency is less, then the total leakage including runtime and idle-time

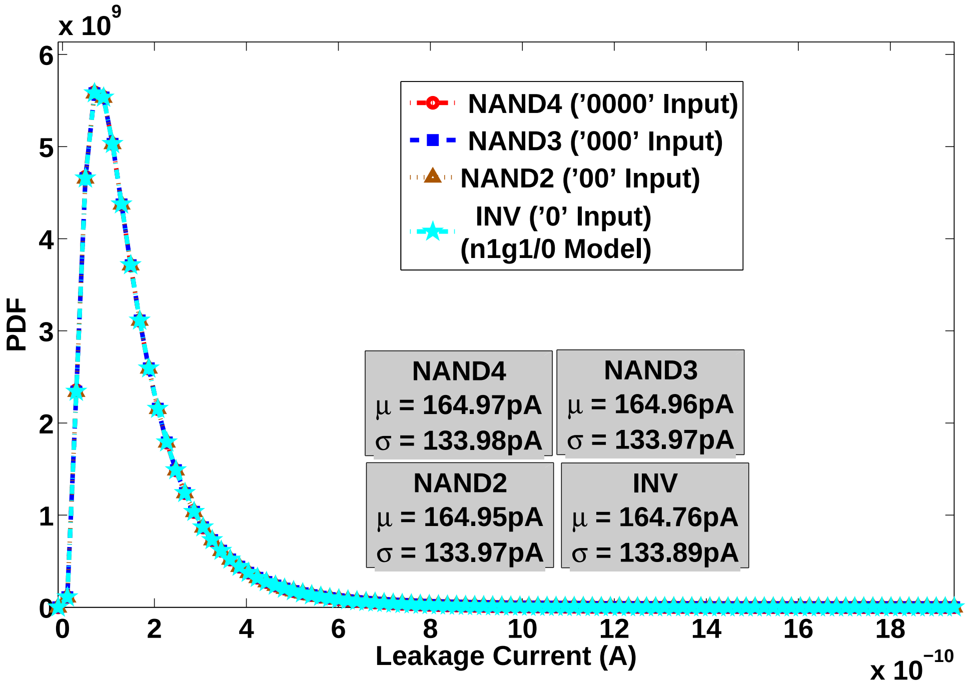


Figure 3.18: PDF curve of I_{gate} for PDN of NAND4, NAND3, NAND2 and INV with all inputs '0' for $W = 28nm$, $T = 27°C$ and V_{dd} =1V and 10000 Monte Carlo Samples

converges to idle-time leakage. To include the runtime leakage current component in this framework, switching between inputs need to be considered but analysis is moreover same as dynamic power estimation of the circuits. Leakage model selection framework need to be redeveloped which will require a larger number of models to include all gates with their inputs switching. A separate analysis is required to account for runtime leakage component. Our analysis in this thesis is consistent with the previous work which only characterizes the idle-time leakage using EQ and ANN based models.

3.2.2 Accuracy of effective width estimation methodology

We proposed a formula in Equation (3.18) to find out the effective width of two parallel transistors with same input. This formula is only valid for the transistors with process parameter and terminal voltage conditions as shown in Figure 3.4.(a).

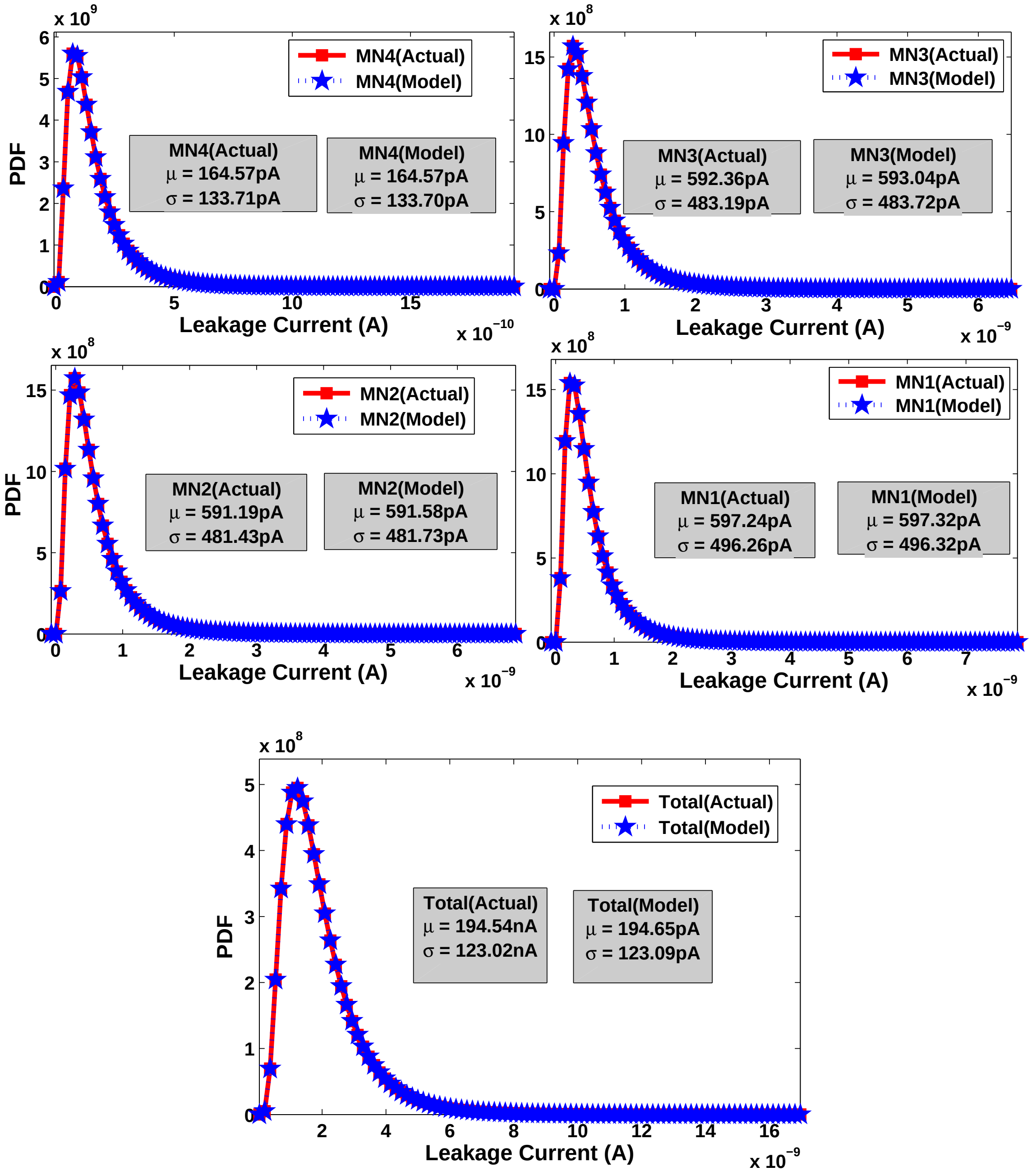


Figure 3.19: PDF curve of Actual and model I_{gate} for PDN of 4-input NAND gate with '1110' input for $W = 28nm$, $T = 27°C$ and V_{dd} =1V and 10000 Monte Carlo Samples

In Figure 3.20, we compare the error in estimating the I_{sub} and I_{gate} for an increasing number of 'OFF' parallel NMOS transistors. Even when the number of parallel transistors is large, our proposed model incurs $\sim 0.15\%$ and $< 0.01\%$ error compared

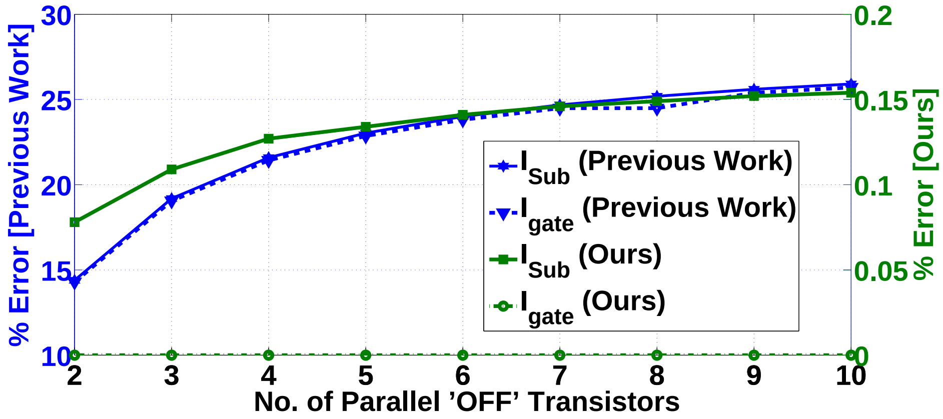


Figure 3.20: Comparison of Error in leakage current between [3] and our approach for varying number of 'OFF' NMOS transistors

to larger than 25% error using model of [3] for 10 parallel 'OFF' transistors in I_{sub} and I_{gate} respectively. In [3], equivalent width is calculated by simply adding the drawn widths of transistors. Similarly, for PMOS transistors, our model incurs very less error compared to previous model.

However, this formula is only valid when process variation is not considered. To calculate effective width even in the presence of process variations, we first calculate the change in width of transistor for the given change in process parameters such that same current flows in both cases. Then Equation (3.18) can be applied to calculate effective width of parallel transistors. In Figure 3.21, we vary the L and T_{ox} of single NMOS 'OFF' transistor within ±10% range from mean value and calculated the I_{sub} with changed process parameters with mean width and mean process parameters with modified width. I_{sub} in both cases accurately matches. Correlation between both cases is $>$0.999 and mean square error is also very low. Maximum and average error of 0.82% and 0.40% is incurred by our effective width estimation methodology for I_{sub} calculation of parallel transistors. Similarly, Figure 3.22 validates the high accuracy of effective width estimation methodology for I_{gate} calculation.

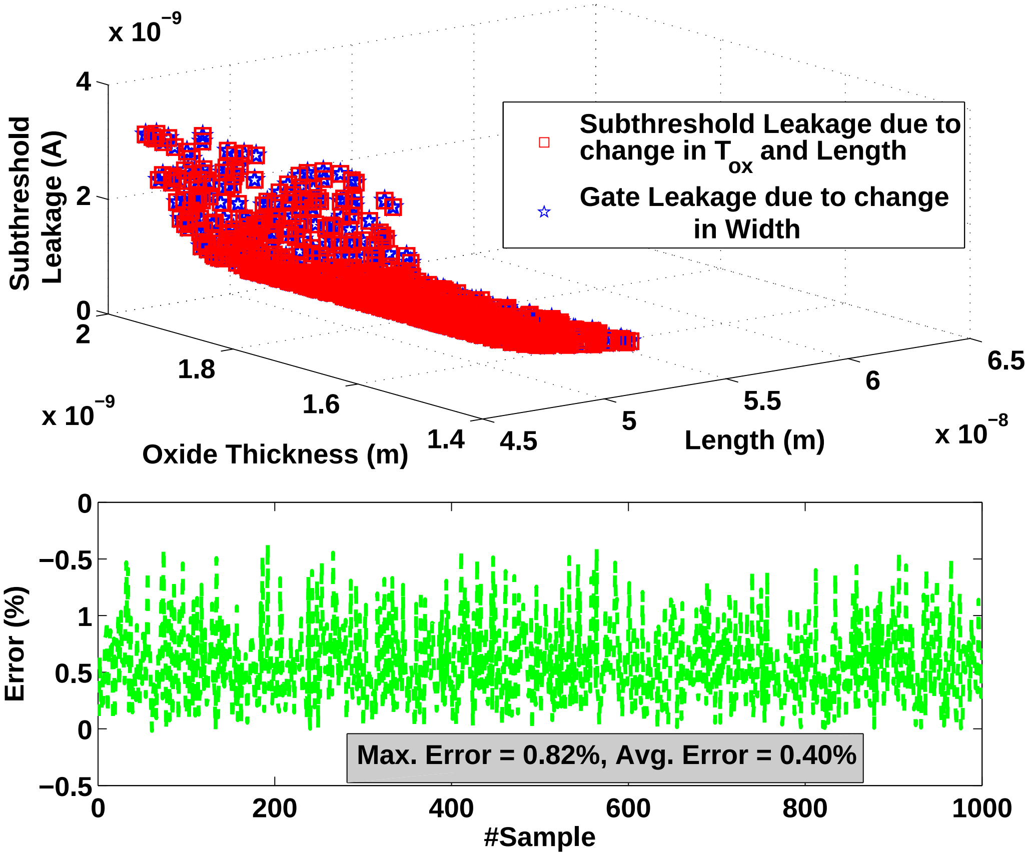


Figure 3.21: I_{sub} estimation- accuracy analysis of effective width estimation methodology; $T = 27°C$ and V_{dd} =1V and 1000 Monte Carlo Samples

3.3 Summary

In this Chapter, we have described the methodology to extract common stack models used to predict the leakage of gates more accurately and efficiently under full PVTW space. Our methodology initiates by first characterizing the leakage of basic stacks and then provides estimate of the leakage in basic gates e.g. NAND4, NAND3, NAND2, NOR4, NOR3, NOR2, INV, based on these stacks. Stack models are developed separately for I_{sub} and I_{gate} estimation. For I_{sub}, stacks are extracted by carefully removing 'ON' transistors from 'OFF' PDN or PUN depending on the input vector combinations. Removal of these 'ON' transistors do not affect the actual I_{sub} of a CMOS gate. While for I_{gate}, PUN and PDN are broken down into a set of basic

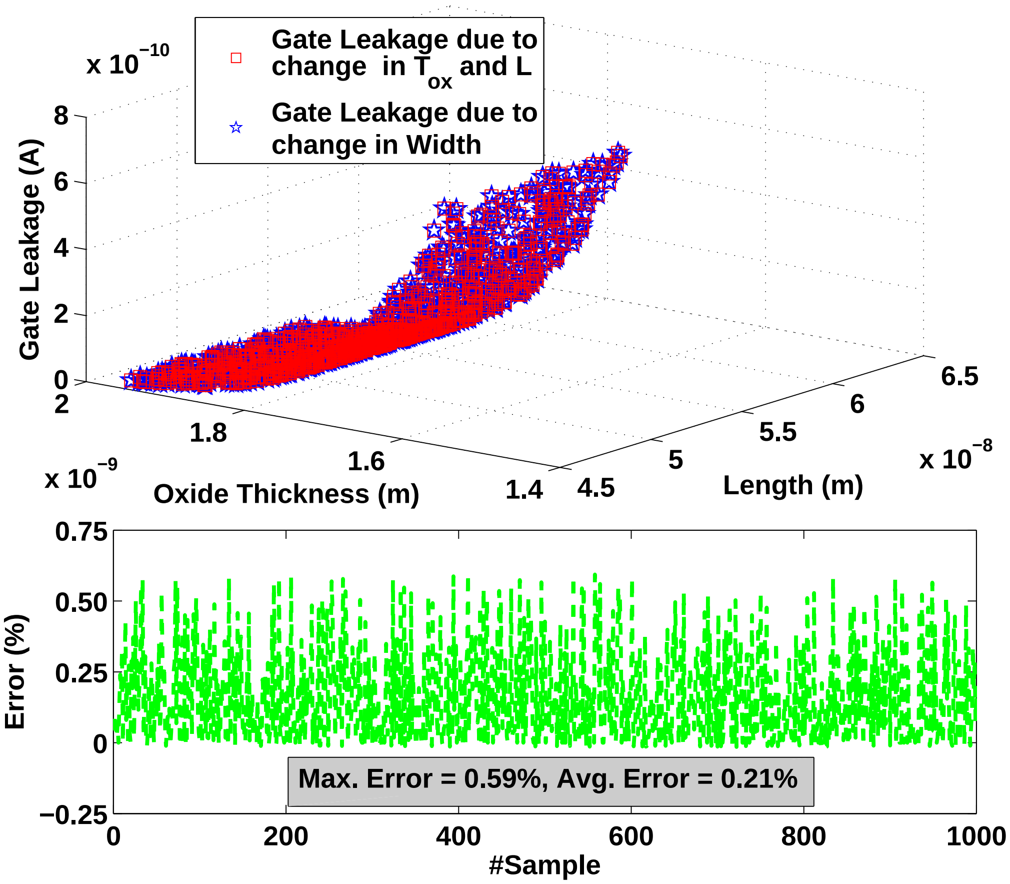


Figure 3.22: I_{gate} estimation- accuracy analysis of effective width estimation methodology; $T = 27°C$ and V_{dd} =1V and 1000 Monte Carlo Samples

stack models. For gates, containing parallel transistor stacks, we replace the parallel transistors having identical inputs with a single transistor of effective width, which in turn allows us to use precharacterized basic stacks for leakage calculation instead of generating new models. Analytical equations for effective width calculation are developed for the parallel transistors in CMOS gates with and without considering process variations. Only 30 basic stacks are modeled for I_{sub} estimation of 7 basic gates - NAND4, NAND3, NAND2, NOR4, NOR3, NOR2, INV. Further, these models can also be used to predict the leakage of standard cell library components such as AND4, AND3, AND2, OR4, OR3, OR2, MUX2×1, D-Flip Flop without the need of characterizing any new model. The gates consisting parallel or series of parallel transistors such as XOR2, MAJ_GATE, AOI22, OAI22 do not require 16 new stack models with the use of effective width computation methodology, which highly reduces our model characterization time. Thus, overall 30 stack models are required for accurately predicting the I_{sub} of 20 gates across 176 input combinations. Similarly, 26 models are developed for I_{gate} estimation. All the approximations made in extracting models incur very small error, thus validating the high accuracy of the proposed stack extraction methodology.

Chapter 4

Support Vector Machine Based Surrogate Models for Leakage Current Modeling

An enhanced version of SVM i.e. Least Squares-Support Vector Machine (LS-SVM) has been proposed in [70]. The main advantage of LS-SVM is that it is computationally efficient while possesses the important properties of SVM. LS-SVM uses the equality type constraints in the problem formulation, thus the solution is obtained through solving a set of linear solutions. Therefore, LS-SVM is free of local minima and also has low computational cost [67]. Thus, LS-SVM is computationally efficient technique than classical SVM for modeling non-linear relationship between input and output parameters.

Consider a set of training data samples $\{(x_1, y_1), (x_2, y_2), ...(x_k, y_k)\} \subset R^N \times R$. where R^N denotes the input space. x_k is the input value and y_k is the the corresponding target value for k^{th} sample. By using the non-linear mapping ϕ, x_k is mapped to y_k using the following relation as follows.

$$\hat{y}(x) = w^T\phi(x) + b \quad with\; w\,\epsilon\, R^N,\; b\,\epsilon\, R \tag{4.1}$$

Here, $\phi(\cdot)$: $R^n \rightarrow R^{n_h}$ is the mapping of input space to some high dimensional feature space (potentially infinite). The approximation error of k^{th} sample can be given as follows.

$$e_k = y_k - \hat{y}_k(x_k) \tag{4.2}$$

In LS-SVM, the minimization error is formulated as primal problem.

$$\mathbf{P}: \quad min\, J_p(w,e) = \frac{1}{2}w^Tw + \gamma\frac{1}{2}\sum_{k=0}^{N} e_k^2 \tag{4.3}$$

with equality constraint as follows.

$$y_k = w^T\phi(x_k) + b + e_k \qquad k = 1, 2, 3.......N \tag{4.4}$$

Here, γ is the regularization parameter. The first term of Equation (4.3) is L_2 norm on regression weights. Second term represents the regression error of all samples. This problem is nothing but ridge regression cost function formulated in feature space. When w becomes infinite, it can not be solved. Therefore, dual problem is developed by constructing Lagrangian as follows.

$$\mathbf{D}: \quad max_\alpha \mathcal{L}(w, b, e, \alpha) \tag{4.5}$$

$$\mathcal{L} = J_p(w,e) - \sum_{k=1}^{N} \alpha_k \left\{ w^T\phi(x_k) + b + e_k - y_k \right\} \tag{4.6}$$

Here, α_k's are the Lagranges multipliers. The conditions for optimality can be given as follows.

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial w} = 0 \rightarrow w = \sum_{k=1}^{N} \alpha_k \phi(x_k) \\ \frac{\partial \mathcal{L}}{\partial b} = 0 \rightarrow \sum_{k=1}^{N} \alpha_k = 0 \\ \frac{\partial \mathcal{L}}{\partial e_k} = 0 \rightarrow \alpha_k = \gamma e_k, \; k = 1, 2.....N \\ \frac{\partial \mathcal{L}}{\partial \alpha_k} = 0 \rightarrow w^T \phi(x_k) + b + e_k - y_k = 0, \; k = 1, 2.....N \end{cases} \tag{4.7}$$

By eliminatinge_k and w through substitution, following solution can be obtained as follows.

$$\begin{bmatrix} \Omega + I_N/\gamma & 1_N \\ 1_N^T & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ b \end{bmatrix} = \begin{bmatrix} y \\ 0 \end{bmatrix} \tag{4.8}$$

Here, $\Omega = Z^T Z$ and the kernel trick can be applied within α matrix as follows.

$$\Omega_{kl} = \phi(x_k)^T \phi(x_l) \tag{4.9}$$

$$\Omega_{kl} = K(x_k, x_l)......k, l = 1, 2....N \tag{4.10}$$

The resulting function is weighted linear combination of inner product between the training points and testing points.

$$y_k = \sum_{k=1}^{N} \alpha_k K(x_k, x) + b \tag{4.11}$$

Here, $K(x_k, x)$ is the kernel function and α_k and b are solutions of the linear systems. For a function to be kernel function, it should be positive definite and must satisfy Mercer condition for the problem to be convex to provide unique and optimum solution. However, a Mercer kernel can be derived from a set of mercer kernels. A list of Mercer kernels is given in Table 4.1

We are evaluating the accuracy of our SVM models using commonly used kernels.

Table 4.1: Mercer kernels, expressions, tuning parameters and their ranges

Kernels	*Expressions*	*Tuning Parameters*
Linear	$x_k^T x$	$0 < \gamma \leq 1000$
RBF	$e^{(\frac{\|\|x-x_k\|\|^2}{\sigma^2})}$	$0 < \gamma,\ \sigma \leq 1000$
Poly	$(x_k^T x + t^2)^d$	$0 < \gamma,\ t \leq 1000, 0 < d \leq 5$
MLP	$tanh(s * x_k^T x + t^2)$	$0 < \gamma,\ s,\ t \leq 1000$
Power	$-\|\|x - x_k\|\|^\beta$	$0 < \gamma \leq 1000, 0 < \beta \leq 2$
Log	$-log(1 + \|\|x - x_k\|\|^\beta)$	$0 < \gamma \leq 1000, 0 < \beta \leq 2$

However, there may exist other kernels which may provide higher accurate and less complex models than we have considered in this thesis. Adding more number of kernels increase the characterization time to develop the final models. Thus, choosing the number of kernels is a trade-off between characterization time, accuracy and runtime of the models. Furthermore, tuning parameters of a kernel should be selected carefully for better regression task.

Before elaborating our methodology in Algorithms 4.2 and 4.1, we have described symbols used in this algorithm with their meaning as shown in Table 4.2. Contributions of the proposed methodology are as follows.

- SVM regression based approach is adopted for modeling leakage of each stack as derived in Chapter 3 for such a large PVTW space, which requires less characterization time and is more accurate than look-up table and analytical equation based techniques.

- Efficient methodology is developed for finding the best kernel among various kernels and finding its corresponding optimum tuning parameters locally and globally in the tuning parameter range. Two way error driven active learning methodology is also employed that selects the new samples from the input space for adaptive training of SVM based surrogate models.

- Sparse SVM models are also developed using Support Vector spectrum pruning

Table 4.2: Symbols used in algorithm 4.1

Symbol	*Meaning*
$X_{train,initial}$ $Y_{train,initial}$	Initial input and output data to train model
X_{test}, Y_{test}	Input and output data to evaluate trained model
$Error_{req}$	Required error of trained model to stop algorithm
$Y_{test,est}$	Estimated output from trained model for X_{test} data
$Error_{est}$	Error between Y_{test} and $Y_{test,est}$
Best_Tune_Param	Best tuning parameters with minimum error
Train_SVM, Test_SVM	Training and testing algorithm from SVM toolbox
Out_Best_Kernel, Out_Best_Tune_Param	Best kernel and best tuning parameters
$Error_{sample}$	Error for one sample of X_{test}
N_{sample}	No. of current samples to train model
Max_{sample}	Maximum samples to train model
X_{error}	List of trained samples with maximum error
$X_{dominant}$	List of training samples with maximum α value
X_{active}	Total samples combining X_{error} and $X_{dominant}$
$X_{active-spice}$	Samples around X_{active} for active learning
$Y_{active-spice}$	SPICE output for samples $X_{active-spice}$
$X_{train-final}$	Total samples after generating active samples
$Y_{train-final}$	SPICE output for samples $X_{train-final}$

method, which reduces the number of training samples to prepare regression model. The resulting model reduces the runtime while negligible increase in the error.

- A methodology is proposed for efficient selection of Monte Carlo samples for fast estimation of leakage current of larger CMOS circuits.

- Our stack based methodology can be used for leakage characterization of post CMOS devices i.e. FinFET, CNTFET based logic gates.

- Proposed models can be used for leakage estimation of CMOS gates for non-Gaussian process parameter variations and does not require to re-characterize the models. The proposed methodology removes the inaccurate log-normal assumption of leakage with respect to process parameters.

4.1 SVM Kernel Parameter Optimization and Active Sample Selection Method

In our methodology, we combine both grid search based technique with genetic algorithm (GA) and differential evolution (DE) to explore the tuning parameter space in guided manner. It is a two way iterative search method in which search space is explored locally and globally both. If any good solution is found than the previous step, then our local grid search technique quickly explores the space near that good solution. Our global search technique randomly searches in the parameter space to find better solutions if any solution exists, which is better than the previous step.

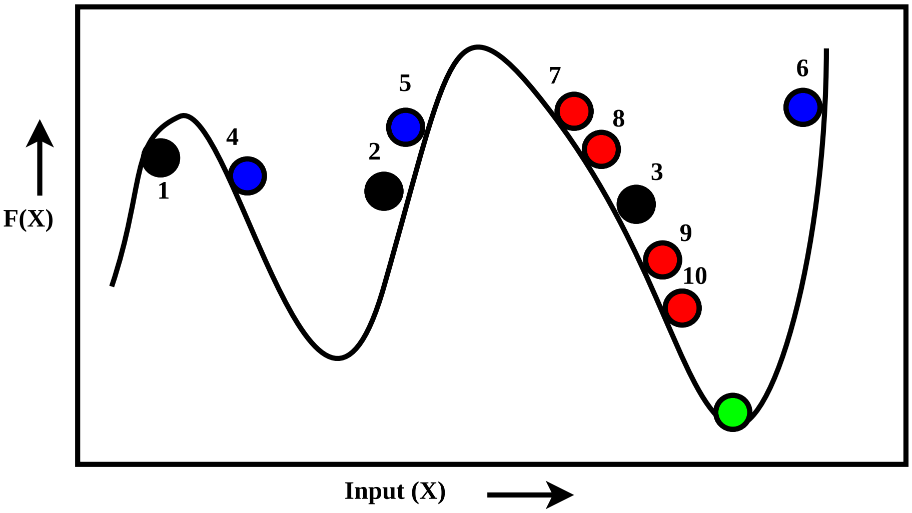


Figure 4.1: Random function F(X) as function of variable X.

To understand the importance of our approach, consider a random function F(X) as a function of variable X as shown in Figure 4.1. Our aim is to obtain green sample which gives minimum value of F(X) in minimum possible time. Samples 1, 2 and 3 are the initially generated samples in which sample 3 denotes the minimum error sample out of samples 1, 2 and 3. Grid based algorithm may give sample 3 as minimum error sample for function F(X). In genetic algorithm, next set of samples are generated randomly with minimum error sample propagated from previous step. Suppose samples 4, 5 and 6 are generated in next step which are generated randomly in parameter space of variable X. Still the minimum error sample is sample 3. But if a local grid is defined at sample 3 then some samples are generated around sample 3. Suppose samples 7, 8, 9 and 10 are generated around sample 3 in conjunction with randomly generated global samples. Thus in second step, sample 10 is minimum error sample which is more nearer to global minimum error sample (green circle) than sample 3. Thus, our algorithm finds best tuning parameters in lesser time than only grid based algorithm and only genetic algorithm/differential evolution.

In the development process of SVM models, we want to use minimum possible number of training samples to reduce the runtime of models. Equation (4.11) is used to evaluate the value of unknown samples. The number of α_k values is directly proportional to the number of training samples. These α_k values affect the computations required to evaluate the newly generated samples (i.e. other than training data). Active learning method is used to generate new samples in next step based on the error between the developed model with small training data-set and SPICE output. In our methodology, we use active learning process to generate new training samples around maximum error sample to achieve desired accuracy. We start training with fewer samples and only generate two samples around maximum error sample to add them with previous training data.

Furthermore, authors in [105] suggest to model I_{sub} as a exponential linear or quadratic model form. If we take *log* of the leakage current, then the exponential linear or quadratic model can be converted to simple polynomial model with linear or quadratic terms which is easier to model with less complexity. More terms can also be added to improve the accuracy of model. Thus, modeling *log* of leakage allows us to characterize less number of models with larger number of parameters for larger range in a single model which consequently reduces the time to develop the models and runtime for larger CMOS circuits. Errors in modeling using exponential quadratic form is evaluated in Section 4.3 and compared with our proposed models. We choose *log* of leakage current as performance parameter to be modeled using SVM.

Our methodology for selecting the best kernel with its corresponding optimized tuning parameters is presented in Algorithm 4.1. This algorithm takes input as initial random training samples, testing samples, kernels list, and kernel tuning parameter ranges in which kernel parameters need to be optimized. Line 1 to 23 denotes our best kernel selection method and its corresponding tuning parameters. In line 1, m denotes the number of kernels available. Lines 2-4 select a kernel from the kernel list and generate the initial predefined random set of samples t in the tuning parameters range. In line 6-8, SVM model is trained using initial random samples $X_{train,initial}$, output $Y_{test,est}$ is calculated using trained SVM model and model mean square error (MSE) is calculated as in Equation (4.12).

$$Error_{est} = \mathrm{MSE}(Y_{test},\ Y_{test,est}) = \frac{1}{n}\sum_{i=1}^{n}(Y_{test} - Y_{test,est})^2 \tag{4.12}$$

Line 11 selects the best set of tuning parameters having least error from initial set of training parameters. Lines 12 and 13 define the search range around previously found best set of tuning parameters and generate predefined set of samples s_1 in this search range. Line 14 generates the s_2 samples in the whole tuning parameter range, which is the size of the population (Number of chromosomes) used in GA and DE based global optimization algorithms and hence, s denotes the effective population size of each generation of optimization problem. In line 16-19, SVM model is trained, tested and error is calculated for every set of tuning parameters in $\mathrm{s}_1$and s_2 using the Equation (4.12). Lines 11-19 are repeated k times for each kernel in kernel_list ,

Algorithm 4.1 Algorithm to find out best kernel and optimal kernel parameters

```
Input: $X_{train,initial}$, $Y_{train,initial}$, $X_{test}$, $Y_{test}$, Tune_Param_Range, $Error_{req}$, Kernels_list (RBF, log, linear, poly, MLP, power).
Output: Best_Kernel, Best_Tune_Param.
1. for i = 1 to m
2. Select Kernel$_i$ from Kernels_list
3. Tune_LSSVM(Kernel$_i$, Tune_Param_Range)
4. Generate t random sets of Tuning Parameters
5.     for n = 1 to t
6.         Train_LSSVM($X_{train,initial}$, $Y_{train,initial}$)
7.         $Y_{test,est}$←Test_LSSVM($X_{test}$)
8.         $Error_{est}$ ← MSE($Y_{test}$, $Y_{test.est}$)
9.     end
10.     for n = 1 to k
11.             Select Best_Tune_Param with minimum Error
12.             Define search range around Best_Tune_Param
13.             Draw s$_1$ random samples in this search range (Grid search Method).
14.             Draw s$_2$ random samples in the Tune_Param_Range (Global Search method).
15.              s ← [s$_1$; s$_2$]
16.             for n = 1 to s
17.                 Train_LSSVM($X_{train,initial}$, $Y_{train,initial}$)
18.                 $Y_{test,est}$←Test_LSSVM($X_{test}$)
19.                 $Error_{est}$ ← $MSE$($Y_{test}$, $Y_{test.est}$)
20.             end
21.     end
22.  Out_Best_Tune_Param ← Best_Tune_Param
23. end
24. Out_Best_Kernel ← {Best_Kernel, Best_Tune_Param}
25. end
```

where k denotes the number of generations in GA or DE. Predefined set of numbers k, s$_1$ and s$_2$ must be chosen in such a way that algorithm can find the optimum values of tuning parameters. Line 22 saves best set of tuning parameters for every kernel and line 23 saves best kernel with with its optimum tuning parameters. Figure 4.2 shows the MSE and time taken in predicting the output for different kernels. This figure is produced by training and testing the models with 100 and 5000 samples respectively for 100 set of tuning parameters. For some set of tuning parameters, MSE is lower but time taken by model is little bit high and vice-versa. Hence, we take into the effect of both performance parameters by defining the new fitness function in the global optimization problem as follows.

$$Fitness - func = w_1 * MSE + w_2 * time \tag{4.13}$$

Here, *w₁* and *w₂* are the weights assigned to both parameters and denotes how much importance should be given to both parameters.

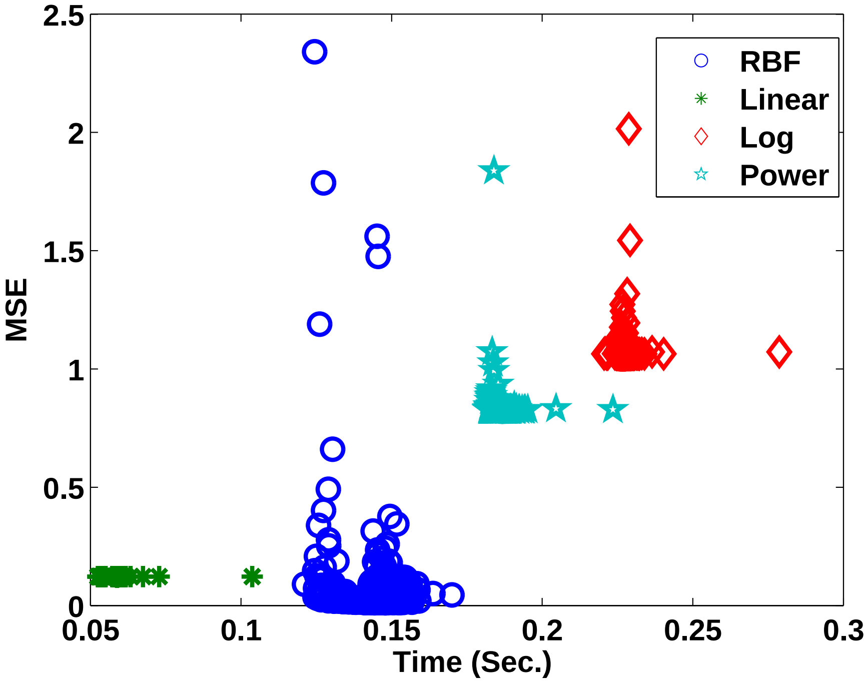


Figure 4.2: Comparing error and time - different kernels

Algorithm 4.2 adaptively generate the samples in the input space. This algorithm takes the best kernel and initial training data samples and provides the final set of training samples. if error of initially trained SVM model is lesser than the required value $Error_{req}$, then our algorithm stops otherwise it initiates our active learning algorithm which efficiently generates the training samples in the input space. These samples are added to initial random training sample set to further reduce the $Error_{est}$ between model estimated output $Y_{test,est}$ and SPICE output Y_{test} for test samples X_{test}. In line 9, samples from the input space are separated in which error between predicted and actual output is maximum. The error related to every sample is calculated using as in Equation (4.14)

$$Error_{sample} = \left|Y_{test} - Y_{test,est}\right|^2 \tag{4.14}$$

Equation (4.14) automatically takes care of generation of samples in empty area of the input space and distance between predicted and actual output rather than taking two different functions for both purpose as in [76]. Line 10 selects the samples from the set of previous training samples having maximum values of α_k weights (the Lagranges multipliers). According to Equation (5.22), we can see that these α_k weights are directly proportional to errors e_k in the training samples. To further reduce the error e_k, samples are generated around the samples having maximum α_k weights. To further speedup the process, we select s_3 and s_4 number of samples.

Figure 4.3.(a) shows the decreasing values of α and Figure 4.3.(b) shows the MSE in subsequent iterations which shows that the significant decrement in the error due to dominant α based active learning method. Figure 4.3.(b) also indicates that the number of samples taken from the previous step for which dominant α values are considered, do not effect the MSE. We select 10 samples from the previous step. These samples are now taken to the SPICE, where s_5 samples are generated around each sample in s_3 and s_4 and corresponding SPICE output is calculated. These samples are now added to previous training data-set and SVM model is trained, tested and error is calculated. This process is repeated until estimated error, $Error_{est}$ is lesser than the $Error_{req}$ or number of training samples are exceeded than the maximum

Algorithm 4.2 Adaptive SVM model learning algorithm

```
Input: $X_{train,initial}$, $Y_{train,initial}$, Best_Kernel, Best_Tune_Param.
Output: $X_{train,final}$, $Y_{train,final}$
Train_LSSVM($X_{train,initial}$, $Y_{train,initial}$, Best_Kernel)
$Y_{test,est}$←Test_LSSVM($X_{test}$)
$Error_{est}$← MSE($Y_{test}$, $Y_{test.est}$)
if ($Error_{est} < Error_{req}$)
    then stop
else
    $Error_{sample} \leftarrow |Y_{test}$ - $Y_{test.est}|^2$
while ($Error_{est} > Error_{req} \,||\, N_{sample} < Max_{samples}$)
    $X_{error} \leftarrow$ Separate samples $s_3$ with maximum error
    $X_{dominant} \leftarrow$ Separate samples $s_4$ with maximum $\alpha$ values
    $X_{active} \leftarrow [X_{error}; X_{dominant}]$
    $X_{active-spice} \leftarrow$ Generate samples $s_5$ around $X_{active}$
    $Y_{active-spice} \leftarrow$ Simulate_SPICE($X_{active-spice}$)
    $X_{train,final} \leftarrow [X_{train,initial}; X_{active-spice}]$
    $Y_{train,final} \leftarrow [Y_{train,initial}; Y_{active-spice}]$
    Train_LSSVM($X_{train,final}$, $Y_{train,final}$)
    $Y_{test,est}$←Test_LSSVM($X_{test}$)
    $Error_{est} \leftarrow MSE(Y_{test}, Y_{test.est})$
    $Error_{est} \leftarrow |Y_{test}$ - $Y_{test.est}|^2$
    $X_{train,initial} \leftarrow X_{train,final}$
    $Y_{train,initial} \leftarrow Y_{train,final}$
end
```

number of training samples.

4.2 Sparse SVM using Support Vector (SV) Spectrum Pruning Method

Sparseness is an important property of any regression based model, which removes the non-dominated (outliers) samples from the training data-set used in model prepa-

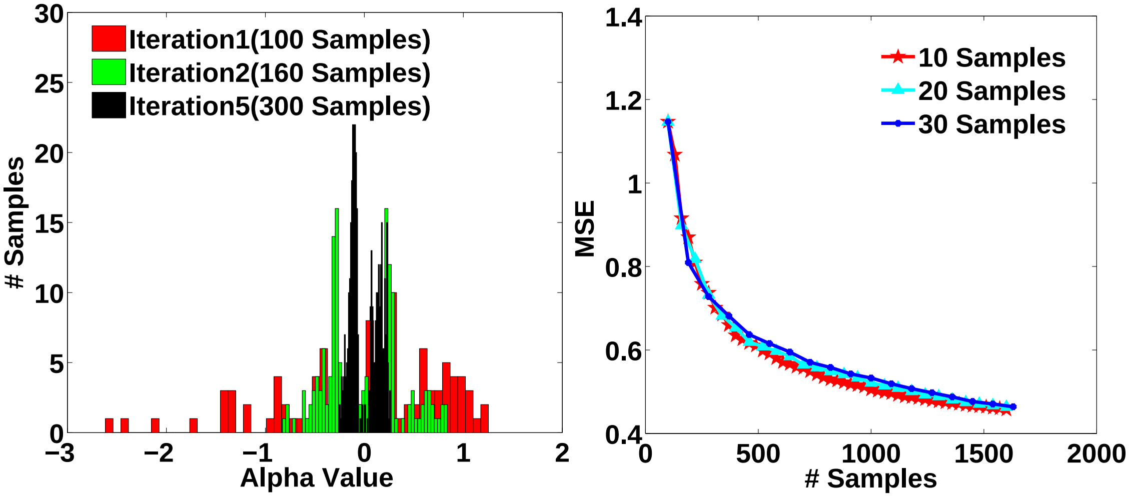


Figure 4.3: Dominant α based active learning method (a) Alpha value for different iterations (b) Mean Square Error (MSE) for different set of samples

ration to reduce the model complexity. With the advantage of high dimensional complexity reduction of LS-SVM over SVM, it has one major drawback of loss of sparseness in SVM. In LS-SVM, all training samples take part in computing the unknown samples as opposed to traditional SVM case. It is due to the simplification of equations into linear form. To define a generalized sparse regression problem, consider an output sample $y \epsilon R^m$ and a dictionary $D \epsilon R^{m \times n}$(the columns of D are referred to as the atoms), sparse vector x can be given as follows.

$$P: \qquad min \parallel x \parallel_0 \quad s.t. \qquad y = Dx \tag{4.15}$$

Here, $\parallel x \parallel_0$ (referred as ℓ_0-norm) is the cardinality or number of nonzero elements in training sample vector x. Many algorithms such as Greedy algorithm [106], Pruning based algorithm [77], Orthogonal Matching Pursuit (OMP) algorithm [107], Coordinate descent algorithm with non-convex penalties [108] have been proposed in literature. In this thesis, our main focus is on pruning based algorithm, which is very simple to apply yet effective in reducing complexity of the regression model. However, other algorithms can also be used in place of pruning based algorithms.

Karush-Kuhn-Tucker conditions in Equation (4.8) can be cast into generalized sparse regression problem as in Equation (4.15).

$$P: \quad min \left\| \begin{bmatrix} \alpha \\ b \end{bmatrix} \right\|_0 \quad s.t. \quad \begin{bmatrix} y \\ 0 \end{bmatrix} = \begin{bmatrix} \Omega + I_N/\gamma & 1_N \\ 1_N^T & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ b \end{bmatrix} \tag{4.16}$$

Removing an element from vector α is equivalent to removing a sample from the training samples because number of α values will be as much as the number of training samples. Thus sparseness in LS-SVM can be imposed by removing the outliers, which is identified by observing the SV-spectrum ($|\alpha_k|$ values). These $|\alpha_k|$ values are the solution of linear equations in LS-SVM formulations and denotes the important of the training samples in evaluation of unknown samples. Sparse LS-SVM has an advantage over Artificial Neural Network (ANN), Radial Basis Function (RBF) because deleting any training sample from the trained network requires Hessian or it's inverse to be calculated, which is a more computational complex than solving linear equations. Equation (5.22)C shows that the α_k's are directly proportional to the error e$_k$ of the model. In Equation (4.11), output for a test sample x_k will be more affected by high values of α_k. Thus, by simply removing the samples with smaller $|\alpha_k|$ values, runtime of the model can be greatly reduced while incurring a small error in the model. The algorithm can be described in formal steps below. These steps are performed for best kernel only.

Step (1) - Load the trained model obtained from algorithm in Figure 4.2.

Step (2) - Prune 2% set of training samples having lower $|\alpha_k|$ values in SV-spectrum.

Step (3) - Re-train the model with reduced sample set.

Step (4) - If Error$_{est}$ < Error$_{req}$, then go to step 1. Else, stop.

4.3 Results

We have used $28nm$ Predictive technology model (PTM) model file for all simulations presented in this Chapter. For model characterization, LS-SVM toolbox is used in MATLAB (version 2007b). Inter-die and Intra-die variations are considered on three process parameters L, V_{th} and T_{ox} with Gaussian distribution (3σ =10%) . V_{dd}, T and W are sampled with uniform distribution.

4.3.1 Accuracy and timing analysis of SVM models

In Table 4.3 and 4.4, we compare our methods and genetic algorithm for finding the best kernel and its optimal tuning parameters. The values of tuning parameters with our first method (grid search + GA) and second method (grid search + DE) is exactly similar but our method is 2× and 3× faster than genetic algorithm respectively, which means our algorithm searches the tuning parameter space faster than genetic algorithm. For all stack models, RBF kernel is the best kernel but the value of optimal tuning parameters is different for different set of models. The best kernel with optimal tuning parameters highly depends upon the stack type and the input applied.

Table 4.5 and 4.6 Shows the information of some of the SVM and Sparse SVM model for NMOS and PMOS stacks for I_{sub} and I_{gate} calculation. We use maximum 1540 samples for training of SVM model which gives sufficient accuracy of our stack models. ρ for testing data of all basic stacks is greater than 0.999 and MSE is also very less. Our SVM model will require ≈4.13 sec. to predict the PDF at any V_{dd} and T value for 10000 Monte Carlo samples. For SVM model, time required to predict the leakage is almost same for all stacks which suggests that the model is very less dependent on inputs applied to the same type of stack model i.e. runtime for n4/0000 and n4/0001 is approximately same. Now, We apply SV-spectrum pruning algorithm from Section 4.2 to check whether the reduction in the number of training samples in possible.

Table 4.3: Best kernel and optimal tuning parameters - proposed Grid Search + GA method

Model	*Best*	*GA method*			*Grid Search + GA*			*Speed*
	Kernel	*Tuning Param.*		*Training*	*Tuning Param.*		*Training*	*up*
	GA/M1	γ	σ^2	*Time (S)*	γ	σ^2	*Time (S)*	*(×)*
n4s/0	RBF	1000	313.08	455.01	1000	313.17	220.29	2.07
n3s/0	RBF	1000	273.20	454.25	1000	273.50	218.33	2.08
n2s/0	RBF	1000	211.04	419.06	1000	210.37	205.11	2.04
n1s/0	RBF	1000	173.94	411.83	1000	173.69	201.69	2.04
p4s/15	RBF	236.4	1000	427.80	236.5	1000	208.90	2.05
p3s/7	RBF	189.8	409.48	421.72	190.4	409.97	205.33	2.05
p2s/3	RBF	1000	408.40	412.78	1000	407.64	203.81	2.03
p1s/1	RBF	232.1	147.51	374.77	231.9	147.50	189.58	1.98
n4g/7	RBF	1000	213.08	426.37	1000	213.08	209.79	2.03
n3g/3	RBF	1000	119.07	420.89	1000	119.07	201.45	2.09
n2g/1	RBF	1000	541.45	395.55	1000	541.45	190.56	2.07
n1g1/0	RBF	1000	310.68	389.69	1000	310.68	188.50	2.07
p1g1/1	RBF	100.7	383.33	369.57	100.7	383.33	185.45	1.99

Table 4.4: Best kernel and optimal tuning parameters - proposed Grid Search + DE method

Model	*Best*	*Grid Search + DE*			*Speed*
		Tuning Parameters		*Training*	*up*
	kernel	γ	σ^2	*Time (S)*	*(×)*
n4s/0	RBF	1000	313.09	154.54	2.95
n3s/0	RBF	1000	273.51	149.10	3.04
n2s/0	RBF	1000	210.37	148.30	2.83
n1s/0	RBF	1000	173.67	145.66	2.83
p4s/15	RBF	236.5	1000	149.99	2.85
p3s/7	RBF	1000	409.97	148.73	2.84
p2s/3	RBF	1000	407.64	144.59	2.85
p1s/1	RBF	231.9	147.50	140.21	2.67
n4g/7	RBF	1000	212.05	136.66	3.12
n3g/3	RBF	1000	121.01	132.43	3.18
n2g/1	RBF	1000	177.35	131.47	3.01
n1g1/0	RBF	1000	310.53	125.33	3.11
p1g1/1	RBF	100.7	383.33	125.01	2.97

In each step, we remove 2% of the samples from the trained model in SVM case. Results in Table 4.6 show the reduction in the runtime of the models without loosing the accuracy. Here, it can be concluded that with the increased time on finding the optimal tuning parameters, active learning of the model and developing sparse regression model, complexity and simulation time of the model can be reduced while keeping low MSE with respect to SPICE. However, modeling is one time process, more time on training part can be invested with the advantage of low MSE and low runtime of the model. Until the number of samples are same, runtime of the model will be same.

Table 4.5: Evaluating proposed SVM method - NMOS & PMOS stacks in PVTW space; test samples = 10000. (T_{train} → Model training time, T_{sim} → Model simulation time)

Model	*Input*	*Our Method (SVM)*				
		Training Samples	T_{train} *(S)*	*MSE* ($\times 10^{-6}$)	ρ	T_{sim} *(S)*
n4s/0	0000	1540	1564.20	8.44	0.9991	4.1938
n3s/0	000	1540	1514.30	7.90	0.9992	4.1877
n2s/0	00	1540	1504.80	3.80	0.9995	4.1870
n1s/0	0	1540	1154.40	1.71	0.9997	4.1864
p4s/15	1111	1540	970.049	8.93	0.9990	4.2858
p3s/7	111	1540	735.457	7.94	0.9991	4.2595
p2s/3	11	1540	672.165	4.09	0.9993	4.2504
p1s/1	1	1540	485.027	1.97	0.9996	4.2476
n4g/7	0111	1540	1563.32	5.01	0.9993	4.0812
n3g/3	011	1540	1510.93	3.42	0.9995	4.0545
n2g/1	01	1540	1501.46	1.23	0.9996	4.0476
n1g1/0	0	1540	944.79	0.12	0.9998	4.0321
p1g1/1	1	1540	265.83	0.13	0.9998	4.0485

Figure 4.4 shows the PDF predicted by our models and SPICE for different V_{dd} and T ($27°C$), which illustrates that both SVM and Sparse SVM model predict the PDF quite accurately and as well as for different T values at fixed V_{dd} ($1V$).

Table 4.6: Evaluating proposed Sparse SVM method - NMOS & PMOS stacks in PVTW space; testing samples = 10000. (T_{train} → Model training time, T_{sim} → Model simulation time)

Model	*Input*	*Our Method (Sparse SVM)*				
		Training Samples	T_{train} *(S)*	*MSE* ($\times 10^{-6}$)	ρ	T_{sim} *(S)*
n4s/0	0000	1310	2267.65	8.47	0.9990	2.9666
n3s/0	000	1250	2109.43	7.91	0.9991	2.8217
n2s/0	00	1250	2089.90	3.82	0.9994	2.8159
n1s/0	0	1220	1445.40	1.73	0.9995	2.6579
p4s/15	1111	1310	1379.75	8.94	0.9990	3.1169
p3s/7	111	1280	1034.31	7.97	0.9989	2.9151
p2s/3	11	1280	911.10	4.14	0.9991	2.9040
p1s/1	1	1220	747.79	1.97	0.9995	2.4909
n4g/7	0111	1250	2356.67	5.02	0.9993	2.7862
n3g/3	011	1220	2125.56	3.43	0.9995	2.6451
n2g/1	01	1220	2095.90	1.25	0.9995	2.6338
n1g1/0	0	1190	1449.56	0.12	0.9998	2.4854
p1g1/1	1	1190	745.89	0.14	0.9998	2.4903

4.3.2 Effect of Technology Scaling, Number of Dimensions and Training samples on Model Runtime

Since we have developed our SVM models using $28nm$ bulk-CMOS technology. BSIM models are used to relate process and environmental parameters to leakage current. Our proposed methodology in this step is technology dependent because the fitting parameters are extracted using regression data generated in that technology using SPICE simulations. However the model development process in Figure 4.1 and 4.2 is technology independent and can be used in any technology node with any underlying device model i.e. BSIM, PSP etc. In lower technology nodes than $28nm$, non-linearity related to leakage current in PVTW space may be higher which may require more number of samples than the model in $28nm$ technology node for the same accuracy. If the number of samples are increased then the runtime of the model is also increased, as explained previously. One important advantage of our model is that it does not

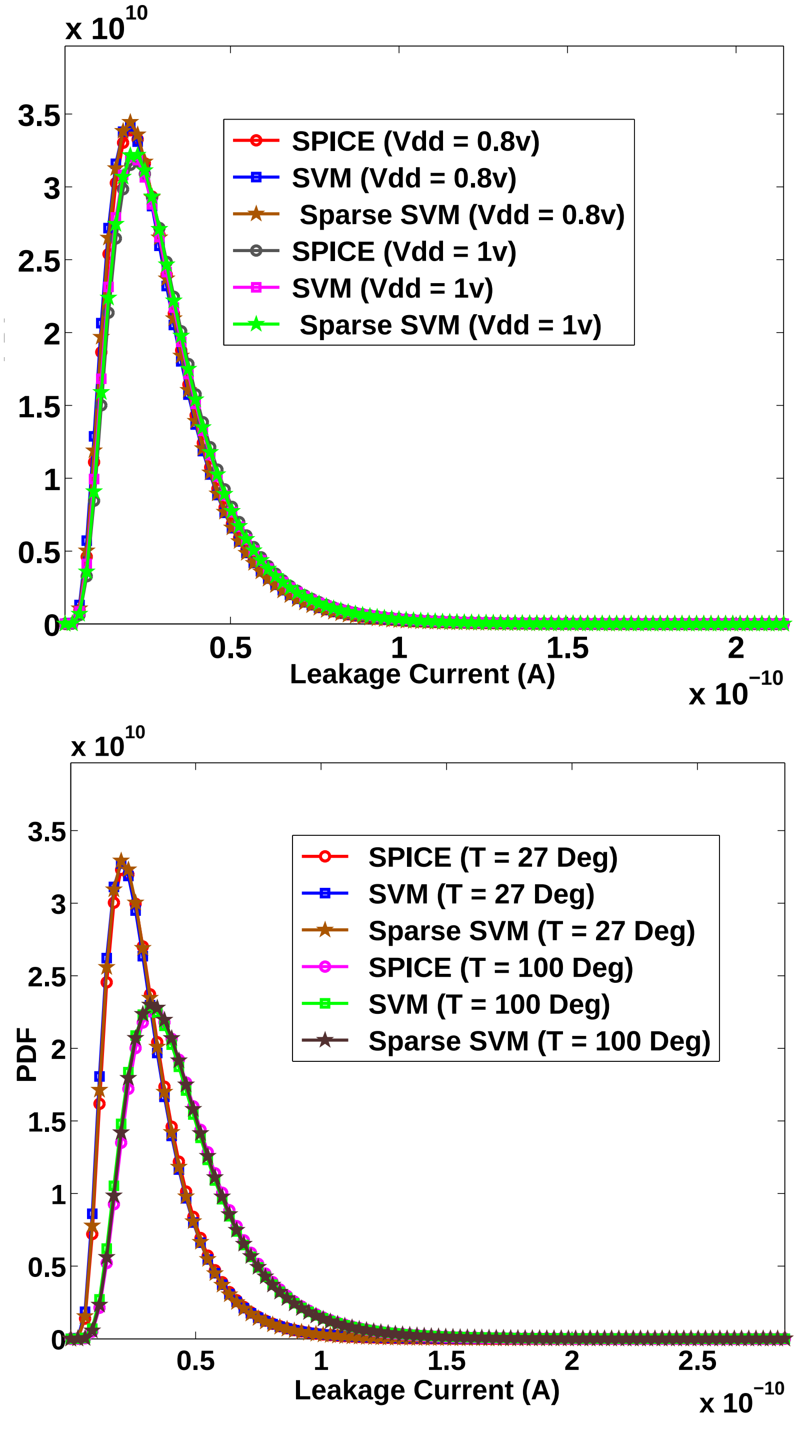


Figure 4.4: Comparison of PDF generated using SPICE and SVM model at different values of V_{dd} (top) and T (bottom) for n4s/0 stack

presume any kind of underlying model like exponential linear or quadratic model which is highly required to increase the accuracy of model. Our model is a kind of dynamic model whose runtime and accuracy depends upon the training data and has the ability to model highly non-linear performance parameters in the complete PVTW space.

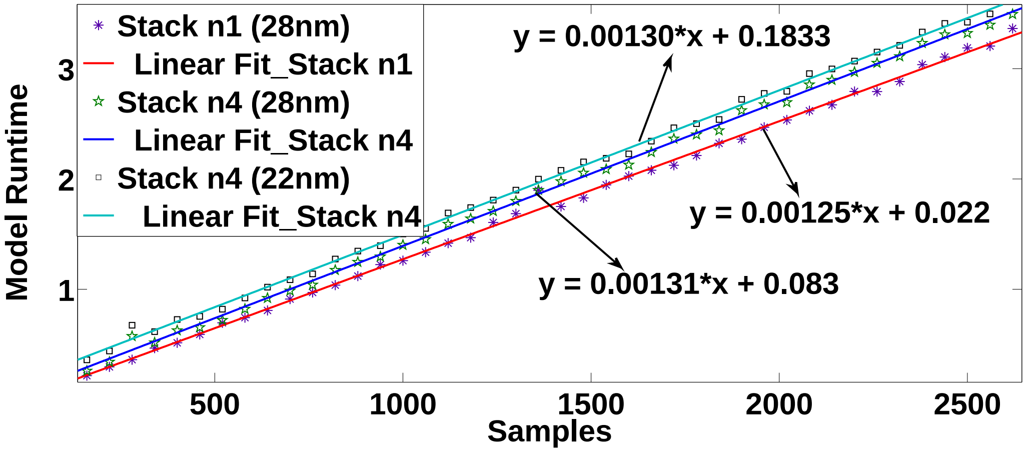


Figure 4.5: Effect of training samples on model runtime

Figure 4.5 shows the variation in the runtime of models for 5000 testing samples by n1s/0 and n4s/0 stack for increasing value of training samples in different technologies. Linear curve fitting is used in MATLAB toolbox for both models. Runtime linearly increases with the increasing number of training samples used to develop the model. In 28nm technology, runtime difference between the n1s/0 and n4s/0 stack models is little bit higher at the higher value of samples. Runtime complexity for both models is $O(n)$, however, the runtime of n4s/0 model is little bit higher than n1s/0 model due to the higher number of fitting parameters. Furthermore, in advanced technologies, complexity is also $O(n)$ but there may be variation in the fitting parameters. This experiment clearly shows the SVM model's runtime is mainly dependent on number of training samples, not on the number of variables and process technology nodes.

Next, we examine the effect of adding width as another dimension to our SVM models over model in [2] on characterization time, runtime and MSE. Similar effect can be elaborated on sparse SVM model. The dependence of I_{sub} and I_{gate} is linear with respect to width of transistors of a gate. Since we are modeling *log* of leakage which further reduces the non-linearity introduced by width into SVM models.

Table 4.7: Effect of excluding width in LS-SVM model (with respect to same MSE)

Stack model	*Training Samples*	*Characterization Time (S)*	*Runtime (S) (10000 samples)*	*MSE* ($\times 10^{-6}$)
n4s/0	1480	1533.29	3.9106	8.43
n3s/0	1420	1468.21	3.7734	7.90
n2s/0	1420	1450.38	3.7721	3.81
n1s/0	1360	1020.72	3.3962	1.70
p4s/15	1480	935.042	4.0642	8.93
p3s/7	1480	701.216	4.0548	7.93
p2s/3	1420	580.121	3.8410	4.10
p1s/1	1420	415.088	3.8271	1.97

Table 4.8: Effect of excluding width in LS-SVM model (with respect to same number of samples)

Stack model	*Training Samples*	*Characterization Time (S)*	*Runtime (S) (10000 samples)*	*MSE* ($\times 10^{-6}$)
n4s/0	1540	1560.12	4.1913	8.18e-6
n3s/0	1540	1509.28	4.1876	7.29e-6
n2s/0	1540	1495.80	4.1822	3.20e-6
n1s/0	1540	1148.71	4.1790	1.52e-6
p4s/15	1540	963.041	4.2799	8.59e-6
p3s/7	1540	729.110	4.2589	7.43e-6
p2s/3	1540	665.180	4.2489	3.80e-6
p1s/1	1540	477.076	4.2441	1.71e-6

To evaluate this effect, we conducted two set of experiments. **In first case**, we derive the number of samples required to train model for same MSE with width consideration. Table 4.7 shows that less samples are required which also reduces the runtime of the model as explained previously. If we do not consider the width in the model, average 30 seconds are saved for each model. Total time saving for basic stack models is 18×30 seconds. If we do not combine parallel transistors, 16 models other than 18 basic stack models are required for parallel transistor stacks to characterize whole standard cell library as shown in Figure 3.10, in which each model consist 3 or 4 transistors. On an average, 1500 seconds for one model and total 16×1500 are required for parallel transistor stacks. Thus, less characterization time is needed if width is added to the model. In a similar way, characterization time is also saved for I_{gate} models. **In Second case**, we keep the number of training samples same when width is considered in the model i.e. 1540 samples. For this case, Table 4.8 shows that the excluding width in the model affects very less in terms of both characterization time and runtime of the model but reduces error up to some extent. Finally it can be concluded that if any parameter that is to be included in the model increases the number of training samples, then both characterization time and runtime of the model get affected to a large extent. In our case, we include width as a design parameter in the models to reduce the number of models to be characterized which ultimately reduces the characterization time. If the number of training samples are same as before and after adding width, then runtime is almost same for both cases. We use sparse LS-SVM to reduce the number of training samples for reducing the simulation time of models.

4.3.3 Comparison with exponential polynomial models

In this section, we compare the accuracy, number of SPICE simulations required and memory requirement to save model parameters for our SVM models with the existing analytical exponential quadratic (EQ) models in [109].

4.3.3.1 Accuracy

In case of EQ models, input - output relation is represented by a presumed form of the equation and expectation of that equation is calculated to find out μ and σ of I_{sub} of a gate in presence of process variations. While our SVM models are black-box kind of models, which do not assume any pre-defined set of equation to establish the relationship between input and output. For SVM models, we do not consider the equations. Hence, μ and σ for a gate are evaluated using efficient sampling techniques. SVM models are slower than EQ models but provide better accuracy and we will show that our SVM model outperforms EQ model in presence of process variations. In EQ model of [109], I_{sub} of a gate in terms of process parameters L and V_{th} is represented as follows.

$$I_{leak} = C.e^{A_1 \triangle L + B_1 \triangle L^2 + A_2 \triangle V_{th} + B_2 \triangle V_{th}^2} \tag{4.17}$$

For a fair comparison with our model, we add T_{ox} as another dimension in Equation (4.17). We use same 1540 optimal set of samples to develop EQ model as used in Table 4.5 to characterize our SVM models. MATLAB lsqcurvfit tool is used to fit the data in the form of Equation (4.17) and constant terms are determined. To test both models, 10000 disjoint set of samples are generated in the process variation space. I_{sub} of n1s/0 and n4s/0 stacks are modeled using EQ model for varying value of standard deviation of process parameters from 5% to 20% with the step size of 2.5% while V_{dd}, T and W of all transistors on a stack are kept at $1V$, $25°C$ and $28nm$. Three variables are required to include global process parameters in the 'P' transistor stack model, each corresponding to L, V_{th} and T_{ox}. Three variables are required to include spatial component and 3*P variables are required to account for the random component of local process variations. Thus, n1s/0 and n4s/0 model is 9 and 18 dimension model respectively. MSE on testing data is calculated using both models as shown in Table 4.9.

Results show that the rate of increase of MSE for EQ model for n1s/0 stack model is very large than SVM model as the percentage variation in process parameters increase. In the next experiment, we keep σ at 20% and T is varied from $25°C$ to $100°C$. MSE for n1s/0 model is increased to 5.4921 and 3.81*10^{-6} at $100°C$ for EQ and SVM model respectively. We have also compared our method for higher dimensional n4s/0

Table 4.9: Mean square error using SVM and EQ model in process variation space

Standard	*MSE (n1s/0 stack)*		*MSE (n4s/0 stack)*	
Deviation (σ)	*EQ Model*	*SVM Model*	*EQ Model*	*SVM Model*
5%	0.0166	1.35e-6	1.1032	7.68e-6
7.5%	0.0726	1.49e-6	1.3566	8.01e-6
10%	0.4501	1.71e-6	1.5231	8.44e-6
12.5%	1.1026	2.03e-6	1.9406	9.13e-6
15%	2.8803	2.78e-6	3.1487	9.79e-6
17.5%	3.0100	3.30e-6	4.5691	1.31e-5
20%	4.0014	3.45e-6	6.3478	2.71e-5

stack model with EQ model. Accuracy of EQ model is further reduced if stacks with more transistors are used. Similarly, I_{gate} was also modeled in the form of EQ model and found to be highly inaccurate as similar to I_{sub} EQ model.

To improve the accuracy of the EQ model for larger variations and higher order stacks, higher order terms i.e., L^3, V_{th}^3, T_{ox}^3 can be used. However beyond quadratic terms, the model has a limitation in calculating the expectation of cubic or higher order terms. This expectation may not converge because cubic or higher terms will grow faster than decaying of quadratic terms for a particular value of process parameters. However, in the case of SVM model, non-linearity of the model is tackled by increasing the number of training samples, kernel function and its tuning parameters. Thus, SVM models are more suitable for modeling leakage for larger range of process variations and larger number of parameters.

4.3.3.2 SPICE simulation and Memory requirement

Consider a four transistor NMOS stack with 3 inter-die parameters (L, V_{th}, T_{ox}), 3*4 = 12 process parameters to account intra-die random variations and 3 process parameters to consider intra-die spatial variations. Leakage of this stack is to be modeled in EQ form as shown in Equation (4.17). EQ model will have a total 18 first order terms and 171 second order terms. To develop this model with 189 coefficients at least 189 SPICE simulations are required. We actually may require some extra samples i.e. 225 SPICE simulations may be needed to model the leakage accurately along with 500 disjoint SPICE simulations to check accuracy of the developed model. Generally, the state-of-art leakage models do not include large range of V_{dd} and T in a single model. Thus V_{dd} and T range is divided into multiple regions and model is developed for each region. Now assume V_{dd} is divided in steps of $100mV$ and T in steps of $25°C$, then we need 24 different EQ models in a supply voltage range of 0.6 - $1.2V$ and temperature range of 0 - $100°C$. This requires 24*225 = 5400 SPICE simulations to model leakage of a four transistor NMOS stack in complete PVTW space. Our SVM model required a maximum of 1540 SPICE simulations to cover complete PVTW space in a single model, which is $\sim 3.5\times$ less than EQ model. Sparse model further reduces the number of training samples, results in saving of time consumed in SPICE simulation. Our model is able to predict leakage at any V_{dd} and T, making it voltage and temperature scalable. The important application of voltage and temperature scalable models can be the use of these models for leakage estimation during adaptive voltage scaling (AVS) and full-chip thermal analysis. From memory point of view, each out of 24 EQ models has 189 coefficient. Some memory is needed to save these coefficients. let us assume each coefficient requires one unit memory, thus total 189*24 = 4536 unit memory is required for the coefficients of all models. In our SVM models, training procedure generates the model coefficients as 'α' and 'b' value. Number of 'α' values will be same as the number of training samples and one 'b' value is generated for each model. Thus, total 1540+1 = 1541 unit memory is required which is $\sim 2.94\times$ less than EQ model. Finally it can be concluded that our SVM model outperforms analytical equation based EQ models not only in terms of accuracy but also SPICE simulation requirement to develop model and memory space requirement to save model coefficients. This process is to be repeated for 176 input vector combinations of 20 CMOS gate standard cell library for the case of EQ models, while our methodology requires this model generation procedure only for 30 times (I_{sub} models). Thus, our methodology results in $\sim 21\times$ less characterization time and $\sim 17\times$ less memory requirement. It can be concluded that proposed SVM models require less memory and less characterization time to develop leakage models in the considered PVTW space.

4.3.4 Comparison with the methods based on scaling the single transistor leakage

I_{sub} and I_{gate} models presented in [6] and [7] are based on scaling the leakage of single transistor. Scaling factors are calculated depending on the stack type (NMOS or PMOS), input applied to the stack and number of transistors in that stack. However, process, V_{dd} and T variations are not considered while developing these models. In our experiments, we show that the models in [6] and [7] are not valid for all CMOS gates with their input vectors considering variations. These models are based on two assumptions: 1) Only one scaling factors to scale the leakage of single transistor, 2) Same scaling factor for different input vectors of a stack. To evaluate the inaccuracy and efforts made to develop the models presented in [6] and [7], we performed an experiment in which I_{sub} of four transistor NMOS stack is calculated by scaling the I_{sub} current of single NMOS 'OFF' transistor at nominal values of varying parameters. Scaling factor can be calculated as in Equation (4.18).

$$Scaling\_factor(s.f.) = \frac{I_{leak,n4\_1110}}{I_{leak,n1\_0}} \tag{4.18}$$

This scaling factor evaluates to approx 0.998. In our first experiment, we assume 10% variation in global and local process parameters in L, V_{th} and V_{th}. Figure 4.6 shows that scaling factor can vary from 0.1 to 25. In our second experiment, we kept the process parameters at their nominal values, T and V_{dd} is varied in between $0°C$ to

$100°C$ and $0.9V$ to $1.2V$. Figure 4.6 indicates that scaling factor is varied from 0.1 to 2. These experiments establish that calculation of scaling factors neither provides accurate result in the presence of process variations nor present good scalable models at desired T and V_{dd}.

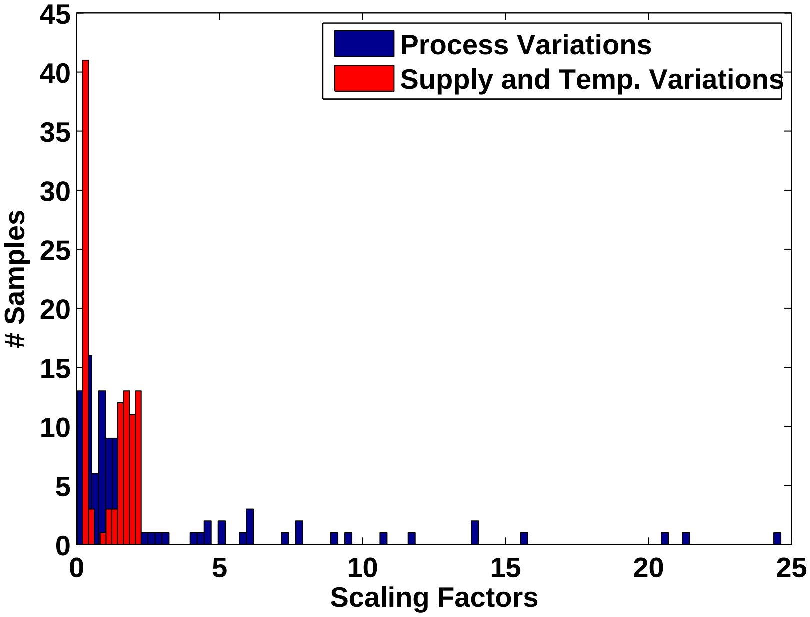


Figure 4.6: Scaling factors required for accurate I_{sub} estimation of NAND4 gate with input '1110' using single NMOS 'OFF' transistor with Process, V_{dd} and T Variations

Now, consider a NMOS stack of 3-input NAND gate with three input vectors '011', '101' and '110'. I_{sub} for these input vectors is assumed to be same as single transistor I_{sub} leakage. Table 4.10.(a) shows the error in I_{sub} for these input vectors. Column 2 and 3 in Table 4.10(a) denotes the representation of I_{sub} models for considered input vectors. I_{sub} for the considered input vectors can not be calculated from n1s/0 model due to the very large difference between actual and model output. $I_{sub}$for all three input vectors is also not same. Thus, all three input vectors require different models to estimate I_{sub}. In our approach, we use separate models for all three input vectors. Higher error for these input vectors can also be confirmed from our analysis in Figure 3.2. For some of the samples in process variation space, I_{sub} for these input vectors lies in the high error region of Figure 3.2.

Table 4.10: Error for modeling (a) I_{sub} and (b) I_{gate} of NMOS stack of NAND3 gate using methodology in [6] and [7] in $28nm$ technology ($V_{dd} = 1V$, $T = 27°C$)

I_{sub}					
Gate	*Input*	*Stack Type*	*Model*	*SPICE(nA)*	*[6] [7](nA)*
				μ/σ	μ/σ
NAND3	011	n3s/3	n1s/0	2.61/12.73	20.06/176.1
NAND3	101	n3s/5	n1s/0	3.45/18.40	20.06/176.1
NAND3	110	n3s/6	n1s/0	16.99/103.7	20.06/176.1

(a)

I_{gate} *(for NMOS stack only)*					
Gate	*Input*	*Stack Type*	*Model*	*SPICE(pA)*	*[6] [7](pA)*
				μ/σ	μ/σ
NAND3	001	n3g/1	n1g1/0	58.48/48.32	163.34/132.71
NAND3	010	n3g/2	n1g1/0	309.43/187.61	164.57/133.70
NAND3	011	n3g/3	n1g1/0	34.31/30.18	163.34/132.71

(b)

Next, consider I_{gate} of NMOS stack for 3-input NAND gate with input vectors '001', '010' and '011'. I_{gate} for these input vectors is calculated from n1g1/0 model shown in Table 4.10(b). I_{gate} for '001' and '011' is derived by scaling n1g1/1 model's leakage through same scaling factor, but results in Table 4.10.(b) show that the μ and σ estimated through SPICE is completely different for both input vectors, thus same scaling factor can not be used for both inputs. These input vectors require different I_{gate} models for each input vector. While I_{gate} for '010' input is considered as same as I_{gate} of n1g1/0 model neglecting the I_{gate} of 'ON' transistor but our results in Table 4.10.(b) indicate that the differences in μ and σ estimated through SPICE and the model are very large. Under consideration of process variation, scaling factor itself is a non-linear function which require more efforts to characterize the leakage of all CMOS gates. Another disadvantage is that the CMOS gates with parallel transistors or stacks of parallel transistors i.e. AOI22, AOI32, OAI22, OAI32 are not considered. As the size of the standard cells increases, this method will give more error than the error for basic CMOS gates. Based on our experiments, it can be concluded that the I_{sub} and I_{gate} of single transistor leakage with single scaling factors considered for different gates in standard cell library give high errors in leakage modeling. Similar to the EQ models, our SVM model outperforms these kind of models in terms of SPICE simulation and memory requirement.

4.3.5 Comparison with Artificial Neural Network models

Methodology proposed by authors in [2] reduces the error in I_{sub} estimation by developing extra models. This method does not remove the 'ON' transistors which are directly connected to the output node. For example, consider NAND3 gate with '011', '101' and '110' input vectors as shown in Table 4.10.(a). Middle, leftmost and all 'ON' transistors are removed from '011', '101' and '110' inputs respectively. But still this is a simplified assumption and may give large errors in leakage modeling. Since, I_{gate} was not modeled in [2], analysis in Tables 4.11 and 4.12 for ANN model is based on I_{sub} only. While, our approach is evaluated based on both I_{sub} and I_{gate} leakage. For the purpose, we have implemented the ANN based surrogate models based on the stacks presented in [2] with 20 neurons in hidden layer for each stack model. Average 30 sec. are required for training of the ANN model for each stack, which is very less than our method. However, it is for fixed ANN model parameters. Active learning will also increase the training time in the case of ANN model. Error in μ and σ for various stacks is given in Table 4.11. For NAND3 gate with inputs '011' and '101', error in I_{sub} estimation is reduced with the use of methodology in [2] compared to [6] and [7], but assumption made for these input vectors still results in high error and thus, require a new model for each input vector. Maximum average error in μ and σ is 58.6% and around 203.7% respectively, if the leakage current of p4s/1 stack is evaluated using p1s/1 stack which clearly indicates that this stack requires a new stack model i.e. the drop across 'ON' transistors can not be ignored.

Table 4.11: Error for modeling stacks [2] in $28nm$ technology ($V_{dd} = 1V$, $T = 27°C$)

Gate	*Input*	*Stack Type*	*Model*	*Scaling factors*	*SPICE(nA)*	*[2](nA)*	*% Error*
				$I_{Spice}/\ I_{Model}$	μ/σ	μ/σ	μ/σ
NAND4	1110	n4s/12	n1s/0	3.5873/3.5893	15.20/89.48	20.05/176.0	31.90/96.7
NAND3	110	n3s/6	n1s/0	3.5846/3.5893	16.99/103.7	20.06/176.1	18.06/69.8
NAND3	011	n3s/3	n2s/1	1.1637/1.3294	2.61/12.73	2.99/16.81	14.56/32.05
NAND3	101	n3s/5	n2s/1	1.3291/1.3294	3.45/18.40	3.55/20.09	2.90/9.18
NAND2	10	n2s/2	n1s/0	3.5820/3.5893	18.12/127.5	20.07/176.2	10.76/38.2
NOR4	0001	p4s/1	p1s/1	2.3873/2.4011	15.04/71.70	23.86/217.8	58.6/203.7
NOR3	001	p3s/1	p1s/1	2.3919/2.4011	16.37/84.14	23.90/218.2	45.9/159.3
NOR2	01	p2s/1	p1s/1	2.3965/2.4011	20.43/136.3	23.33/218.5	14.2/60.3

4.3.6 Leakage calculation of a gate using stack models

Leakage of gates for a given input vector can be estimated by breaking down the gates into characterized stack models. Thus, leakage estimation of a gate can be mapped to leakage of stack models and can be represented as a linear weighted sum of leakage of the stacks which are to be used for the given gate with particular input vector. Suppose, if 'M' and 'N' number of I_{sub} and I_{gate} stacks are characterized. $I_{sub-stack,j}$ and $I_{gate-stack,j}$ denotes the I_{sub} and I_{gate} through j^{th} and k^{th} stack. Total leakage of a gate 'i' for given input vector 'v' can be given as follows.

$$I^{v}_{leak,i} = \sum_{j=1}^{M} I_{sub-stack,j} + \sum_{k=1}^{N} I_{gate-stack,j} \tag{4.19}$$

We have calculated the total leakage of each gate in considered standard cell library by applying all input vector combinations. Table 4.12 and 4.13 compares our method's accuracy and runtime with HSPICE simulation and ANN model [2]. The average error in μ and σ for total leakage is 0.447% and 0.812% using our SVM model respectively across 176 input vector combinations for 20 CMOS gates compared to the ANN model's 2.6975% and 9.855% for I_{sub} only. Our SVM model's average runtime for calculating μ and σ is 22.1660 sec. compared to 221.55 sec. using SPICE. SVM and

Sparse SVM model is on an average $\sim 10\times$ and $\sim 17\times$ faster than SPICE respectively. However, the maximum runtime improvement is observed for D-flip flop because it only requires the single transistor models to predict the leakage power across all input vector combinations. Sparse SVM further reduces the runtime of the proposed method with negligible increase in error. Last three rows in Table 4.12 indicates that total 34 I_{sub} stack models need to be characterized by ANN model to calculate the leakage current of 20 gate standard cell library across 176 input vector combinations i.e. 4 stack models more than our approach. We also accurately model I_{gate} with only 26 models for all possible input vectors of the considered standard cell library.

Finally, it can be concluded that the error in estimating leakage current is very high using the methodology based on stacks given in [2], which in completely outperformed by our proposed method in terms of accuracy. However, the I_{sub} models developed using ANN are found to be faster than our approach but runtime for ANN model is evaluated only for 20 hidden neuron network model. To improve the accuracy of the ANN model, number of hidden neurons may be increased at the cost of increased runtime of ANN model. One biggest advantage of our model using approach is that we can estimate μ and σ for other gates containing parallel transistor stacks without any pre-characterization, whereas ANN model [2] requires new models to be characterized.

4.3.7 Efficient selection of Monte Carlo samples for Statistical leakage current estimation of benchmark circuits.

Yield of any analog or digital circuit has to be maximized considering variations in process parameters. This step requires sufficient samples to be simulated in process variation space for accurate Probability Distribution Function (PDF) estimation of performance parameters. Authors in [110] proposed a 2-level Design of Experiments (DoE) method to select samples for frequency optimization of Voltage Controlled

Table 4.12: Comparing method in [2] and proposed methodology for error - 28nm technology ($V_{dd} = 1V$, $T = 27°C$, 10000 Monte Carlo samples) (M1→ SVM, M2→ Sparse SVM)

Gate	*ANN Model [2]* *		*Proposed*			
	$\triangle\mu$ %	$\triangle\sigma$%	*M1*		*M2*	
			$\triangle\mu$%	$\triangle\sigma$%	$\triangle\mu$%	$\triangle\sigma$%
NAND4	2.092	11.721	0.287	1.768	0.289	1.769
NAND3	2.249	11.330	0.323	1.821	0.324	1.823
NAND2	2.352	6.911	0.177	0.299	0.179	0.301
NOR4	5.377	24.991	0.638	1.794	0.640	1.796
NOR3	7.098	24.001	0.499	1.954	0.499	1.957
NOR2	4.499	23.886	0.046	0.301	0.047	0.303
INV	0.017	0.016	0.015	0.016	0.017	0.017
BUFFER	0.032	0.036	0.031	0.035	0.032	0.037
AND4	0.292	0.296	0.041	0.071	0.043	0.072
AND3	0.682	1.061	0.051	0.075	0.053	0.078
AND2	2.919	17.290	0.053	0.089	0.055	0.091
OR4	2.459	6.449	0.080	0.164	0.082	0.166
OR3	3.676	4.201	0.048	0.089	0.049	0.091
OR2	7.762	24.066	0.036	0.079	0.037	0.080
MUX2×1	0.199	0.349	0.197	0.350	0.199	0.353
XOR2	2.338	8.699	0.485	1.158	0.489	1.161
MAJ_GATE	4.257	10.923	0.430	0.150	0.434	0.153
AOI22	2.682	9.901	2.635	2.692	2.638	2.695
OAI22	2.824	10.772	2.726	3.130	2.729	3.133
DFF	0.144	0.202	0.140	0.201	0.143	0.203
Avg =	2.6975	9.855	0.447	0.812	0.449	0.814
Proposed	**Basic I_{sub} Models = 30, New I_{sub} Models = 0,**					
	I_{gate} Models = 26					
[2]	**Basic I_{sub} Models = 18, New I_{sub} Models = 16**					

*Error for ANN model is evaluated based on I_{sub} only.

Oscillator (VCO) circuit. It generates 2^n samples, where n is number of variable parameters. The work in [110] optimizes the analog circuit considering only global variations in 5 parameters, which only require $2^5 = 32$ samples to be simulated. But simulations of lower number of samples result in large error in μ and σ estimation of performance parameters. In addition, local variations increases the dimensionality of process variation space, thus simulation time increases exponentially. In digital cir-

Table 4.13: Comparing method in [2] and proposed methodology for runtime - 28nm technology ($V_{dd} = 1V$, $T = 27°C$, 10000 Monte Carlo samples) (M1→ SVM, M2→ Sparse SVM)

Gate	*Avg. Runtime (S)*				*Speed up (×)*		
	[SPICE]	*[2]* *	*Proposed*		*[2]*	*Proposed*	
			M1	*M2*		*M1*	*M2*
NAND4	221.30	3.5478	13.7979	8.7775	62	16	25
NAND3	195.02	3.5403	13.5604	8.7650	55	14	22
NAND2	171.20	3.5370	13.2947	8.3706	48	13	20
NOR4	217.10	3.5487	13.9073	8.8992	61	16	24
NOR3	191.92	3.5130	13.6485	8.8676	54	14	21
NOR2	172.04	3.5088	13.3357	8.4776	49	13	20
INV	164.50	3.4281	12.3076	7.6401	47	13	21
BUFFER	173.69	7.0077	24.5952	15.2802	24	7	11
AND4	226.20	7.0260	26.1186	16.2525	32	8	14
AND3	203.20	7.0201	25.6075	15.8771	28	8	13
AND2	176.45	7.0179	25.1001	15.7255	25	7	11
OR4	223.96	7.0275	26.1640	16.4803	31	8	13
OR3	198.69	7.0203	25.6180	15.9758	28	8	12
OR2	175.28	7.0176	25.1791	15.7375	24	7	11
MUX2×1	232.24	6.9453	28.6352	17.6180	33	8	13
XOR2	242.91	10.648	35.7112	18.5063	22	7	13
MAJ_GATE	237.80	8.7588	37.2854	19.3517	27	6	12
AOI22	233.21	7.2420	21.4694	13.7053	32	11	17
OAI22	234.42	7.3899	21.5263	13.7551	31	11	17
DFF	539.83	10.211	26.4578	16.7287	52	20	32
Avg =	221.55	6.2478	22.1660	13.5396	35	10	17

*Runtime for ANN model is evaluated based on I_{sub} only.

cuits, due to higher number of transistors on a single IC, larger set of local parameters are need to be considered. DOE is completely infeasible to analyze the performance of digital circuits. Thus, efficient MC samples are need to be developed for efficient analysis of digital circuits. Now we explain our proposed efficient sample selection methodology, which can fully utilize our reduced set of stack models.

A process parameter P_{ijk} having the value P_{nom} for the j^{th} transistor in the i^{th} gate of k^{th} grid can be given as follows.

$$P_{ij} = P_{nom} + \triangle P^{inter} + \triangle P_{ij}^{intra,rand} + \triangle P_{k}^{intra,spa} \tag{4.20}$$

Here, P_{nom} is the nominal value of process parameter, $\triangle P^{inter}$ is the zero mean random variable (RV) representing the global component of process variation. $\triangle P_{ij}^{intra,local}$ denotes random local variation of the j^{th} transistor in the i^{th} gate and $\triangle P_{k}^{intra,spa}$ represents the variable related to spatial correlation between same process parameters of different grids. In our work, we are considering the inter- and indra-die process variations in the parameters length (L), threshold voltage (V_{th}) and oxide thickness (T_{ox}). The process parameter of j^{th} transistor in i^{th} gate of grid k can be represented as follows.

$$\begin{aligned} L_{ijK} &= L_{nom} + \triangle L^{inter} + \triangle L_{ij}^{intra,rand} + \triangle L_{k}^{intra,spa} \\ V_{th_{ijK}} &= V_{th_{nom}} + \triangle V_{th}^{inter} + \triangle V_{th_{ij}}^{intra,rand} + \triangle V_{th_k}^{intra,spa} \\ T_{ox_{ijK}} &= T_{ox_{nom}} + \triangle T_{ox}^{inter} + \triangle T_{ox_{ij}}^{intra,rand} + \triangle T_{ox_k}^{intra,spa} \end{aligned} \tag{4.21}$$

For full-chip leakage estimation, we can use equations used in SVM models but it will be very difficult to obtain the mean and standard deviation in analytical form as shown in [2]. Thus, we select sampling based methodology for full-chip leakage estimation. Earlier work based on sampling methods for full-chip leakage estimation use SPICE simulation. One important advantage of proposed SVM models is that these models can replace SPICE simulation loop in sampling based full-chip leakage estimation methods. To further reduce runtime, efficient sample selection based methodologies are required, which can reuse the previously simulated samples and modifying them according to the process parameters [47] [111]. In the proposed framework, our main aim is to use SVM model simulation to the minimum extent possible.

The important advantage of reducing the number of models in our proposed work is to lower down the runtime complexity due to the use of reduced number of leakage components to be added for full chip leakage estimation. For example, method in [21] uses the dominant I_{sub} and I_{gate} concept among various input vector combinations of CMOS gates, which only adds the leakage of only dominant components. Due to large variation space, this assumption may provide large error in leakage estimation. Equivalence concept between different input vectors of CMOS gates is also used. For example, I_{sub} of NAND4 gate with '1110' input, NAND3 with '110', NAND2 with '1110' is assumed to be same as INV with '0' input. This assumption also provide inaccurate results in leakage estimation. Thus, reduction in number of characterization models is very important for characterization time and runtime complexity reduction. We exploit the equivalence concept between our stack models. For example, n3s/0 stack model will be used for I_{sub} estimation of NAND4 gate with '1000' input and NAND3 gate with '000' input.

Further advantage of our methodology lies in the selection procedure of Monte Carlo samples of the individual gate. If we select the same values of process parameters for different gates, then we only need to run the model once per gate per input vector. For example, Monte Carlo simulation for two CMOS gates with the same input can be performed with respect to parameter say, Length (L), then the Monte Carlo samples for one gate can be used for another gate i.e. the same leakage power values can be used for both gates. The only thing is to combine different values for joint PDF calculation. The total leakage under process variation can be determined by randomly combining the samples i.e. combining the leakage of two gates randomly. Figure 4.7 shows two inverters in two different grids, whose leakage is a function of transistor length (L). Four different values are selected in the variation range of length. Joint PDF can be generated by randomly combining these four values from two different gates, which gives total 16 samples in the process variation space. Joint leakage can be find out by randomly combining the leakage of both inverters instead of simulating two inverters for 16 samples, which requires only 4 simulations to generate 16 samples.

The efficiency of our method comes from saving the simulation time of CMOS gates in the circuit.

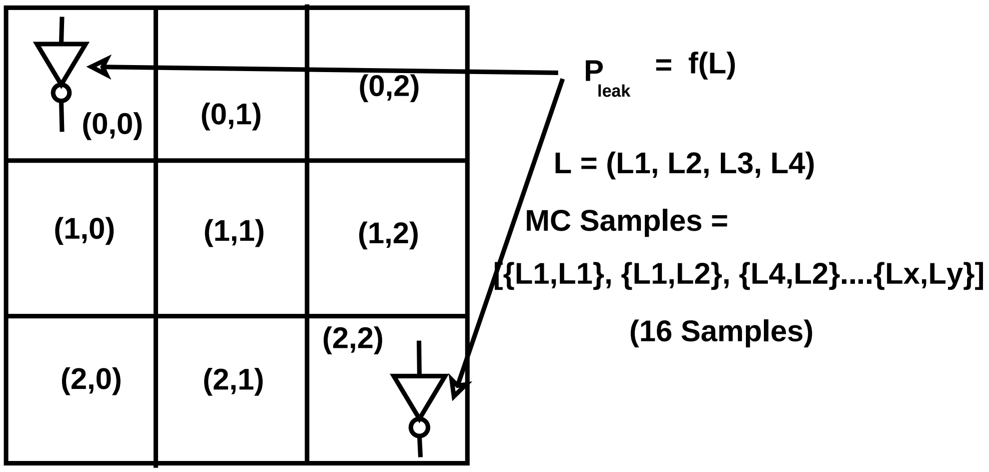


Figure 4.7: Illustrating efficient sample selection methodology

To include spatial component of intra-die variation, correlation matrix is generated by partitioning the overall chip into k grids as shown in Figure 4.7. k variables are required to define the correlation matrix between k grids. Correlation matrix can be given as in [112]. We describe our approach for sample selection in following formal steps.

Step (1) - Apply input vector to the circuit. Get the type of gates and corresponding input vectors in a grid. A gate_type (GT_l) represent many gates of CMOS circuit depending on the type of gates in the standard cell library and input vector applied. For each GT_l, only one SVM model simulation is required. So, the total number of gate_type can be the number of precharacterized models. This process is applied for all grids.

Step (2) - Generate samples to consider spatial component of intra-die variation $P_K^{Intra,spa}$ from k- dimensional joint PDF of k normally distributed random variables. Each sample is a k- dimensional vector and generate separately for each variable. For example: k- dimensional vector for variable length (L) can be described as - $\triangle L_k^{intra,spa}$ =($\triangle L_1^{intra,spa}$, $\triangle L_2^{intra,spa}$, ... $\triangle$ $L_m^{intra,spa}$). Similarly samples for V_{th} and

T_{ox} can be generated.

Step (3) - Generate Monte Carlo samples from the normal distribution of intra-die process parameters of every gate type (GT_l) to account for random component of intra-die variation. The corresponding sample for j^{th} transistor in i^{th} can be represented as $P_{ij}^{intra,rand}= (\triangle L_{ij}^{intra,rand}, \triangle V_{th_{ij}}^{intra,rand}, \triangle T_{ox_{ij}}^{intra,rand})$

Step (4) - Add inter-die process parameters to all samples generated in step (3). This variable independently generates the sample P^{inter} for each sample $P_{ij}^{intra,rand}$. The referred sample can be represented now as- $P^{inter}+P_{ij}^{intra,rand}= (\triangle L^{inter}+\triangle L_{ij}^{intra,rand}, \triangle V_{th}^{inter} + \triangle V_{th_{ij}}^{intra,rand}, \triangle T_{ox}^{inter} + \triangle T_{ox_{ij}}^{intra,rand})$.

Step (5) - Add samples to account for spatial component $P_k^{Intra,spa}$, generated in step (2). Now the sample becomes as- $P^{inter}+P_{ij}^{intra,rand}+P_k^{Intra,spa}= (\triangle L^{inter} + \triangle L_{ij}^{intra,rand}+\triangle L_k^{intra,spa}, \triangle V_{th}^{inter}+\triangle V_{th_{ij}}^{intra,rand}+\triangle V_{th_k}^{intra,spa}, \triangle T_{ox}^{inter}+\triangle T_{ox_{ij}}^{intra,rand}+ \triangle T_{ox_k}^{intra,spa})$.

Step (6) - Add the nominal values of process parameters. The overall Monte Carlo sample for a gate can be represented as- $P_{nom}+P^{inter}+P_{ij}^{intra,rand}+P_k^{Intra,spa}= (L_{nom}+ \triangle L^{inter} + \triangle L_{ij}^{intra,rand} + \triangle L_{ij}^{intra,spa}, V_{th_{nom}} + \triangle V_{th}^{inter} + \triangle V_{th_{ij}}^{intra,rand} + \triangle V_{th_{ij}}^{intra,spa}, T_{ox_{nom}} + \triangle T_{ox}^{inter} + \triangle T_{ox_{ij}}^{intra,rand} + \triangle T_{ox_{ij}}^{intra,spa})$.

Step (7) - Repeat steps (2) to (6) for each grid.

Step (8) - Get the values of leakage current of a gate corresponding to its input vector in each grid k. It should be noted that a gate with same input vector from different grids have different leakage values due to spatial component of intra-die variation parameters which does not allow the use of same set of parameters between different grids.

Step (9) - Replicate each gate_type GT_l, k_m times depending on the type of the gate and input vector applied. This gives the leakage samples of all gates in a grid i.e. N $= k_1 + k_2 + k_3 +$ .... $+ k_m$, Here N $=$ Total gates in a grid. This process is repeated for each grid.

Step (10) - Randomly combine the values of leakage power of individual gate from all the grids to get the final value of leakage power of whole circuit.

We have applied our methodology on ISCAS'85 benchmark circuits. The leakage current of a circuit is input pattern dependent. However, it is impractical to calculate leakage for all input patters of a circuit due to the large number of patterns. In [21], leakage of a circuit is calculated using the average leakage current of a gate and probability of input vectors of each gate in the circuit. Generally, in some applications, input vector applied to the circuit is known a priori. For example, in minimum leakage input vector method for leakage optimization, a specific input vector is applied to reduce leakage through the circuit in sleep mode. However, in other scenarios, input vectors are not known at a given time. Also under process variations, it is very difficult to consider the leakage at particular time instance. Instead, some time-average leakage based on the signal probability is much useful. This is true at the system level leakage and power estimation. Thus, average leakage based on input signal probabilities is calculated to define the leakage of a gate. Effect of input signal probabilities can be easily accommodated in Equation (4.19). Let probability of appearing input vector 'v' for a gate 'i' is p_i^v. Equation (4.19) can be modified to calculate average leakage of a gate as follows.

$$I_{leak,i}^{v} = \sum_{j=1}^{M} p_i^v * I_{sub-stack,j} + \sum_{j=1}^{M} p_i^v * I_{gate-stack,j} \tag{4.22}$$

We have generated 50000 input vector combinations for each benchmark circuit and calculated the probability of input vectors for the gates applied with primary inputs. Input signal probabilities are propagated through each gate to find out input vector probabilities for other gates [113]. Equation (4.22) is used to estimate the average leakage of a gate for each input vector with calculated input vector probabilities. We have used 10000 samples to evaluate the effect of process variations for each input vector. Number of grids for a circuit are selected based on the size of the circuit, as used in [21]. Table 4.14 shows the error in mean and standard deviation and runtime of our approach. Our SVM model's maximum error in μ is 1.75% compared to 55.96% and maximum error in σ is 3.80% compared to 70.40% using stack models described in [2]. Maximum runtime improvement is approximately 221× for C6288 circuit using SVM model, because this circuit only contains the maximum 2-input gates which requires less number of stack models to calculate the leakage power. Sparse SVM increases the maximum runtime efficiency to 323× with negligible increase in the error. Figures 4.8 shows the cumulative density function curve for C6288 benchmark circuit, which indicates that estimation using our approach is very close to the results of SPICE. Our approach predicts the accurate μ and σ values for larger circuit containing the smaller gates. It should also be noted that runtime improvement is dependent on the size of the circuit, type of the gates used in the circuit and the input vector applied at the primary inputs of the circuit.

Table 4.14: Comparing error of proposed method with method in [2] on ISCAS'85 benchmark circuit (V_{dd} = $1V$, T = $27°C$) (M1→SVM, M2→Sparse SVM)

Circuit	*#Grids/ Gates*	μ *SPICE* (μA)	σ *SPICE* (μA)	*[2]*		*Proposed*			
				$\Delta\mu$ *(%)*	$\Delta\sigma$ *(%)*	*M1*		*M2*	
						$\Delta\mu$*%*	$\Delta\sigma$*%*	$\Delta\mu$*%*	$\Delta\sigma$*%*
C17	1/6	0.233	0.1442	6.56	31.77	0.16	0.46	0.18	0.47
C432	4/261	4.301	1.9471	5.81	37.51	0.99	1.91	1.02	1.95
C499	4/771	17.095	6.4491	7.94	47.07	1.08	2.05	1.08	2.10
C880	4/383	3.892	2.0297	15.49	50.41	0.79	1.65	0.80	1.67
C1355	4/562	7.521	3.0112	15.03	48.13	0.87	1.68	0.88	1.69
C1908	16/972	16.573	8.9716	22.89	57.35	1.24	1.99	1.24	2.02
C2670	16/1211	25.992	10.4977	34.77	60.71	1.34	2.15	1.35	2.21
C3540	16/1705	46.820	17.0110	31.28	62.73	1.35	2.18	1.35	2.20
C5315	16/2351	51.0171	20.8079	38.62	65.86	1.59	2.76	1.62	2.79
C6288	64/2416	67.991	26.7333	55.96	67.51	1.14	2.39	1.15	2.41
C7552	64/3624	83.803	31.2141	45.66	70.40	1.75	3.80	1.77	3.84
Average				25.46	54.50	1.12	2.09	1.13	2.12

Table 4.15: Comparing runtime of proposed method with method in [2] on ISCAS'85 benchmark circuit ($V_{dd} = 1V$, T = $27°C$) (M1→SVM, M2→Sparse SVM)

Circuit	*#Grids/ Gates*	*Runtime (Sec.)*				*Speedup (×)*		
		SPICE	*[2]*	*Proposed*		*ANN Model [2]*	*Proposed*	
				M1	*M2*		*M1*	*M2*
C17	1/6	1920	28.70	60.82	44.02	67	31	43
C432	4/261	17520	98.22	197.28	156.58	178	89	112
C499	4/771	34100	197.96	309.12	208.42	172	110	164
C880	4/383	30058	113.28	225.96	181.44	265	133	166
C1355	4/562	33712	141.04	253.78	195.86	239	132	172
C1908	16/972	41120	347.08	501.56	299.60	118	82	137
C2670	16/1211	58558	425.16	738.42	551.86	137	79	106
C3540	16/1705	82202	498.78	953.10	662.82	165	86	124
C5315	16/2351	106396	540.96	1009.3	706.18	197	105	151
C6288	64/2416	130118	358.32	588.64	403.04	363	221	323
C7552	64/3624	190100	747.64	1499.3	1165.7	254	127	163
Average		65982	317.92	576.11	415.96	207	114	159

4.4 Using Characterized Stack Models for Standard Cells in other Logic Styles

In this Section, We show that our characterized stack models are enough to estimate leakage for all input combinations even if the standard cells are implemented with different logic styles. Stack modeling methodology for leakage estimation saves large characterization overhead imposed by developing models for every input vector of each gate.

Figure 4.9 shows the implementation of NAND3 gate in dynamic logic. Next, we only explain that how our characterized models can be used for I_{sub} estimation for different input vector combinations. When CLK is low (precharge phase), output node F will be charged to supply voltage and thus, I_{sub} can be calculated using 'OFF' PDN for different input vector combinations of A, B, C. For these conditions, PDN of dynamic NAND3 gate can be termed as NAND4 gate in Figure 3.1 and we do not need to characterize any new model.

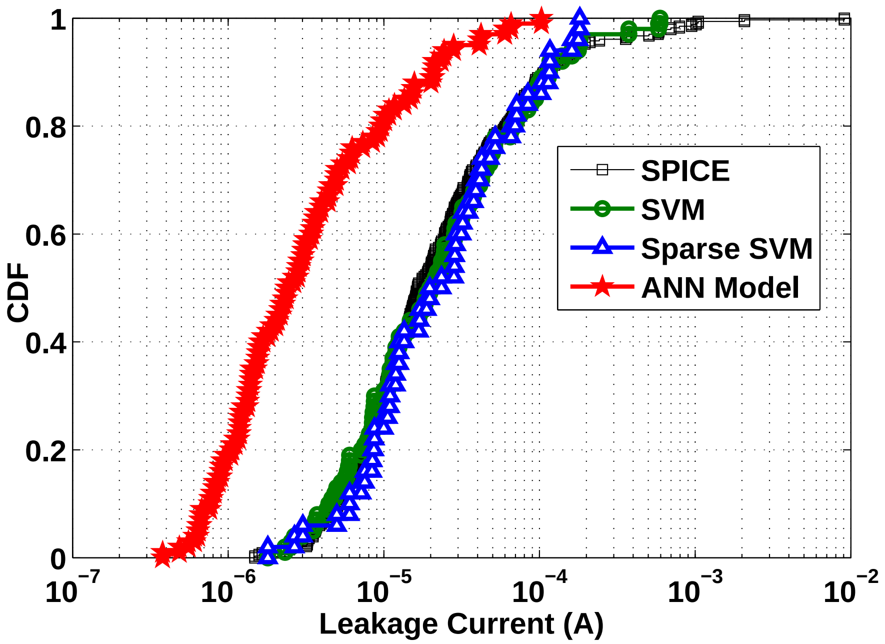


Figure 4.8: Cumulative Density Function (CDF) curve for C6288 Benchmark circuit

When CLK is high (evaluate phase), value at output node depends on value of inputs A, B, C. If all inputs are at logic ‘0’, then out put node F will be at logic ‘1’, making both PDN and PUN ‘OFF’. All these conditions do not change the output node voltage in precharge phase. PUN is ‘OFF’, but can not be used to calculate I_{sub} due to the same drain and source voltage. I_{sub} is to be calculated using PDN, which require same models as in low CLK case. But when CLK is high and ABC = ‘111’, then PDN will be ‘ON’ and output node will be discharged to ground voltage. Stack model p1s/1 can be used to calculate I_{sub} of PUN i.e. single PMOS ‘OFF’ transistor for this condition.

Figure 4.10 shows the complementary pass transistor logic (CPL) implementation of 2-input XOR gate as opposed to CMOS implementation in Figure 3.8.(b). Inverters (INV1 – INV4) are implemented in CMOS style, thus our characterized leakage models are sufficient for leakage estimation of these inverters. Next, we only explain the sub threshold leakage (I_{sub}) estimation of this gate using remaining transistors.

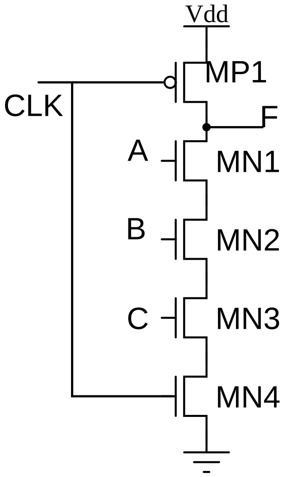


Figure 4.9: NAND3 gate using Dynamic Logic style

When input AB = ‘00’, NMOS transistors MN2 and MN4 are in ON condition, while PMOS transistors MN1 and MN3 are in OFF condition. Node ‘X’ discharges to ground voltage, making PMOS transistor MP2 ‘ON’ and Node ‘Y’ charges to supply voltage, making PMOS transistor MP1 ‘OFF’. I_{sub} of remaining transistors will be calculated using only ‘OFF’ transistors as follows: i) Bbar to X current in MN1 transistor, can be calculated using characterized n1s/0 model ii) current flowing from Y to B in MN3 transistor, can be calculated using characterized n1s/0 model iii) current flowing from supply to ‘X’ in MP1 transistor, can be calculated using characterized p1s/1 model.

For input AB = ‘11’, MN2, MN4 and MP2 ‘OFF’transistors will be used to evaluate I_{sub}. MN2, MN4 transistors only require n1s/0 model and p1s/1 model is used for MP2 transistors.

For input AB = ‘01’, MN1, MN3 transistors will be in ‘OFF’ condition. n1s/0 model is sufficient to estimate I_{sub} for these transistors. While p1s/1 model is only needed for I_{sub} estimation of MP2 transistor. Similarly for input ‘10’, single transistor models are enough to estimate I_{sub}.

Gate tunneling leakage can occur in both ‘ON’ and ‘OFF’ transistors. Node ‘X’ and

‘Y’ are perfect at either supply or ground voltage. Thus, single transistor models as shown in Figure are enough to estimate I_{gate} for all input vector combinations.

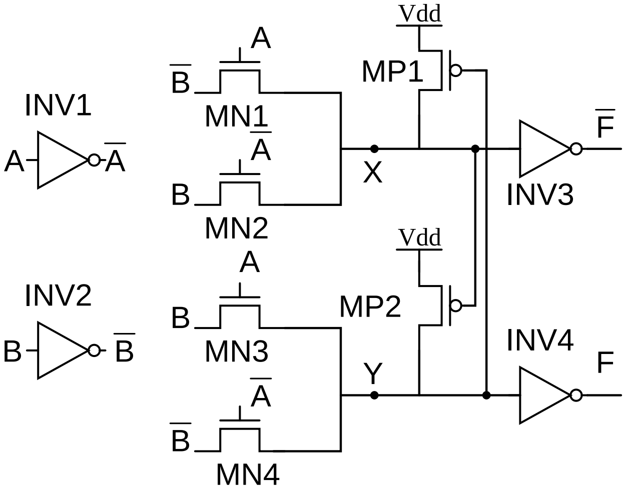


Figure 4.10: 2-input XOR gate in Complementary Pass Transistor Logic (CPL) style

4.5 Stack Models for FinFET Based Standard Cells

In this section, we show that the leakage estimation methodology developed for bulk CMOS technology is also valid for FinFET based technology. New technology such as FinFET has been proposed to reduce the problems related with bulk-MOSFET based technology. Still the leakage is a main concern for these new technologies. It has been shown that I_{gate} contribution in total leakage for FinFET based logic circuits is very less. I_{sub} is a dominant component in FinFET based logic cells [44].

In our methodology, we extract the stack models among basic gates and effective width estimation methodology is developed in case of parallel transistor gates. Consider a NAND2 gate in FinFET based logic style in different configurations as shown in Figure 4.11.

I_{sub} in terms of varying parameters can be describes as [114]:

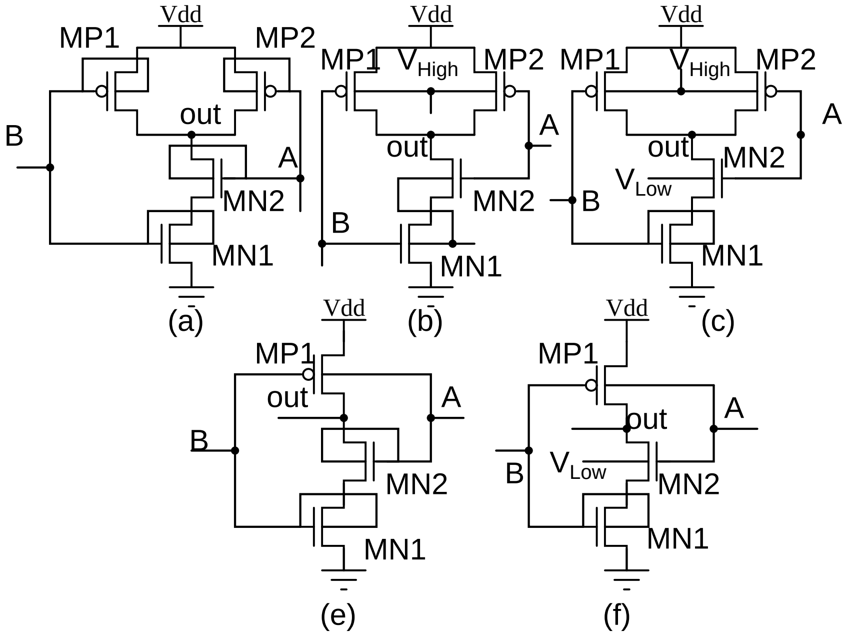


Figure 4.11: NAND2 gate configurations (a) Surrounded Gate (SG) mode (b) Low-Power (LP) mode (c) Mixed-Terminal (MT) mode (d) Independent Gate (IG) mode (e) Variant of IG (IG2) mode

$$I_{sub} = W_{eff}.e^{a_0+a_1L_g+a_2T_{si}+a_3T_{ox}+a_4\phi_{N/P}+a_5\phi_{N/P}^2+a_7V_{dd}} \tag{4.23}$$

Here W_{eff} is the effective width, L_g, T_{si}, T_{ox}, $\phi_{N/P}$, V_{dd} are gate length, Fin thickness, oxide thickness, work function of n/p-FinFET and supply voltage respectively. For Double Gate FinFET transistor, W_{eff} for multiple Fin can be written as follows.

$$W_{eff} = (2.HFIN - DELTAW).NFIN \tag{4.24}$$

Here, $HFIN$, $NFIN$ are height and number of Fins respectively and $DELTAW$ is the variation in the height of Fin due to lithographic variations. Generally, variation in Fin height is assumed to be zero [115] i.e. $DELTAW = 0$. Now, effective width can be represented as follows.

$$W_{eff} = 2.HFIN.NFIN \tag{4.25}$$

Consider two parallel transistors with same terminal voltages and same other parameters except *HFIN* and *NFIN*, effective width for parallel transistors can be given as in and both transistors can be replaced by a single transistor of calculated equivalent effective width.

$$W_{eff,equi} = 2.(HFIN_1.NFIN_1 + HFIN_2.NFIN_2) \tag{4.26}$$

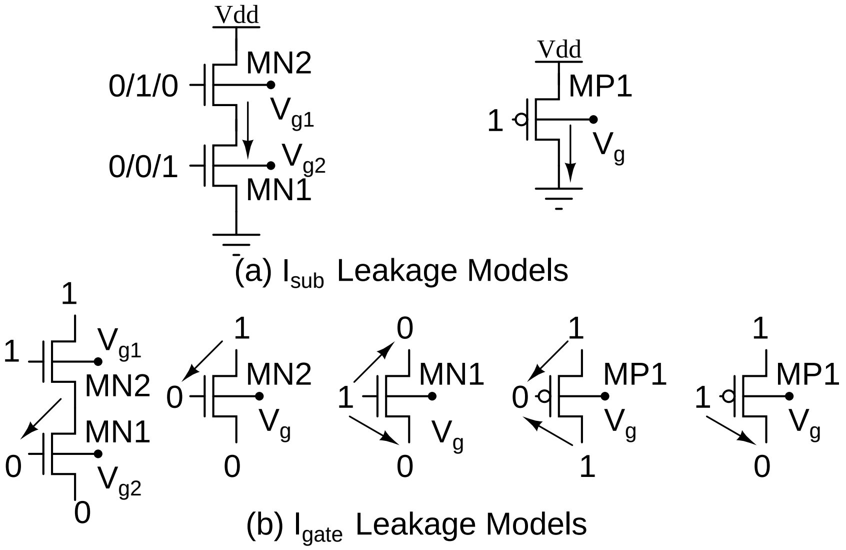


Figure 4.12: Stack models for leakage characterization of FinFET based NAND2 gates in Figure 4.11

Aforesaid formula is for effective width estimation without considering process variations. However in presence of process variation, any variation in process parameter can be accounted for similar change in *HFIN* and now Equation (4.26) can be used for effective width estimation. As can be observed from (4.23), dependence of leakage on process parameters in FinFET based technology is same as in bulk-MOSFET based technology. All the advantage described for bulk-MOSFET technology are also applicable here. To develop a single model for same input vector in all three configurations, leakage must be the function of process parameters, W_{eff}, V_{dd}, T, back-gate voltage (V_g). Based on the input vectors and leakage, stack models can be extracted and modeled using SVM with same procedure used for bulk-CMOS gate models. Figure 4.12 shows the common stack based models for leakage estimation of possible NAND2 configurations shown in Figure 4.11.

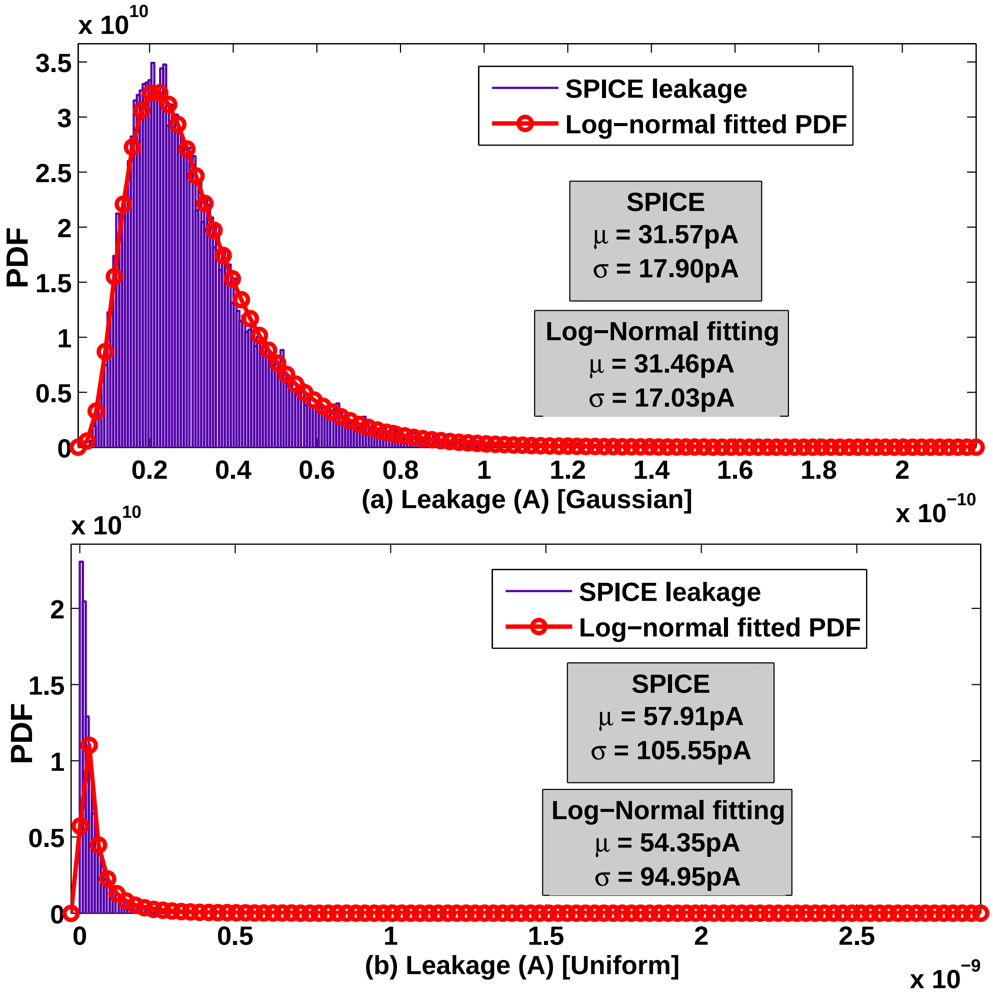


Figure 4.13: I_{sub} distribution of NAND4 gate with '0000' input for (a) Gaussian and (b) Uniform parameter variations at $V_{dd} = 1V$, $T = 27°C$ and Monte Carlo samples $= 10000$

4.6 Generalized Stack Models to account for Arbitrary parameter distributions

In this section, we show that our stack based SVM models are valid for any kind of parameter variations. Equation based analytical techniques shown in Equation (4.17) suffers from the large error incurred in leakage prediction of CMOS gates. Apart from this inaccuracy, two more problems are also associated with these techniques:

- For any change in parameter variation type, model fitting parameters need to be calculated again. This results in increase in the total model characterization time. Figure 4.13 shows the large difference between leakage distribution for Gaussian and uniform parameter variations, which suggests to develop separate fitting models for each case.

- μ and σ of leakage is predicted by underlying assumption of log-normal distribution of leakage in presence of process variations. However, this assumption is only true for parameter variations based on the Gaussian distribution. In advanced technologies with enhanced short channel effects, parameters can vary in arbitrary manner and different parameters can vary in a different vary. Thus, log-normal assumption is not valid for arbitrary parameter variations and may result into large error in leakage estimation of CMOS gates. Figure 4.13.(a) shows the log-normal assumption of I_{sub} is completely valid in case of Gaussian parameter variations but same log-normal assumption for uniform parameter variations is not valid and results in large error forI_{sub} leakage prediction as shown in Figure 4.13(b). However, this ∼5% - 10% error is for single gate and this error can result in large error for full-chip leakage estimation.

Our leakage models are more generalized and can analyze the effect of arbitrary variations in process parameters, supply and temperature on leakage because in our methodology, leakage of a gate is calculated by simulating samples in PVTW space using accurate leakage models without any assumption on leakage distribution.

4.7 Summary

We have proposed Support Vector Machine (SVM) regression based surrogate models to characterize the transistor stacks of CMOS gates simultaneously, accounting the effects of variations in transistor length (L), threshold voltage (V_{th}), oxide thickness (T_{ox}), supply voltage ($0.6\,V$-$1.2\,V$), temperature ($0°\,C$-100°C) and width ($28nm$-$200nm$), all scalable at the same time. Efficient methodology is developed by combining grid based techniques and global optimization techniques such as Genetic Algorithm (GA) and Differential Evolution (DE) in order to find out the best kernel among available kernels along with corresponding optimum kernel tuning parameters locally as well as globally in the tuning parameter range. Two way error driven active learning methodology is also employed, which selects the new samples from the input space for adaptive training of SVM based surrogate models. Sparse SVM models are also developed using Support Vector spectrum pruning method, which reduces the number of training samples used to prepare regression model. The resulting model has reduced the runtime with negligible increase in the error. SVM regression models generated in our approach have the ability to predict the leakage with maximum average error of 2.7% in mean (μ) and maximum average error of 3.1% in standard deviation (σ), both for OAI22 gate. Our results establish that there is on an average 10× improvement in runtime for estimating the μ and σ of leakage of a gate within 10000 Monte Carlo simulation loop. We have further developed Sparse SVM models using Support Vector spectrum pruning method, which reduces the runtime of the regression models with negligible increase in the error. Runtime efficiency of 17× is achieved on standard cell library using Sparse SVM models. For ISCAS'85 benchmark circuits, maximum 221× runtime improvement is achieved for C6288 circuit using our efficient sample selection methodology. Sparse SVM models are further developed, which improve the runtime efficiency of the model by reducing the number of training samples required to prepare the model with negligible increase in the mean and standard deviation error. Our models outperform previously available models based on either analytical equations or Artificial Neural Network in terms of accuracy. Our stack based methodology can be used for leakage characterization of post CMOS devices i.e. FINFET, CNTFET based logic gates. Proposed models can be used for leakage estimation of CMOS gates for non-Gaussian process parameter variations and methodology does not require to re-characterize the models. Proposed methodology removes the inaccurate log-normal assumption on leakage with respect to process parameters. Proposed models can be conveniently used in sampling based full-chip leakage estimation methodologies.

Chapter 5

Surrogate models for Static Virtual Ground Voltage Estimation

Figure 1.(a) shows the ground gating case, in which footer transistor is inserted between logic cluster and ground, Similarly, header in supply gating case as shown in Figure 5.1.(b) and combined gating case in Figure 5.1.(c).

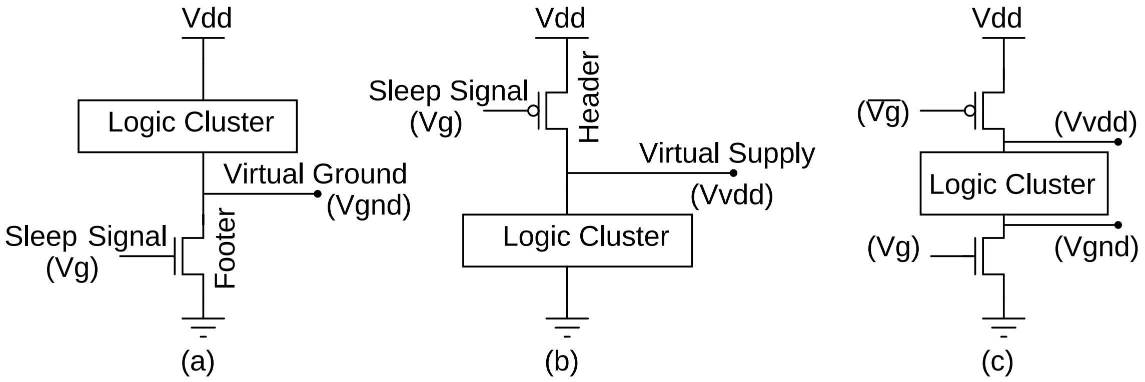


Figure 5.1: Power gating a) Ground gating case b) Supply gating case c) Combined ground and supply gating case

Large error incurred by inaccurate leakage models for power gated circuits can be removed by using machine learning methods such as Support Vector Machine (SVM), which is a kind of dynamic model without presuming any kind of exponential or polynomial form and can establish an accurate relation of I_{LC} in terms of V_{gnd} depending

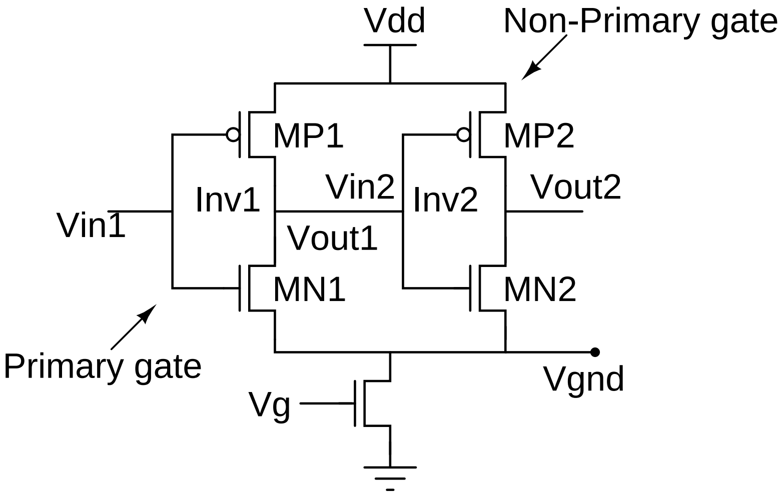


Figure 5.2: Input voltages conditions for non-primary inputs in ground gating case

on the complexity of the model. Another problem of inaccurate voltage assumption at input of CMOS gates is mitigated by using accurate input voltages at the input of CMOS gates. To explain it, first we classify all CMOS gates in the circuit in two categories i.e. primary gates and non-primary gates. Primary gates are CMOS gates in the circuit whose all inputs are supplied by users and inputs of non-primary gates are outputs of preceding gates. In Figure 5.2, Inv1 is the primary gate which is the receiving input from primary inputs whereas Inv2 is a non-primary gate whose input is a output of preceding gate 'Inv1'. Now suppose Vin1 = '1', MP1 is in 'OFF' condition but 'ON'/'OFF' condition for MN1 and the input value at 'Inv2' gate will depend upon the the value of V_{gnd}. **We define V_p as the maximum value of V_{gnd} for which pull down network (PDN) is 'ON' for primary gate (In Figure 5.2, transistor MN1) and pull up network (PUN) is 'ON' for non-primary gate (In Figure 5.2, transistor MP2).** This V_p defines the input voltage at non-primary gate Inv2. For the lower values of V_{gnd} than V_p, NMOS transistor MN1 is 'ON', which makes the Vin2 similar as V_{gnd}. For $V_{gnd} > V_p$, MN1 is 'OFF', i.e. Vin2 resides at a little bit higher value than V_{gnd}. Thus Vin2 can be defined using Equation (5.1).

$$Vin2 = \begin{cases} V_{gnd} & V_{gnd} < V_p \\ V_{gnd} + \triangle V & V_{gnd} \geq V_p \end{cases} \tag{5.1}$$

Here, $\triangle V$ is defined as the voltage drop across the PDN. Table 5.1 shows the output/input gate voltage of primary/non-primary gate Inv1/Inv2 and output of Inv2 gate of the circuit given in Figure 5.2. The difference between V_{gnd} voltage and Vout1/Vin2 is very less for lower values of V_{gnd} because the PDN is 'ON'. The drop across the PDN is very less and hence, can be removed from the circuit. Similarly, output of the non-primary gate Inv2 is close to the V_{dd} for lower V_{gnd} values and for higher values, this difference is high due to 'OFF' PUN network. For the calculation of V_p, MN1 and MP2 transistors should be 'ON' simultaneously for input Vin1 = '1'. Transistor MN1 will be 'ON' if gate to source voltage (V_{gs}) is greater than threshold voltage of the NMOS transistor (V_{thn}). i.e.

$$V_{gs}(MN1) > V_{thn} \tag{5.2}$$

$$V_{dd} - V_{gnd} > V_{thn} \tag{5.3}$$

$$V_{gnd} < V_{dd} - V_{thn} \tag{5.4}$$

Similarly, the condition for V_{gnd} to turn 'ON' MP2 PMOS transistor can be described as follows.

$$V_{gnd} < V_{dd} + V_{thp} \tag{5.5}$$

From Equation (5.4) and Equation (5.5), value of V_p can be given as

$$V_p = min(V_{dd} - V_{thn}, V_{dd} + V_{thp}) \tag{5.6}$$

For Multiple input NAND type gates, if any input is connected to connected to logic

‘0’, then output of that gate will be logic ‘1’ irrespective of the V_{gnd} voltage. When all inputs are at logic ‘1’, V_{gnd} voltage will play an important role to turn ‘ON’ the PDN network. On individual basis, maximum V_{gnd} for a gate depends upon the leakage i.e. input vectors of the gate. Leakage of a gate and V_{gnd} can be related as follows.

$$I_{LC} = I_{footer} \tag{5.7}$$

I_{footer}in terms of V_{gnd} can be given as[4].

$$I_{footer} = \begin{cases} \hat{I}_f.e^{K_N(V_{gnd}-V_{dd})} & V_{gnd} > 4V_T \\ 0 & V_{gnd} < 4V_T \end{cases} \tag{5.8}$$

From Equation (5.7) and Equation (5.8).

$$I_{LC} = \hat{I}_f.e^{K_N(V_{gnd}-V_{dd})} \tag{5.9}$$

$$V_{gnd} = V_{dd} + K_N^{'}.log(\frac{I_{LC}}{\hat{I}_f}) \tag{5.10}$$

From Equation (5.10), V_{gnd} will be higher for high logic circuit leakage. Generally leakage of a single gate is not enough to force the V_{gnd} to cross V_p. But, in a circuit, maximum V_{gnd} for a gate will be decided by the leakage contribution of other CMOS gates and can also cross the V_p voltage. In a NAND type logic, PDN will be ‘ON’ if all transistors in a stack are ‘ON’. Every transistor will have different V_{th} depending on its location and terminal voltages in the stack. For different gates, V_p will also be different. V_p for NAND type gates can be decided by the intersection of the V_p value of all NAND type gates as in Equation (5.11)

$$V_{p,NAND} = min(V_{p,INV}, V_{p,NAND2}, V_{P,NAND3}, V_{p,NAND4}....) \tag{5.11}$$

Similarly, V_p for NOR type gates can be described as in Equation (5.12)

$$V_{p,NOR} = min(V_{p,INV}, V_{p,NOR2}, V_{P,NOR3}, V_{p,NOR4}....) \tag{5.12}$$

From the definition, PDN in NAND and PUN in NOR type gates must be 'ON' simultaneously to calculate V_p. V_p of the complete circuit can be given by Equation (5.11) and Equation (5.12) as follows.

$$V_{p,circuit} = min(V_{P,NAND}, V_{P,NOR}) \tag{5.13}$$

The variation in V_{th} of different transistors in different CMOS gates makes the calculation of V_{gnd} little bit difficult. For accurate calculation of V_p, we slowly increase the V_{gnd} voltage and note the output node voltages of each gate presented in the circuit. However, V_p is one time calculation and will be same for all circuits. Thus, SPICE simulation can be used to calculate V_p. By observing the node voltages in Table 5.1, V_p can be given as $0.7\,V$.

Table 5.1: $V_{out_1}/V_{in_2}$and V_{out_2} node voltages for circuit in Figure 5.2 for varying V_{gnd}

V_{gnd} (mV)	200	500	700	800	900
$V_{out_1}/V_{in_2}(mV)$	200.01	500.01	700.63	808.61	980.87
$V_{out_2}(mV)$	999.98	999.99	999.99	999.95	992.64

To check whether V_p voltage will be the same for larger circuits, we simulate C880 ISCAS'85 benchmark circuit for different values of V_{gnd}. In Figure 3, we plot the output node voltages of each gate presented in the circuit for different V_{gnd} values. For lower V_{gnd}, all the output node voltages have their values either at V_{gnd} or V_{dd}. At $V_{gnd} = 0.7\,V$, there are very less values which are different than V_{gnd} but the difference is very less and can be treated as similar to V_{gnd} But, at $V_{gnd} = 0.8V$, the nodes with different voltages are in larger number, which significantly affect the leakage calculation of whole circuit. This can also be verified by observing the tip of the bar at V_{gnd} value which is highly reduced from $0.7V$ to $0.8V$. V_p value in this case is also $0.7V$ which is same as the circuit in Figure 5.2. Now, we can say that if different circuits consist of CMOS gates from same logic library, then the V_p value will be same. One important observation can be made here is that the input gate voltages of the CMOS gates in any circuit can be predicted only for $V_{gnd} < V_p$ only. For $V_{gnd} > V_p$, we can not use leakage models for CMOS gates due to unknown input gate voltages.

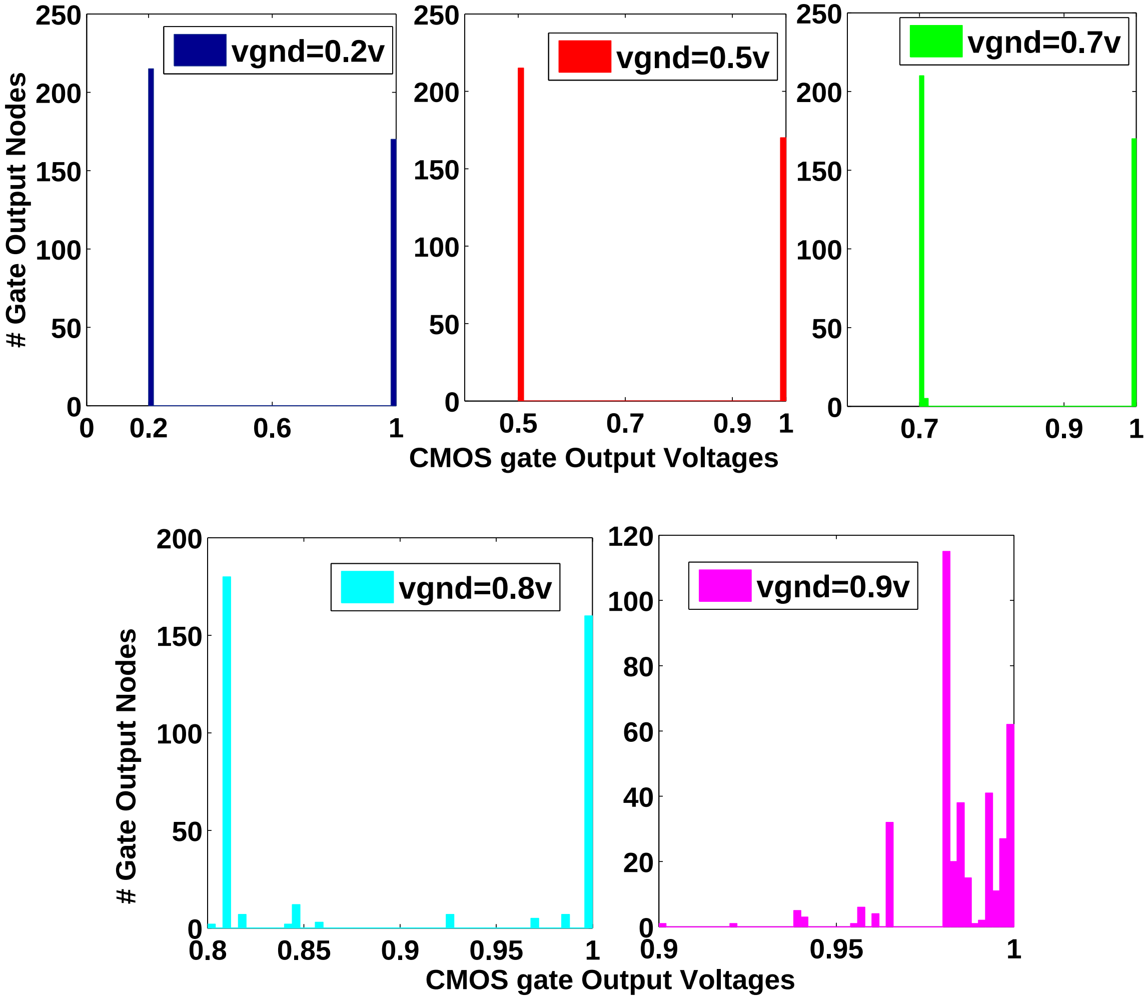


Figure 5.3: Fraction of number of CMOS gates presented in C880 ISCAS'85 benchmark circuit with output node voltages ranging between $0V$ to V_{dd} for varying V_{gnd}

For different values of sleep transistor parameters, V_{gnd} can be greater than $0.7V$.

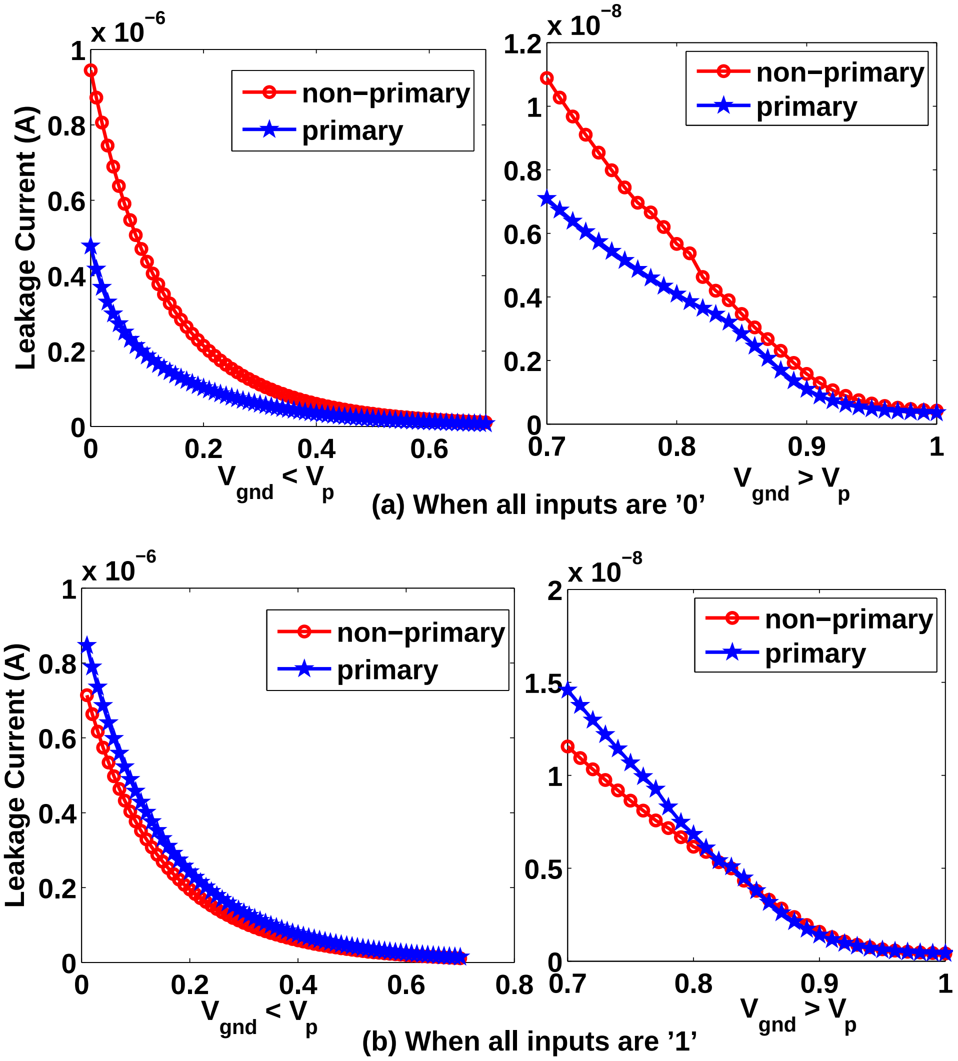
We can not use leakage models due to unknown input values at non-primary gates (Figure 5.3, $V_{gnd} = 0.8$v and 0.9v). Thus, KCL and KVL at V_{gnd} node can not be applied. To accurately predict the leakage of larger CMOS circuits for $0 < V_{gnd} < V_{dd}$, accurate leakage values of CMOS gates must be used. However, in [4], Equation (5.14) is used to calculate the leakage of a CMOS gate.

$$I_N = \begin{cases} \hat{I}_N.e^{-K_N V_{gnd}} & V_{gnd} < V_{dd} - 4V_T \\ 0 & V_{gnd} > V_{dd} - 4V_T \end{cases} \tag{5.14}$$

Authors in [4] handle above situation by assuming zero leakage for higher V_{gnd} values. This assumption may not be true for larger CMOS circuits because non-primary gates will be in larger quantity, giving significant leakage of CMOS circuit. Error in leakage modeling is propagated at higher level as leakage model for large circuits is prepared from simple CMOS gates. Figure 5.4 shows the leakage of primary and non-primary gates for two different input patterns with respect to the V_{gnd} voltage. The plot suggests that the leakage of a circuit is still significant at $V_{gnd} > V_p(0.7V)$. Leakage of the non-primary gates can not be neglected in comparison to primary gates because their leakage contribution in total circuit leakage can be higher than primary gates as shown in Figure 5.4.(a). Hence, zero leakage assumption for $V_{gnd} > V_p$ is a limitation of the model in [4]. In our methodology, we handle this situation by using leakage models only for $V_{gnd} < V_p$ and accurate SPICE simulation for $V_{gnd} > V_p$.

Our Contribution

- The leakage is characterized by SVM based regression models, which removes the inaccurate assumption of exponential linear dependency of leakage current of CMOS gates as a function of input voltages and V_{gnd}. This kind of model is highly desirable because for different values of V_{gnd}, the input gate voltages may take any value from $0\,V$ to V_{dd}.



(a) When all inputs are ’0’

(b) When all inputs are ’1’

Figure 5.4: Leakage variation of primary and non-primary gates of C880 ISCAS’85 benchmark circuit as a function of V_{gnd}

- To develop SVM models, our methodology uses the accurate value of leakage for all CMOS gates whether it is primary or non-primary for complete input range varying from $0\,V < V_{gnd} < V_{dd}$ while previous reported models have neglected the non-primary gates leakage for $V_p < V_{gnd} < V_{dd}$.

- SVM based regression methodology is used to develop static V_{gnd} models for higher accuracy and efficient computation.

- However, computations to generate training data using SPICE simulation for regression based models may take longer time for larger CMOS circuits. We develop a V_{gnd} partition based data generation methodology which uses SPICE simulation for higher values of V_{gnd} and our leakage current models with bisection search algorithm for lower values of V_{gnd}. This methodology results in significant saving of the model generation time. Unlike previous work, accurate input voltages of CMOS gates are considered for complete range of V_{gnd}.

- SVM classifier is developed to partition the V_{gnd} voltage depending as function of footer transistor parameters, providing the value of footer transistor parameters for which SPICE simulation is used ($V_{gnd} > V_p$) or our leakage current models with bisection search algorithm (Algorithm 5.1) are used ($V_{gnd} < V_p$).

- SVM based regression models are developed for capacitance estimation at V_{gnd} node i) due to CMOS gates as a function of input voltage and V_{gnd} voltage ii) due to footer transistor as a function of input voltage (V_g), width (W_{footer}), threshold voltage ($V_{th,footer}$) and V_{gnd} voltage.

- Piecewise simulation based methodology is developed for dynamic V_{gnd} estimation that efficiently uses the pre-developed leakage and static V_{gnd} models along with extra capacitance models.

Figure 5.5 shows the flowchart describing our proposed methodology. Steps of the proposed methodology in 'Yellow' boxes are independent of each other and can be performed parallelly. In Section 5.2, transistor stacks are characterized using regression based SVM models for leakage estimation of CMOS gates. Section 5.3 elaborates our equivalent stack identification method based on the input gate voltages and virtual ground voltage. In Section 5.4, Support Vector Classification (SVC) model is developed which tells us that for which values of input variables i.e. V_g, W_{footer},

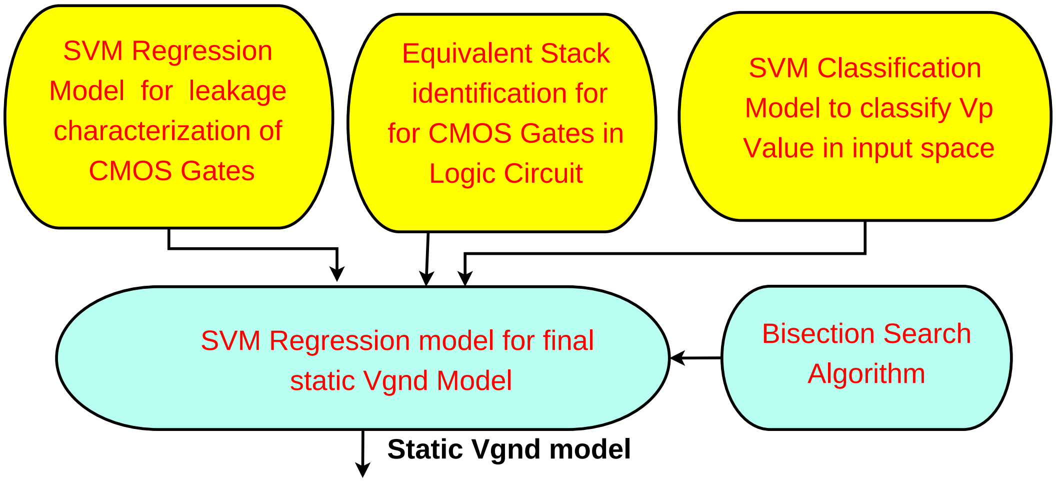

SVM Regression Model for leakage characterization of CMOS Gates
Equivalent Stack identification for for CMOS Gates in Logic Circuit
SVM Classification Model to classify Vp Value in input space
SVM Regression model for final static Vgnd Model
Bisection Search Algorithm
Static Vgnd model

Figure 5.5: Proposed methodology for static V_{gnd} estimation

$V_{th,footer}$, our stack based leakage models can be used. Outputs from Sections 5.2, 5.3 and 5.4 are used in Section 5.5 with bisection search algorithm to develop final static V_{gnd} model. A piecewise simulation based dynamic V_{gnd} estimation methodology is developed which needs the capacitance models to calculate capacitance at virtual ground node along with leakage models and static V_{gnd} model developed in previous sections.

5.1 SVM for Classification Problem

SVM can be used for two types of problems i.e. Support Vector Classification (SVC) and Support Vector Regression (SVR). In our work, we are using SVC for virtual ground voltage estimation of power gated circuits in terms of footer transistor parameters. In previous Chapter, we have described SVR formulations to model performance parameters. Formulations used for SVC are same as SVR except the values of performance parameters. In SVR, actual values of performance parameters are used while in SVC, it will be +1/-1. To the best of our knowledge, it is the first time that we are applying LS-SVM for virtual ground or supply voltage estimation of power gated circuits. SVC classifies the data into two sets in the multi-dimensional parameter space depending on the constraints (C_f). The feasible design space $S \subseteq R^N$ can be defined as in Equation (5.15). Note that x is the vector for all design parameters.

$$S = \{x : x \in R^N, C_f\} \tag{5.15}$$

Feasible function, $y(x)$, can only take two values {+1,-1} depending on whether $x \,\epsilon\, S$, defined as:

$$\hat{y(x)} = \begin{cases} +1 & if\, x \in S \\ \\ -1 & if\, x \notin S \end{cases} \tag{5.16}$$

Now, consider that a set of training data samples $\{(x_1, y_1), (x_2, y_2), ...(x_k, y_k)\} \subset R^N \times R$. Here, R^N denotes the input space. x_k is the input value (V_g, W_{footer}, $V_{th,footer}$) and y_k is the the corresponding target value in the form of +1 and -1 for k^{th} sample. The objective of SVC is to find a hyper-plane $w^T x_k + b$ with maximum separation of $2/ \parallel w \parallel_2$ between the data points of +1 and -1 type classes as shown in Figure 5.6. In high dimensional space, two classes may not be linearly separable. In LS-SVM classification, this problem is solved using kernel functions. Kernel functions transfer the original data into another feature space. Linear classification is applied to transferred input data into feature space. Kernel functions reduces the complexity in separating two classes and gives better accuracy than Neural network based approaches.

This problem is solved by formulating an optimization problem as follows.

$$\mathbf{P}: \quad min\, w, b = \frac{1}{2} w^T w \qquad s.t. \quad y_k[w^T x_k + b] \geq 1 \tag{5.17}$$

Here, the $\frac{1}{2} w^T w$ term denotes a cost function which is to be minimized for maximizing the separation. However for the Least-Squares SVC, modification is done on the target value such that an error variable e_k is allowed so that misclassifications can be

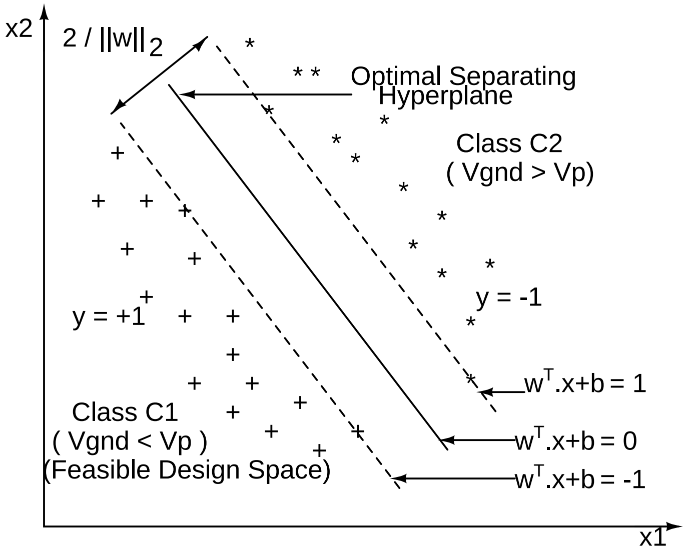


Figure 5.6: Linear SVM Classifier : Class C1 ($V_{gnd} < V_p$) and C2 ($V_{gnd} > V_p$)

tolerated in case of overlapping distributions and following optimization problem is formulated in the primal weight space for given a training set.

$$\mathbf{P}: \quad min\, J_p(w,e) = \frac{1}{2}w^T w + \gamma \frac{1}{2}\sum_{k=0}^{N} e_k^2 \tag{5.18}$$

This formulation involves a trade-off between the cost function term and sum of squared errors governed by the trade-off parameter γ.

$$y_k[w^T\phi(x_k) + b] = 1 - e_k, \qquad k = 1, 2......N \tag{5.19}$$

To solve primal minimization problem, we construct the dual maximization of Equation (5.18) using the Lagrangian form.

$$\mathbf{D}: \quad max_\alpha \mathcal{L}(w, b, e, \alpha) \tag{5.20}$$

$$\mathcal{L} = J_p(w,e) - \sum_{k=1}^{N} \alpha_k \left\{ y_k[w^T\phi(x_k) + b] - 1 + e_k \right\} \tag{5.21}$$

Dual problem is developed by constructing Lagrangians - where α_k's are the Lagranges multipliers. The conditions for optimality can be given as:

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial w}=0 \rightarrow w = \sum_{k=1}^{N} \alpha_k y_k \phi(x_k) \\ \frac{\partial \mathcal{L}}{\partial b} = 0 \rightarrow \sum_{k=1}^{N} \alpha_k y_k = 0 \\ \frac{\partial \mathcal{L}}{\partial e_k} = 0 \rightarrow \alpha_k = \gamma e_k,\ k = 1, 2.....N \\ \frac{\partial \mathcal{L}}{\partial \alpha_k} = 0 \rightarrow y_k[w^T \phi(x_k) + b] - 1 + e_k,\ k = 1, 2.....N \end{cases} \tag{5.22}$$

By eliminating e_k and w through substitution, following solution can be obtained as follows.

$$\begin{bmatrix} 0 & y^T \\ y & \Omega + I/\gamma \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ 1_v \end{bmatrix} \tag{5.23}$$

Here, $\Omega = Z^T Z$ and the kernel trick can be applied within α matrix as:

$$\Omega_{kl} = y_k y_l \phi(x_k)^T \phi(x_l) \tag{5.24}$$

$$\Omega_{kl} = y_k y_l K(x_k, x_l)......k, l = 1, 2....N \tag{5.25}$$

The resulting SVC model will become as follows.

$$y_k = sign \left[\sum_{k=1}^{N} \alpha_k K(x_k, x) + b \right] \tag{5.26}$$

Here, $K(x_k, x)$ is the kernel function and α_k, b are solution of the linear systems. For a function to be kernel function, it should be positive definite and must satisfy the mercer condition for the problem to be convex and hence, giving unique and optimum solution.

5.2 SVM based regression models for leakage current modeling of CMOS gates in power gated circuits

Stack based models have been proposed for leakage modeling in Chapter 3 of this thesis but those were based on varying the process parameters for a given input vector [54]. Since, in power gating case, the inputs of a CMOS gate can take any value out of 3 values i.e. $0\,V$, $1\,V$ and V_{gnd}. Thus, the number of models required will be very large for leakage estimation of CMOS logic circuit. To remove this disadvantage from previous work, we develop stack based models as a continuous function of input gate voltages, V_{gnd}. To understand our leakage modeling methodology through stacks, consider 3-input NMOS stacks as shown in Figure 5.7 with some of their possible terminal voltage conditions for both primary and non-primary CMOS gates. Stack1, Stack2, Stack 3 represent primary gate while Stack4, Stack5, Stack6, Stack7 represent non-primary gate terminal voltage conditions. For non-primary gates, any input can take V_{gnd} as input also. If all possible combinations are considered in discrete sense, then a large number of models will be required. To remove this disadvantage, we model subthreshold leakage (I_{sub}) as a continuous function of input voltages of stack by increasing the modeling space such that all possible combinations fall into this space. For example, let us the assume that $V_{gnd} = 0.1\,V$ for Stack4, then three inputs $1\,V$, $0.1\,V$ and $0.1\,V$ will be applied to the stack. If I_{sub} is modeled as a continuous function of input voltages where inputs can be varied from $0\,V$ to $1\,V$ for given $V_{gnd} = 0.1\,V$ then considered case will fall into modeled space. In our proposed methodology leakage models can be applied for $V_{gnd} < V_p$, therefore V_{gnd} and input voltages can take any value between $0\,V$ to V_p. We develop a transregional model, which combines both 'ON' and 'OFF' conditions of transistors of a stack into single model. To develop this model, input voltages of a stack and V_{gnd} is varied from $0\,V$ to V_{dd}. Another advantage of the proposed model is that we do not need to remove 'ON' transistors from the 'OFF' stack. 'ON' transistors can have high voltage drop across them in some cases such as Stack3, Stack4, thus affect the I_{sub} significantly.

While developing transregional models, leakage current's order can go from 10^{-12} to 10^{-6} for completely 'OFF' and 'ON' stack respectively. SVM may not be able to model this large range and provides error in modeling leakage. Authors in previous work suggest to model I_{sub} current as a exponential linear or quadratic model form. If we take *log* of the I_{sub} current then the exponential linear or quadratic model can be converted to simple polynomial model with linear or quadratic terms which is easier to model with less complexity [2]. More terms can also be added to improve the accuracy of the model. The larger difference between lowest and highest leakage is suppressed to 6 to 12 only. This range is easier to model. Thus, modeling *log* of leakage allows us to characterize lesser number of models with larger number of parameters for larger range in a single model which consequently reduces the time to develop the models and runtime for larger CMOS circuits.

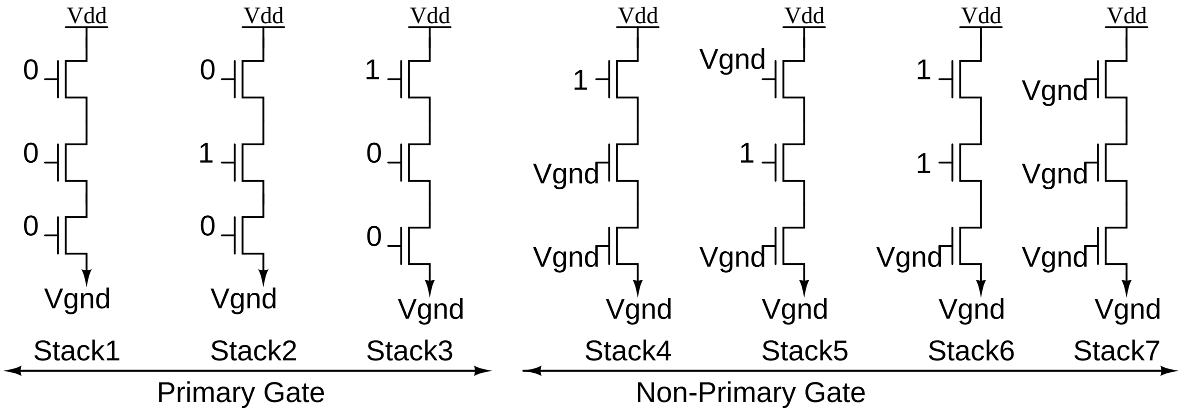


Figure 5.7: Possible terminal voltage conditions - NMOS stacks of primary and non-primary gates

In this work, we assume that the maximum stack-size is 4, as higher order stack will increase the delay of a gate due to increased logical effort. We have used conventions for labeling transistor stacks as follows.

{stack type}{stack size}

Here, stack type indicates whether it is an NMOS stack or a PMOS stack. Stack size represents the number of transistors on a stack.

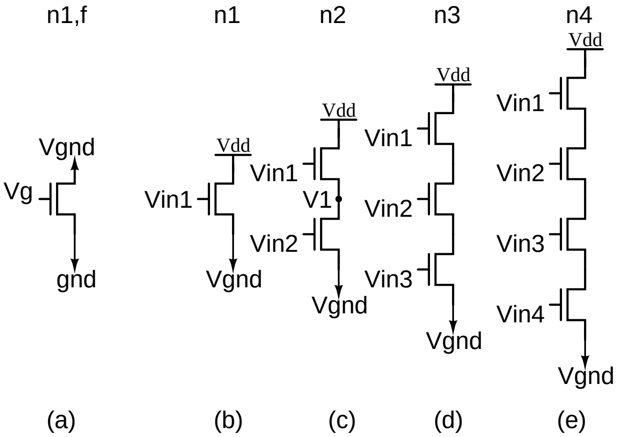


Figure 5.8: Characterized NMOS stack models in ground gating case

Figure 5.8 shows characterized transistor stacks for AND family of CMOS gates. Similarly, transistor stacks for OR family of gates can also be described. We model a transistor stack of each size of NMOS and PMOS type. Models n1 to n4 are for CMOS gates in logic cluster whereas n1,f is for footer transistor, which is different than model n1 with respect to the parameters used in the modeling. Now, we explain the conservative model (Equation (5.30)) used in previous work and thus, leads to errors in leakage modeling of CMOS gates. Consider a 2-transistor NMOS stack as shown in Figure 5.8(c). Leakage current for each transistor in terms of input voltages, V_{gnd}, and width of transistors on stack (W_{stack}) can be modeled as:

$$\begin{cases} I_1 = \hat{I}_1.W_{stack}.e^{k_1 V_{dd}+k_2 Vin1+k_3 V1} = I_{leak} \\ I_2 = \hat{I}_2.W_{stack}.e^{k_4 V1+k_5 Vin2+k_6 V_{gnd}} = I_{leak} \end{cases} \tag{5.27}$$

Dividing Equation (5.27).(a) with Equation (5.27).(b), we get:

$$m.e^{k_1 V_{dd}+k_2 Vin1+k_3 V1-k_4 V1-k_5 Vin2-k_6 V_{gnd}} = 1 \tag{5.28}$$

Here, $m = \hat{I_1}/\hat{I_2}$. Intermediate node voltage $V1$ can be calculated by solving Equation (5.28), as represented in Equation (5.29).

$$V1 = P.V_{dd} + Q.Vin1 + R.Vin2 + S.V_{gnd} \tag{5.29}$$

Putting the value of $V1$ in Equation (5.27), we get I_{leak} of stack as a function of input gate voltages and V_{gnd} as in Equation (5.30).

$$I_{leak} = \hat{I_{leak}}.W_{stack}.e^{-A.Vin1-B.Vin2-C.V_{gnd}} \tag{5.30}$$

It should be noted that Equation (5.30) is obtained by neglecting the term $(1-e^{-V_{ds}/V_T})$ in Equation (2.12) and does not consider the effect of 'ON' transistors on 'OFF' stack, which results in ∼20% error in leakage estimation of transistor stacks. To reduce this error, we use SVM based regression models which are better modeling methods for non-linear mapping of input to output parameters. Simulation data for regression model is generated from SPICE tool. Figure 5.9 shows the fitted transregional curve for n1 stack given in Figure 5.8(b). This transregional model shows smooth characteristics in the considered input space, which is incorporated into SVM models with high accuracy. The advantages of using SVM regression models are: 1) accurate and reduced number of models, 2) consideration of effect of states of CMOS gates and 3) effect of 'ON' transistors in 'OFF' network. Our SVM based models can also be used for leakage current estimation of CMOS consisting parallel combination of 'OFF' transistors or series of 'OFF' stacks. Width of the transistors is also added to the model for leakage estimation of CMOS gates with parallel transistors i.e. AOI22, AOI32, OAI22, OAI23. Methodology based on effective widths can be used to combine parallel transistors based on input voltages [116]. In this work, we only consider simple CMOS gates in logic clusters because most of the gates have simple parallel structures such as- NAND gates, Buffers, NOR gates, OAI and AOI gates. To the best of our knowledge, it is the first attempt to calculate leakage of CMOS gates in power gated circuits with a very less number of models considering actual node voltage conditions during circuit simulation.

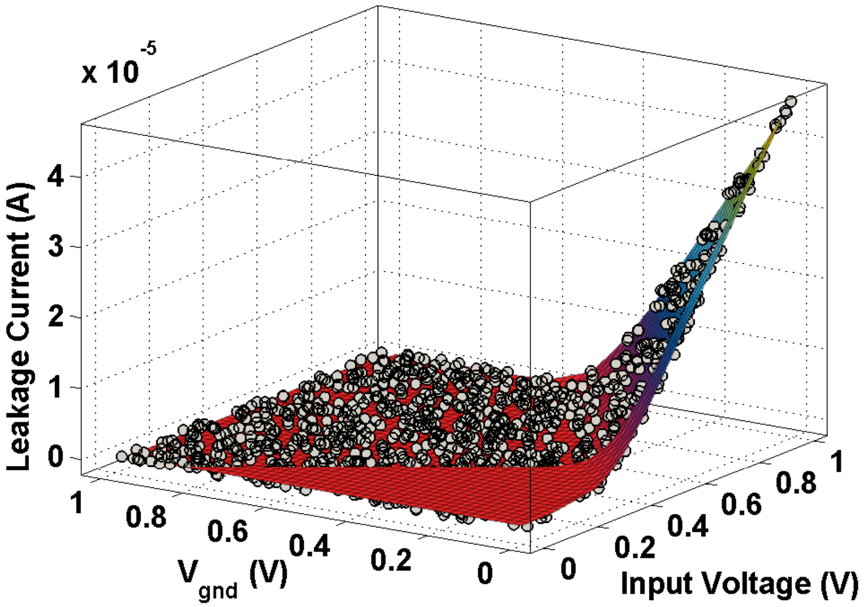


Figure 5.9: Fitted transregional leakage model for n1 stack in Figure 5.8 as function on input voltages and V_{gnd} voltage

5.3 Finding the equivalent stack models of CMOS gates

In this Section, we formulate some rules for finding the equivalent stack model of a CMOS gate on the basis of the type of gate whether it is a primary or non-primary gate and input to the gate because the input vectors have a significant impact on the leakage current of a CMOS gate. We have also made following valid assumptions in extracting stacks of CMOS gates.

- For $V_{gnd} < V_p$, one of the network either PDN or PUN is removed. Drop across the PDN or PUN is very less and causes $< 0.1\%$ error in connecting V_{dd} or

V_{gnd} to output node of the gate. Thus. the effect of removing PDN or PUN on leakage is negligible [4]. This follows from the fact that drain to source I_{sub} is the cause of drop across that network. But, this current is negligible across 'ON' network and turns into negligible drop. The validity of our assumption is confirmed in Figure 5.3, illustrating that all gates in CMOS circuit have their output either at V_{gnd} or V_{dd} for $V_{gnd} < 0.7V$. This drop is very high for V_{gnd} $> 0.7V$ due to 'OFF' both PUN and PDN. We accurately consider this drop by SPICE simulation in development of models.

- We have ignored the gate tunneling current (I_{gate}) in the CMOS gates. Consideration of gate tunneling current will increase the number of models and characterization time. This assumption does not affect the stack extraction methodology and V_p voltage. However, including it also increase the SPICE simulation time in a similar manner and still our model has advantages over SPICE simulation based methodology for data generation. In addition, I_{gate} is reduced by more than 25× in NMOS and more than 1000× in PMOS by introduction of High-K dielectrics [117]. I_{sub} is still a dominant leakage component due to its dependence on temperature and short channel effects. Hence, adding I_{gate} is just adding the extra models to our methodology.

- In our methodology, while estimating dynamic V_{gnd}, input vector to the circuit is assumed to be known and remains constant during mode transition. Power gating technique is applied in the standby mode only because it causes the loss in information saved at the output node of CMOS gates. In standby mode, there is no point of changing the input vectors. Constant input vectors during the circuit's mode transition is a valid condition in power gated circuits. Same condition has been assumed by many previous researchers also [4, 118, 119, 120]. To apply power gating during runtime, some circuitry is required for data retention of flip-flops [121]. Runtime power gating approach is completely different than standby power gating approach where sleep periods are generally very long. Dynamic V_{gnd} model is important which can estimate the time for which sleep period should be applied such that the energy saved due to power gating is higher than the energy consumed by the circuit going from wake-up to sleep mode [4].

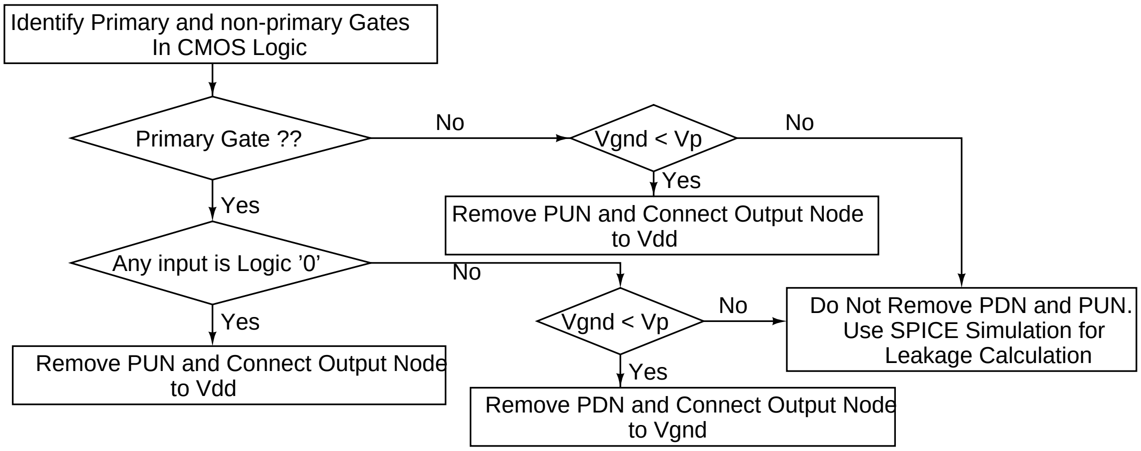

Identify Primary and non-primary Gates In CMOS Logic
Primary Gate ??
No
Vgnd < Vp
No
Yes
Yes
Remove PUN and Connect Output Node to Vdd
Any input is Logic '0'
No
Vgnd < Vp
No
Do Not Remove PDN and PUN. Use SPICE Simulation for Leakage Calculation
Yes
Remove PUN and Connect Output Node to Vdd
Yes
Remove PDN and Connect Output Node to Vgnd

Figure 5.10: Stack extraction - NAND family of gates; depending on the input as well as V_{gnd} voltage

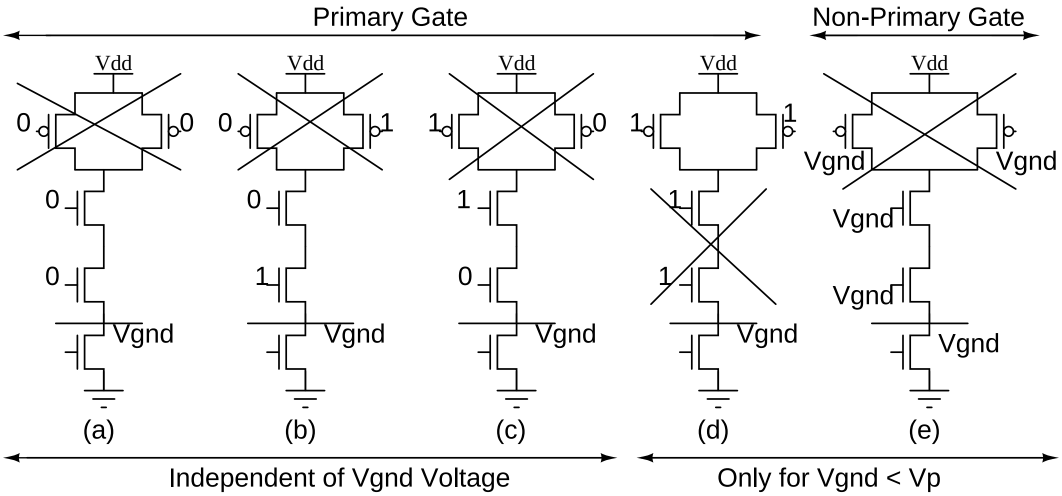


Figure 5.11: Equivalent stack models - different input vectors of 2-input NAND, ground gated

A flowchart to extract stack for NAND family of gates is shown in Figure 5.10. Consider a 2-input NAND gate as shown in Figure 5.11. In our methodology, we remove either PDN or PUN based on the input voltages and V_{gnd} value. These rules can be described as follows (The following rules are only for NAND family of gates. Similarly , rules can be described for NOR family of gates):

1. For any primary gate, if any input to the CMOS gate is at logic '0', then remove PUN from that CMOS gate and connect output node to the V_{dd} because logic '0' input will make the PUN 'ON'. This rule is independent from the value of V_{gnd} because it does not affect the gate to source voltage of PMOS transistors in PUN.

2. For any primary gate, if all inputs are at logic '1' and if V_{gnd} is less than V_p, remove PDN and connect output node to V_{gnd} otherwise don't remove PUN and PDN.

3. For any non-primary gate whose all inputs are outputs of preceding gates, remove PUN if and only if $V_{gnd} < V_p$ because PMOS transistor in PUN makes it 'ON' and output node can be connected to V_{dd} otherwise don't remove PUN and PDN.

Above rules can also be applied to parallel 'OFF' transistor stacks whether there is only one 'OFF' transistor in each stack or multiple 'OFF' transistors with 'ON' transistors in a stack as shown in Figure 5.12. Equivalent model can be derived by summing the currents from all the stacks as represented in Equation (5.31).

$$I_{stack,eq} = I_{stack1} + I_{stack2} + I_{stack3} + ... + I_{stackn} \tag{5.31}$$

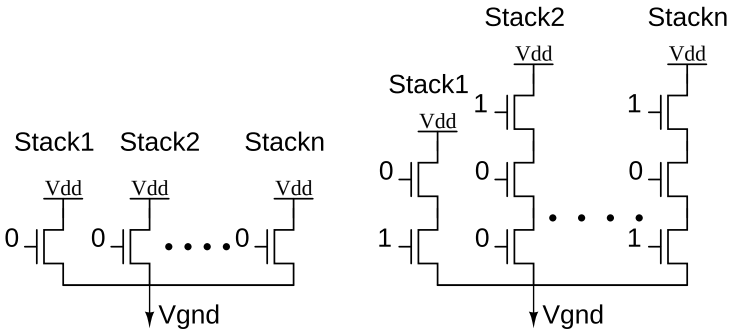
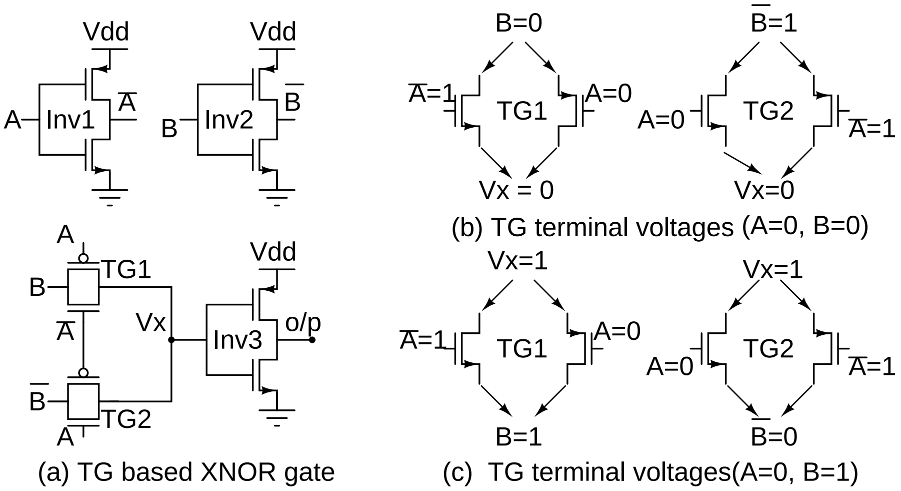


Figure 5.12: Parallel 'OFF' transistor stacks in CMOS gates

To apply above stack extraction rules in other technologies, we consider a transmission gate (TG) based 2-input XNOR gate as shown in Figure 5.13. Power gating scheme only reduces the subthreshold leakage current flowing from supply to ground. TG based gates can not be connected directly to power gated footer/header transistor. Generally, DTCMOS (Dual-threshold CMOS), MTCMOS (Multi-threshold CMOS) techniques are used to reduce the leakage TG based circuits. However, CMOS gates in TG based circuits (i.e. inverter in Figure 5.13.(a)) can be connected to power gated transistor. Now we show that our stack extraction rules can be applied to TG based circuits. One important observation on the input voltage of CMOS gates which can be made here is that CMOS gates either have input voltages perfectly $0\,V/\,V_{dd}$ (In case of primary gates) or V_{gnd} voltage at all inputs. TG based circuits also comes under this category. To explain this, we first consider 2-input TG based XNOR gate with two input cases as A=0, B=0 and A=0, B=1. For first case, TG1 is 'ON' and TG2 is in 'OFF' condition. Value at intermediate node V_x is decided by 'ON' TG. This makes the Vx voltage equal to the value at input 'B' i.e. Vx = $0\,V$. Terminal voltages for both TG's TG1 and TG2 is shown in Figure 5.13.(b). I_{sub} of TG1 will be zero due to TG1's 'ON' condition. TG2's NMOS and PMOS transistors have terminal voltages perfectly at $0\,V/V_{dd}$. Leakage of both transistors in TG2 can be calculated by our pre-characterized stack based models. For second input case, again TG1 and TG2 is in 'ON' and 'OFF' condition respectively. Terminal voltage conditions for both TG's are shown in Figure 5.13.(c). Pre-characterized stack models can also be used in this case. The input voltage conditions for the inverters Inv1, Inv2 and Inv3 are same as the primary gates in power gated CMOS gates. Similarly, stack models can be extracted for other input vectors also.



(a) TG based XNOR gate

(b) TG terminal voltages (A=0, B=0)

(c) TG terminal voltages(A=0, B=1)

Figure 5.13: Transmission Gate (TG) based XNOR with terminal voltages of TG under different input combinations

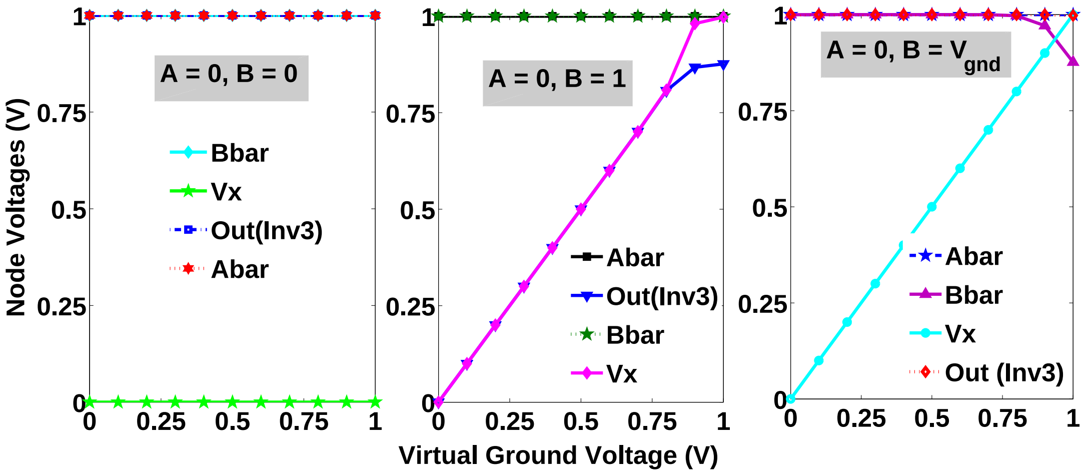


Figure 5.14: Node Voltages under different Input conditions of 2-input TG based XNOR gate

Now we consider case when the inputs of the TG's are outputs of preceding gates. Generally, inputs of any TG in the circuit will be from output of the inverter, whose value will be either at V_{gnd} or V_{dd} for $V_{gnd} < V_p$. Let's assume the value at input A and B is $0\,V$ and $V_{gnd} < V_p$ respectively. One of the important properties of TG is that it will pass any voltage ranging from $0\,V$ to V_{dd}. This condition puts one of the transistor either NMOS or PMOS of TG1 in 'ON' condition and Vx $=$ V_{gnd}. These transistor's terminal conditions are same as the NMOS/PMOS transistors of power gated CMOS inverter with input '1'/'0'. Due to low resistance path provided by TG1, source and drain voltages of TG1 are same and subthreshold leakage across this TG will be zero. Input $\bar{B}$ is the output of the CMOS inverter with input B. According to the rule 3, output of inverter i.e. $\bar{B}$ will be connected to V_{dd}. Terminal conditions of transistor TG2 are perfectly defined and will be defined as shown in Figure 5.13.(b) but with Vx $=$ V_{gnd}. Leakage can be calculated using pre-characterized stack models in this case also. Figure 5.14 shows the node voltages under different input conditions of 2-input TG based XNOR gate. Node voltages in the circuit are perfectly defined for $V_{gnd} < V_p$ which takes one value from set of three values i.e. $0\,V$, V_{dd} and V_{gnd}. Same V_p rule and value is also applicable in TG based circuits. The only discrepancy is in estimation of the node voltages for $V_{gnd} > V_p$.

Other technologies like Domino logic, Dynamic logic are variants of CMOS logic. Already developed stack based leakage models can be used to apply power gating methodology in theses technologies also. Our stack based methodology can be used for a wide variety of technologies.

5.4 SVM classification for input space partitioning based on V_p value

Before developing SVM classifier (SVC), we first need to check whether SVC is required or not in the given parameter space because V_{gnd} highly depends upon the leakage i.e. input vector applied to the circuit. This can result in saving of characterization time. Singh *et. al.* [87] developed the first order V_{gnd} model as in Equation

(5.32). This is required because for an applied input vector to the logic circuit and footer transistor parameters maximum V_{gnd} is less than V_p. This step saves in characterization time for V_{gnd} model generation.

$$V_{gnd} = \frac{-V_g + S_S log_{10}(\frac{W_{circuit}}{W_{footer}}) + (V_{th,footer} - V_{thc}) + \eta V_{dd}}{2\eta} \tag{5.32}$$

Here, V_g is footer gate voltage, $V_{th,footer}$, $W_{circuit}$ and V_{thc}, W_{footer} are threshold voltage and width of logic circuit and footer transistor respectively, η is the DIBL coefficient and S_S is the subthreshold slope. In this work, we are only considering V_g, W_{footer}, V_{thf} as design variables. Thus, maximum value of V_{gnd} will be for the lowest value of both V_g and W_{footer} and highest value of V_{thf} . If this maximum value is greater than V_p, then only classifier is required. Next, we need to calculate V_p value for which classifier is to be developed. According to our discussion in previous sections, we can only use our models if and only if $V_{gnd} < V_p$ and V_p is calculated by varying V_{gnd} and observing the CMOS gate's output voltages in the logic circuit. V_{gnd} is varied until output voltages are at either V_{dd} or V_{gnd}. However, this is only a one time process, which will be same for all CMOS circuits.

Since, our aim is to develop a model for V_{gnd} estimation as a function of input voltage, width and threshold voltage of footer transistor. It will be very costly in terms of modeling time to generate regression data directly from the SPICE simulation. As we know, we can use our proposed leakage models for $V_{gnd} < V_p$. But, we do not know the functional relation between footer transistor parameters and V_{gnd}. In this work, we develop a classifier which can separate V_{gnd} values either less than V_p or greater than V_p in the three dimensional space (input voltage, width and threshold voltage of footer transistor). The original SVM classification problem can be used here as follows: First of all, we define the feasible space i.e. $V_{gnd} < V_p$, for which we can use our pre-characterized stack based models for data generation to formulate regression based models for V_{gnd} estimation in terms of footer transistor parameters. The feasible design space $S \subseteq R^N$ based on feasible constrained C_f can be defined as in Equation (5.33).

$$S = \{x_{footer} : x_{footer} \in R^N, C_f\};\ C_f = \{V_{gnd} < V_p\} \tag{5.33}$$

Feasible function, $y(x_{footer})$, can only take two values {+1,-1} denoted as Class C1 ($V_{gnd} < V_p$) and Class C2 ($V_{gnd} > V_p$) depending on whether $x_{footer} \epsilon S$, defined as in Equation (5.34).

$$y(\hat{x}) = \begin{cases} +1 & if\, x_{footer} \in S \\ -1 & if\, x_{footer} \notin S \end{cases} \tag{5.34}$$

5.5 SVM Regression models for Static V_{gnd} Estimation

In the following section, we discuss about the development of SVM regression (SVR) based surrogate V_{gnd} models using pre-characterized leakage models, SVC and bisection search algorithm. Our SVR model generation methodology starts with the initial training data-set which is directly taken from the data used for SVC model with the actual values of V_{gnd} instead of +1 and -1 as used in SVC model generation. We can do this without incurring any error in models because SVC and SVR models are completely independent. This process saves our time for data generation which can be very high for larger CMOS logic circuits. At this point, the model is trained and tested on testing data set. Accuracy of the model is calculated based on mean square error (MSE) between model output $Y_{test,est}$ and actual output Y_{test} using Equation (5.35).

$$ERROR_{est} = \text{MSE}(Y_{test},\ Y_{test,est}) = \frac{1}{n}\sum_{i=1}^{n}(Y_{test} - Y_{test,est})^2 \tag{5.35}$$

If the accuracy of the model is less than the desired accuracy, then new samples are generated around maximum error sample. SVC selects whether our stack models or SPICE simulation is used for the V_{gnd} estimation for these newly generated samples.

Now consider a CMOS logic circuit applied with some input vector. For the case of $V_{gnd} < V_p$, we separate each gate according to the type and input pattern to a particular CMOS gate. Suppose m$_1$ gates of same gate type (GT) have same input pattern then leakage current for all those gates can be summed to single lumped current source and can be represented as m_1*$GT_{1,leak}$. The total leakage current for the logic cluster can be represented as in Equation (5.36).

$$I_{LC}(logic - cluster) = \sum_{i=1}^{n} m_i.GT_{i,leak} \tag{5.36}$$

$$m_1 + m_2 + ... + m_n = N$$

Here, n and N are the total number of CMOS gate types and CMOS gates present in the logic cluster respectively. The V_{gnd} estimation problem in static condition can be defined as to find out the value of V_{gnd} for which leakage current of logic cluster and footer transistor is same. The above problem can be solved using algorithm shown in Figure 5.1.

Our algorithm 5.1 for V_{gnd} estimation in Figure starts with inputs as pre-characterized stack based leakage current models and maximum virtual ground voltage V_p that can be predicted using our models. Two parameters $V_{gnd,high}$ and $V_{gnd,low}$ define the maximum and minimum value as V_p and $0\,V$ respectively. Another parameter $V_{gnd,start}$ in line 3 defines the middle value in the search range. In line 4, our SVM based regression surrogate models of leakage current are used for leakage estimation of logic cluster and footer transistor which depends on current $V_{gnd,start}$ value and difference between logic circuit leakage and footer transistor leakage is calculated in line 5. $\triangle I_{D,Max}$ defines the maximum tolerable limit which can be possible between

Algorithm 5.1 Bisection search algorithm to find virtual ground voltage value for $V_{gnd} < V_p$

```
Input: Pre-characterized leakage current models, $V_p$ (Maximum value than can be predicted using using our leakage current models)
Output: Virtual ground voltage ( $V_{gnd}$)
1. $V_{gnd,high}$= $V_p$
2. $V_{gnd,low}$= $0\,V$
3.$V_{gnd,start}$= ( $V_{gnd,high}$+ $V_{gnd,low}$ )/2
4. Estimate $I_{LC}$, $I_{footer}$= $f(V_{gnd,start})$
5.$\Delta I_D$= $I_{LC} - I_{footer}$
6. $\Delta I_{D,Max}$= $\in$
7.While ( $|\Delta I_D| \geq \Delta I_{D,Max}$ ) do
8.        If ($\Delta I_D > \in$) then
9.            $V_{gnd,low}$= $V_{gnd,start}$
10.       Else
11.           $V_{gnd,high}$= $V_{gnd,start}$
12.       end If
13.       $V_{gnd,start}$= ( $V_{gnd,high}$+ $V_{gnd,low}$ )/2
14.       Estimate $I_{LC}$, $I_{footer}$= $f(V_{gnd,start})$
15.       $\Delta I_D$= $I_{LC} - I_{footer}$
16. end While
17. $V_{gnd} = V_{gnd,start}$
```

difference of logic circuit leakage and footer transistor leakage for a given V_{gnd} value. If $\triangle I_D$ is positive and larger than the $\triangle I_{D,Max}$, then lower value ofV_{gnd}, $V_{gnd,low}$ is set to $V_{gnd,start}$ otherwise higher value of V_{gnd} is set to $V_{gnd,start}$. Now, new $V_{gnd,start}$ value is calculated by averaging the new $V_{gnd,high}$ and $V_{gnd,low}$ value. Performing the steps in lines 9 - 13, reduce the search space half for the next iteration. This process is repeated until difference in logic circuit and footer leakage is under tolerable limits.

5.6 Extension of the proposed Static V_{gnd} model for supply gating case

As first step, we need to calculate V_p voltage, which can be defined as the value of virtual supply voltage (V_{vdd}) for which PUN is 'ON' for primary gate(In Figure 5.15, transistor MP1) and PDN is 'ON' for non-primary gate(In Figure 5.15, transistor MN2).

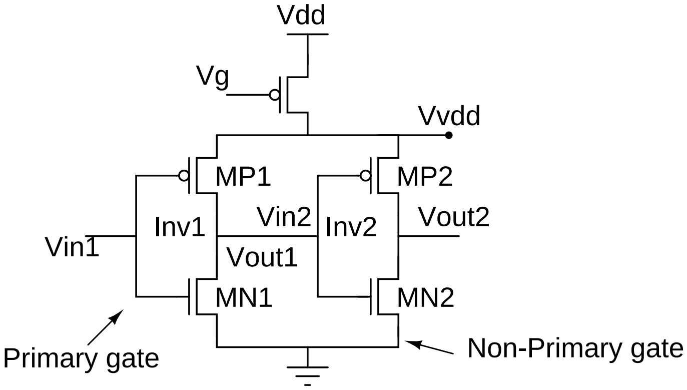


Figure 5.15: Input voltages estimation - CMOS gates with non-primary inputs in supply gating

For Vin1 $= 0\,V$ and $V_{vdd} >\ V_p$, PMOS transistor MP1 is 'ON' which makes the Vin2 similar as V_{vdd}. For $V_{gnd} <\ V_p$, MP1 is 'OFF', i.e. Vin2 resides at the little bit lower value than V_{vdd}. Thus, Vin2 can be defined using Equation (5.37).

$$Vin2 = \begin{cases} V_{vdd} & V_{vdd} > V_p \\ V_{vdd} - \triangle V & V_{vdd} \leq V_p \end{cases} \tag{5.37}$$

Here, $\triangle V$ is defined as the voltage drop across PUN. To calculate V_p, we simulate the circuit shown in Figure 5.15 for different values of V_{vdd} and observed the input and output voltages of each gate in the circuit as shown in Table 5.2. At V_{vdd}= $0.3V$, input of non-primary gate Inv2 is very close to V_{dd} and Vout2 is very close to ground voltage. Hence, V_p voltage can be given as $0.3V$. Same value of V_p is obtained by simulating larger circuits as done previously.

Table 5.2: V_{out_1}/V_{in_2} and V_{out_2} node voltages - circuit shown in Figure 5.15 for varying V_{vdd} and $V_{in_1} = 0\,V$

$V_{vdd}\ (mV)$	200	300	400	700	900
$V_{out_1}/V_{in_2}(mV)$	197.69	299.60	399.89	699.96	899.94
$V_{out_2}(mV)$	0.1425	0.0242	0.0064	0.0038	0.0083

In the second step, SVM regression based leakage models for transistor stacks shown in Figure 5.8, are also need to be changed in supply gating case. In NMOS stacks of Figure 5.8, drain voltage of top transistor was at V_{dd} and source voltage of bottom transistor was varied according to V_{gnd} voltage. In supply gating case, drain voltage of top transistor will be varied according to the V_{vdd} and source voltage of bottom transistor will be at ground voltage as shown in Figure 5.16. Note that footer transistor model will be removed because the footer transistor will be converted to header transistor i.e. PMOS transistor in this case. Similarly PMOS stack models can also be described.

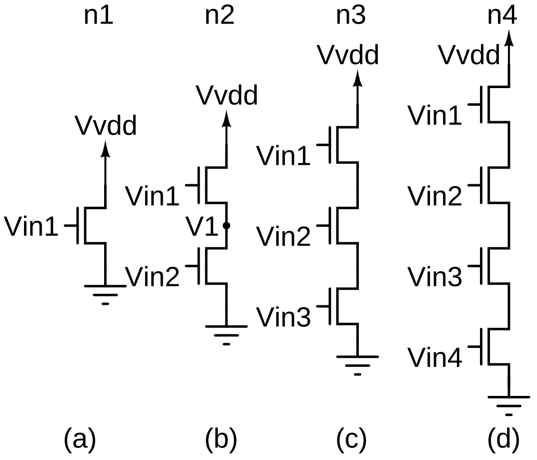


Figure 5.16: Characterized NMOS stack models in supply gating

In the third step, rules for extracting equivalent stack models for CMOS gates should be developed based on V_{vdd} and input voltages of CMOS gates.

Consider a 2-input NOR gate as shown in Figure 5.17. These rules can be described

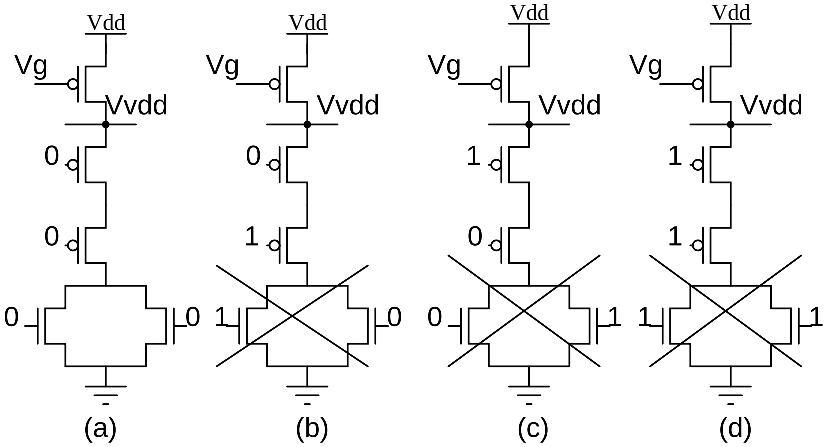


Figure 5.17: Equivalent stack models - different input vectors of 2-input NOR gate in supply gating

as follows (The following rules are only for OR family of gates, similarly, rules can be described for AND family of gates):

1. For any primary gate, if any input to the CMOS gate is at logic '1', then remove PDN from that CMOS gate and connect output node to the ground node because logic '1' input will make the PDN 'ON'.

2. For any primary gate, if all inputs are at logic '0', and if V_{vdd} is greater than V_p, remove PUN and connect output node to V_{vdd}, otherwise don't remove PUN and PDN.

3. For any non-primary gate, remove PDN if and only if $V_{vdd} > V_p$ because NMOS transistor in PDN makes it 'ON' and output node can be connected to ground node, otherwise don't remove PUN and PDN.

In fourth step, SVM regression based methodology can be used to highly accurate stack models. In supply gating case, we can only use our stack models for $V_{vdd} > V_p$, thus SVC model needs to be developed in the next step. V_p value used in this case will be $0.3V$. In final step, final SVM regression based static V_{vdd} model is developed using different models developed in previous steps with bisection search algorithm.

V_{vdd} is to be searched in the range from V_p to V_{vdd} using bisection search algorithm. It can be concluded that the accuracy and efficiency of the final static V_{vdd} model will be of same order as static V_{gnd} model because both cases use same type of stack models, SVC model and training samples are also same to develop all models.

5.7 Extension and use of ground gating and supply gating static models in combined gating case

In combined gating case, header and footer transistors are applied simultaneously as shown in Figure 5.1.(c). To develop the models for V_{vdd} and V_{gnd}, V_p values will be same as the ground gating and supply gating case because we are using the same type of CMOS gates in this case also. SVM regression based stack models and rules to find equivalent stack models can be directly taken from both supply and ground gating cases.

However, SVC model needs to be changed in this case. This is because both the transistors can take any value from their design parameter space. If PMOS header transistor is ‘OFF’, then there will be some drop across this transistor and V_{dd} voltage will be different for the ground gated transistor which will also affect the V_{gnd} voltage. Ground voltage will also be different for supply gated transistor. Thus, we need to develop a classifier model for both $V_{gnd} < V_p$ and $V_{vdd} > V_p$ and can take header and footer transistor parameters at the same time and separate the input parameter space to use our leakage models. Final V_{gnd} and V_{vdd} model should also be developed in terms of both header and footer transistor parameters. Since the number of input parameters have been increased, more number of training samples are required which will increase the model characterization time and runtime. However, the model characterization time will be less compared to SPICE characterization time. In terms of accuracy, SVM models are better than the analytical models and can handle high dimensional model compared to the analytical models [4] [5].

5.8 Experimental Results

We have used LS-SVM toolbox which is an advanced version of SVM for improving efficiency of the model. We have used RBF kernel $K(x, x_k) = exp(||x - x_k||^2/\sigma^2)$ with kernel function variable σ and regularization parameter γ as 1 and 10 respectively. However, there may exist other kernels which may provide more accurate and less complex models than we have considered in this Chapter. Adding more number of kernels increases the characterization time to develop the final models. Thus, choosing the number of kernels is a trade-off between characterization time, accuracy and runtime of the models. We compared our results on different CMOS circuits in $28nm$ technology. Only I_{sub} is considered in leakage current estimation of logic circuit and footer transistor. V_{gnd} model is developed only in terms of $W_{footer})$, V_g and V_{th} of the footer transistor in the range of $28nm$ - $500nm$, $0\,V$ - $0.25V$ and $0.35V$ - $0.5V$ respectively. It should be noted that the maximum value of V_g for V_{gnd} model should be selected for which footer transistor is in 'OFF' condition.

5.8.1 Accuracy and Efficiency Evaluation of Leakage models

This section evaluates accuracy and efficiency of proposed leakage current models. In Figure 5.18, we compare MSE of 4-input NMOS stack model obtained through the training data generated from different sampling techniques. It shows that the MSE of adaptively trained model is lower and constant MSE is also achieved well before than the fixed sampling based models. Our adaptive training methodology is able to reduce the number of samples to achieve the same MSE as compared to the methodology in [122]. Table 5.3 shows the accuracy of our characterized stack models in terms of correlation coefficient and MSE with respect to SPICE output. We compare our

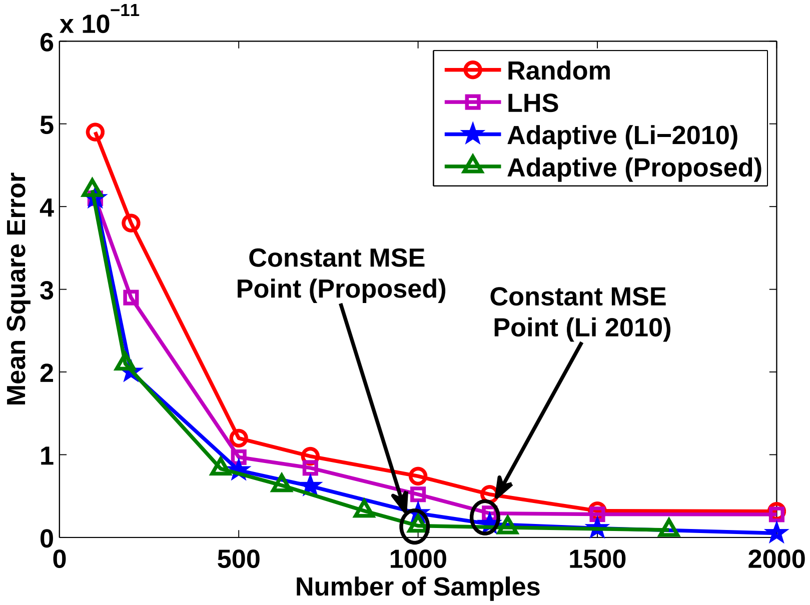


Figure 5.18: Comparison of different sampling techniques - to train 4-input NMOS stack model

Table 5.3: Comparing different sampling techniques - trained NMOS stack models (All data given here is for minimum samples required to reach constant MSE point)

Model	**Random Sampling**					**Latin-Hypercube Sampling**				
	N_S	T_{model} (S)	MSE	ρ	T_{run} (mS)	N_S	T_{model} (S)	MSE	ρ	T_{run} (mS)
n4	1500	7.12	5.1357e-12	0.9925	0.3795	1200	4.71	4.1770e-12	0.9930	0.3045
n3	1500	6.32	1.0593e-12	0.9967	0.3690	1250	3.54	9.0554e-13	0.9975	0.3190
n2	1450	3.87	1.5680e-12	0.9970	0.3591	1100	2.86	9.3728e-13	0.9972	0.2790
n1	1300	2.71	7.3433e-13	0.9989	0.3193	1050	1.72	5.3456e-13	0.9991	0.2663
n1,f	1300	2.83	2.1215e-13	0.9987	0.3206	1100	1.94	2.0331e-13	0.9991	0.2786
Model	**Adaptive Sampling (Li 2010)**					**Adaptive Sampling (Proposed)**				
	N_S	T_{model} (S)	MSE	ρ	T_{run} (mS)	N_S	T_{model} (S)	MSE	ρ	T_{run} (mS)
n4	1200	106.26	1.1397e-12	0.9995	0.3045	1000	147.89	1.1414e-12	0.9995	0.2545
n3	1100	89.32	2.0549e-13	0.9995	0.2804	850	120.45	2.0798e-13	0.9995	0.2164
n2	1050	82.34	1.5671e-13	0.9996	0.2665	700	112.33	1.5690e-13	0.9996	0.1790
n1	900	70.76	9.3911e-14	0.9998	0.2291	550	82.78	9.4210e-14	0.9998	0.1418
n1,f	1000	73.45	1.0112e-13	0.9996	0.2447	600	88.40	1.0139e-13	0.9996	0.1543

$N_S\rightarrow$ Number of training samples, $T_{model}\rightarrow$ Model Characterization time, $\rho\rightarrow$ Correlation coefficient, $MSE\rightarrow$ Mean Square Error, $T_{run}\rightarrow$ Model Runtime adaptive training method with random sampling, Latin-Hypercube (LHS) sampling in [86] and adaptive sampling in [122]. First column under all sampling methods denotes the number of samples required to reach at constant MSE point. We use 5000 unseen samples to test each model and this process is repeated for 100 times. Average MSE and correlation coefficient in 100 trials is represented in Table 5.3. Our adaptive training method gives same MSE as compared to [122] with less number of training samples. Our methodology reduces the runtime of the stack models, consequently reduces the runtime of the overall methodology. In our modeling method, training time is higher due to training and testing of the models repetitively but the error of models generated using adaptive sampling is very less compared to other sampling methods. Model's training time is higher for higher order stacks and simulation time of stack model is heavily dependent on number of training samples. If different order stack models are trained with same number of training samples then simulation time of models is almost same, which indicates that the simulation time for stack models is almost independent of the order of stack models. MSE between the output from the model and *log* of leakage current is found to be of the order 10^{-6}, while it is of the order 10^{-12} with normal leakage current data. It should also be noted that in Table 5.3, we do not include the information related to PMOS stack models. In case of PMOS stacks, training time and simulation time is approximately of the same order because time factor does not depend upon the type of models, instead it depends only on the number of training samples. In case of PMOS stacks, MSE and ρ are also of the same order as in NMOS stacks. As discussed earlier, after neglecting $(1-e^{-V_{ds}/V_T})$ term in the leakage current equation of a transistor, exponential linear (EL) form as in Equation (5.30) and polynomial equation of 3^{rd} order (poly3) can be used to calculate leakage of a stack.

Figure 5.19 shows the fitted curve of the EL model in the form of a*exp(b*x) used in [4] and poly3 model in the form of a+(b*x)+(c*x^2)+(d*x^3) used in [5] for Figure

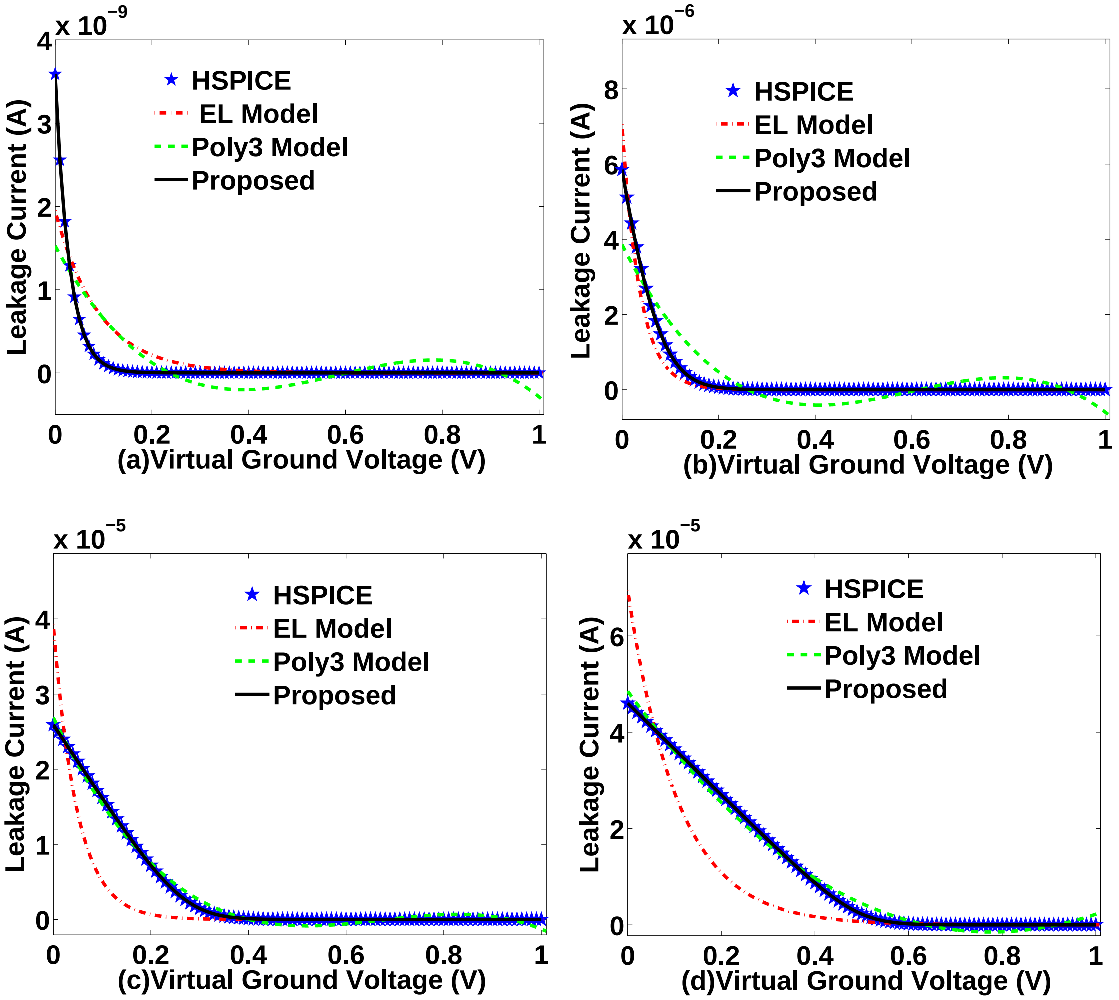


Figure 5.19: EL model [4], Poly3 model [5] and proposed model's comparison for different values of input voltage (Vin1) and virtual ground voltage (V_{gnd}) for single transistor stack in Figure 5.8.(b). (a) Vin1 $= 0\,V$ and $V_{gnd} = 0\,V \rightarrow 1\,V$ (b) Vin1 $=$ $0.4V$ and $V_{gnd} = 0\,V \rightarrow 1\,V$ (c) Vin1 $= 0.7\,V$ and $V_{gnd} = 0\,V \rightarrow 1\,V$ (d) Vin1 $= 1\,V$ and $V_{gnd} = 0\,V \rightarrow 1\,V$

5.8.(b). These curves are obtained by curve fitting toolbox in MATLAB. At Vin1 $= 0\,V$ and V_{gnd} varying from $0\,V$ to $1\,V$, ρ with the SPICE data is 0.7217 for EL model which increases to 0.9486 at Vin1 $= 0.4V$, thereby increasing the accuracy of the model at middle range of input voltage. But on increasing input voltage further, accuracy gets reduced. Correlation coefficient at Vin1 $= 1\,V$ is found to be 0.6353.

Correlation coefficient for poly3 model continuously increases from 0.5779 at Vin1 $= 0\,V$ to 0.9345 at Vin1 $= 1\,V$. As previously described that the input voltage and

V_{gnd} can take any value from $0\,V$ to V_{dd}. Our experiments establish that neither EL model nor poly3 model is able to fit the SPICE data accurately for complete range of Vin1 and V_{gnd}. Thus, we need models that can calculate leakage for whole ranges of input voltage and V_{gnd}. The error incurred using EL model and poly3 model can be larger for higher order stacks shown in Figure 5.8 due to higher number of varying parameters. To develop highly accurate leakage models, SVM can be used to accurately model high dimensional non-linear relationship between input and output parameters. Our experiments justify the use of SVM models over EL and poly3 models. We require very small number of leakage models in comparison to previous work in [4, 5]. If there are M gates and i^{th} gate has k_i inputs, then a total of $\sum_{i=1}^{M} 3^{k_i}$ (3 values for each input i.e. $0\,V$, V_{dd}, V_{gnd}) leakage models are required. The important point here is how to reduce the number of leakage models without sacrificing the accuracy. Our stack models for CMOS gates are a continuous function of input voltages, width of transistor and V_{gnd}. We only require total 9 stack models to calculate the leakage of both CMOS circuits comprising of NAND4, NAND3, NAND2, NOR3, NOR2, INV, AOI22, AOI23, OAI22, OAI32 gates and footer transistor.

Here, we do not vary the V_{th} of the transistors in the logic circuit. Typically, this is the case with all leakage modeling methodologies, i.e. more granularity in model parameter is obtained at increased characterization time. SVM based regression models are more suitable for high dimensional modeling rather than analytical techniques [82]. Adding more parameters of logic circuit like width, threshold voltage only increases the number of samples required to develop final model. It does not change any step of the proposed methodology. However, the adding of parameters to any model depends on which kind of parameters are taken into consideration. Suppose for example, temperature analysis of power gated circuits may require temperature related parameters to be added into the model. However, SVM based models can easily handle this non-linearity with the addition of more dimensions to the model.

Table 5.4: Coefficient values in fitted models of [4] and [5]

[4]		[5]			
a	b	a	b	c	d
1.231e-8	-5.586	1.258e-8	-6.474e-8	1.134e-7	-6.449e-8

Figure 5.20 shows circuit diagram of C17 ISCAS'85 benchmark circuit and its corresponding stack models for input vector '00000'. Equivalent stack models are developed based on the rules described in Section 5.3. Equivalent stack models are only valid for $V_{gnd} < V_p$, hence we compare our results with SPICE results in this range only. Figure 5.21 shows the comparison of leakage current of the logic cluster for different V_{gnd} values using models in [4], [5] and our model. Fitted curve of model in [4] and [5] is obtained in EL and poly3 form using curve fitting toolbox of MATLAB. It can be concluded that our model accurately matches with the SPICE output compared to the large error of model [4] and [5]. Table 5.4 shows the fitting coefficients in the obtained equations for the models in [4] and [5] and Figure 5.21 shows the fitting curve. The error in leakage modeling is higher for lower values of V_{gnd} which is due to neglecting $(1-e^{-V_{ds}/V_T})$ term in the leakage current equation for model in [4], This error is larger for the model in [5] in complete range of V_{gnd} due to truncated representation of the exponential terms with polynomial terms in a predetermined order of the equation.

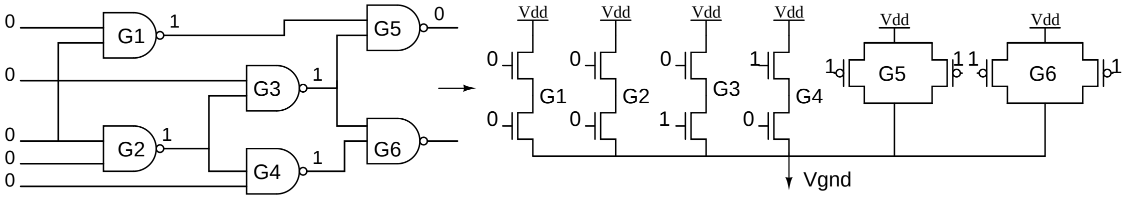


Figure 5.20: C17 circuit diagram and its equivalent stack model representation for input '00000'

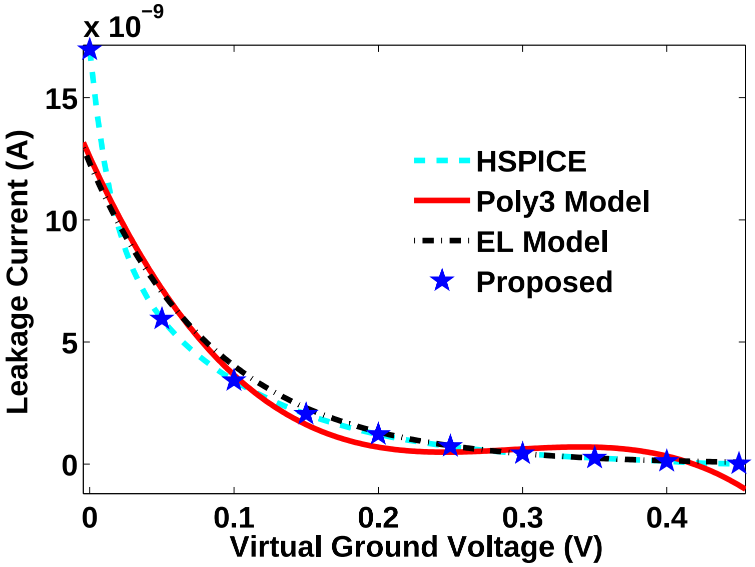


Figure 5.21: Comparing EL model [4], Poly3 model [5] and our method for C17 circuit with input vector '00000'

5.8.2 Gate level Static V_{gnd} Model

In this section, we verify the accuracy and efficiency of our model at gate level. We performed our experiments on AND8 gate [4]. We compare our results with SPICE for input vector '01001000'. Primary gates G1, G2 and G3 are derived from NAND family of gates. According to the rules, PUN is removed because at least one input at logic '0' makes PUN conducting. This connects outputs A, B and C to V_{dd}. Since G4 is a non-primary gate and is from OR family of gates, for $V_{gnd} < V_p$, PDN is 'ON', which is completely opposite to the case of NAND family of gates. Equivalent stack model is presented in Figure 5.22.

Before developing the SVM regression based V_{gnd} model, first we need to check whether SVM classifier is required or not. Maximum V_{gnd} value that can be achieved for this case is evaluated for minimum value of both gate voltage and width of footer transistor. For '01001000' input vector, maximum V_{gnd} is $115mV$, hence no classifier is developed for this case. In general, for small gates, maximum V_{gnd} value in the

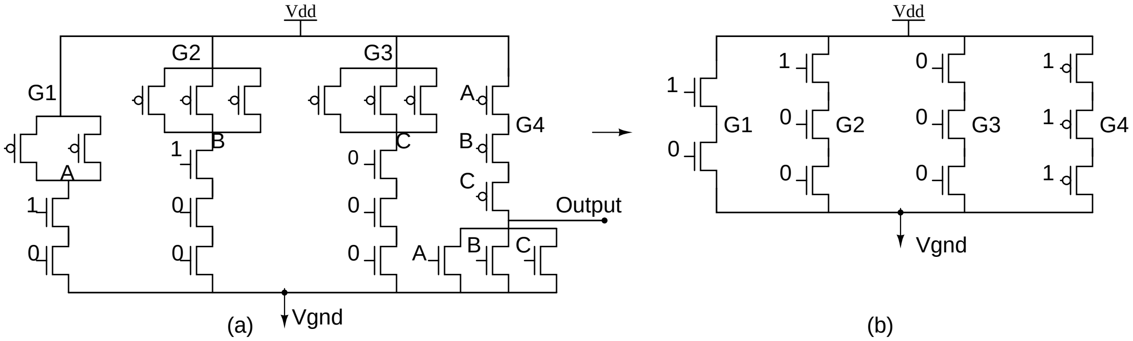


Figure 5.22: (a) AND8 gate (b) Equivalent stack model representation; for input '01001000'

design parameter space is less than V_p. Based on the equivalent stack models, leakage current can be written as in Equation (5.38).

$$I_{leak,AND8} = I_{leak,G1}(n2) + I_{leak,G2}(n3) + I_{leak,G3}(n3) + I_{leak,G4}(p3) \tag{5.38}$$

Equivalent stack model gives MSE of 3.34*10^{-6} and ρ of 0.9998 using proposed SVM model while EL model's MSE and ρ are 0.0651 and 0.416 respectively and model evaluates V_{gnd} value in 0.053 msec. Although, runtime of EL model is less but SVM model completely outperforms the EL model in terms of accuracy. Next, regression data is to be generated for development of final V_{gnd} model. Initially, 100 random samples are generated in the design parameter space to train the SVM model. Then new samples are generated around maximum error sample. Maximum 350 samples have been used to train final V_{gnd} model. For every set of design parameters, algorithm in Figure 5.1 is used to find V_{gnd} value. This algorithm estimates V_{gnd} faster than SPICE simulation which saves our data generation time. Our method takes 4.34 sec. and SPICE simulation will take 11.32 sec. to generate the final V_{gnd} model. Thus, proposed model takes 3× less characterization time. The model is also computationally efficient. Simulation time of our V_{gnd} model is 0.093 msec. while SPICE simulation takes 28.57 msec. Our model is 307× faster than SPICE and accurately matches the results with SPICE as shown in Figure 5.23. We do not compare our results in terms of mean square error of final V_{gnd} model with poly3 model at gate and circuit level because of the following two reasons.

- Since poly3 models are highly inaccurate and out of 3 roots, one root can be real and other two roots are imaginary. But for some cases, real root is not in the range from $0\,V$ to V_{dd}.

- for some cases, all three roots are real and it is very hard to identify the actual V_{gnd} value, discarding the others.

Thus, we are unable to find the actual mean square error using Poly3 model. For runtime comparison, Poly3 model equation is solved in MATLAB. Poly3 model takes 0.063 msec. to calculate one V_{gnd} value, which is higher than EL model and less than SVM model.

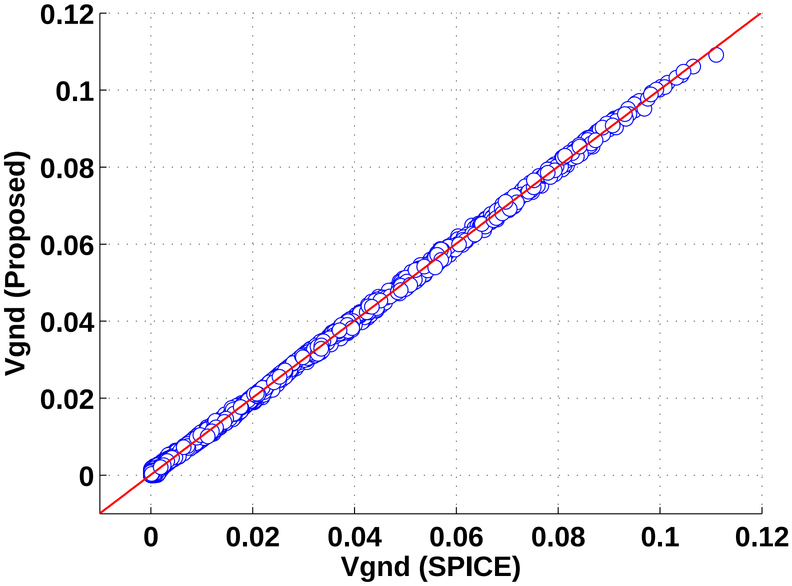


Figure 5.23: Correlation curve between our AND8 gate V_{gnd} model and SPICE; 5000 test samples

5.8.3 Circuit level Static V_{gnd} model

To apply power gating at circuit level, several configurations have been proposed which provides trade-off between parameters such as - area, performance, maximum switching current. A centralized scheme is proposed in [123] to insert footer transistors, which suffers from large interconnect resistance between blocks with large distances. Cluster based approach is proposed to simultaneously reduce dynamic and leakage power [124]. Distributed footer transistor based approach (DSTN) is developed, which is compatible with timing-driven placement as well as reduces area for both footer transistors and wires [125]. To verify circuit level V_{gnd} model, DSTN method is selected. We evaluate our model on ISCAS'85 benchmark circuits and compare simulation results with the SPICE results. Figure 5.24 shows the circuit level DSTN methodology, consisting n number of clusters with n footer transistors. Each footer can be modeled with one current source $I_{footer,n}$. All footer transistors can be modeled as a single current source I_{footer}.

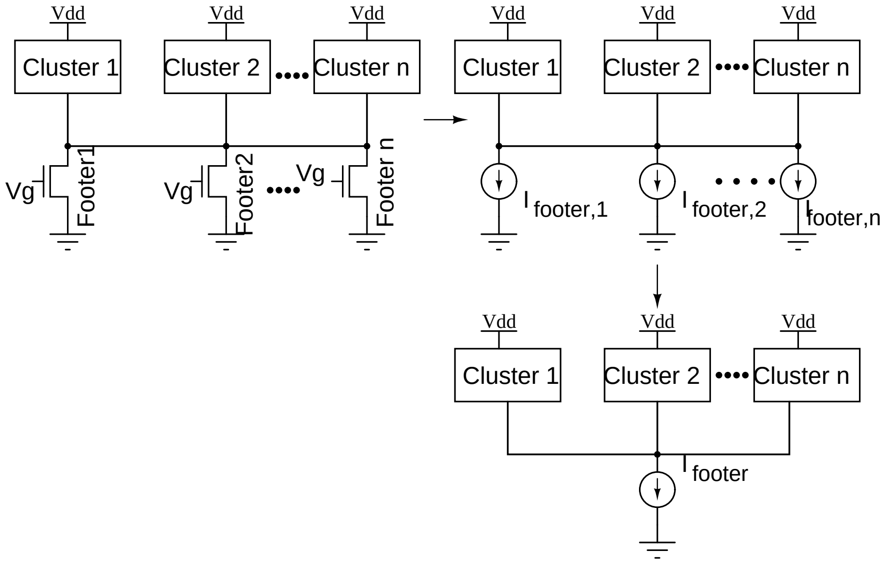


Figure 5.24: Circuit level DSTN method for Static V_{gnd} estimation

We verify the proposed static V_{gnd} model on ISCAS'85 benchmark circuits. Maximum 100 training samples for SVC and 350 samples for SVR V_{gnd} model are used. Table

5.5 and 5.6 show the results of proposed final static V_{gnd} model and EL model [4]. In Figure 5.21, leakage from our model for C17 circuit is very small w.r.t. HSPICE and average error for V_{gnd} model of C17 circuit is 0.14%. For all circuits, error in V_{gnd} voltage is due to cumulative error of leakage, SVC and SVR models. For larger circuits, maximum error is always less than 0.8%, 13× lower than the previous work in [4]. This comparison is based on 5000 test samples across 100 input vectors. Column 2 of Table 5.5 shows time required to develop circuit level V_{gnd} model, combining the time of data generation to develop both SVC and SVR models. Total modeling time of the proposed approach will be the summation of SVC and SVR model generation time. Proposed stack models with SVC help in reducing time to generate the samples for SVR model of V_{gnd}, which significantly reduces total modeling time without degrading the accuracy.

Table 5.5: Circuit level static V_{gnd} model results (Model generation time, Error in V_{gnd} estimation, Model evaluation time) for ISCAS'85 benchmark circuits

Circuit	# Gates	Model generation time			Error (%) for V_{gnd} estimation					
		HSPICE (without SVC)	Proposed (with SVC)		[4]		Proposed			
			SVC	SVR	max	avg	(without SVC)		(with SVC)	
							max	avg	max	avg
C17	6	7.44	3.13	2.57	7.01	5.93	0.17	0.10	0.25	0.14
C432	261	114.21	32.63	14.82	6.29	4.21	0.15	0.10	0.22	0.11
C880	383	140.81	40.23	16.63	10.92	9.42	0.29	0.14	0.43	0.26
C1908	972	231.52	66.15	14.44	6.72	4.57	0.40	0.22	0.55	0.38
C2670	1211	359.11	102.60	23.91	11.89	10.33	0.44	0.27	0.57	0.39
C6288	2351	527.63	125.75	25.7	10.78	9.01	0.72	0.58	0.80	0.73
C7552	3624	766.23	218.92	38.98	13.47	12.27	0.63	0.41	0.75	0.69

SVC and SVR models are trained with different number of samples, therefore runtime of both the models is different. Testing sample in input space are only simulated from final SVR V_{gnd} model. Thus, runtime of the proposed approach is only decided by final SVR V_{gnd} model. From Table 5.6, Our model's runtime is approximately same for all circuits because model's runtime mainly depends on number of training

Table 5.6: Circuit level static V_{gnd} model results (Model evaluation time, Speedup and Error in energy estimation) for ISCAS'85 benchmark circuits

Circuit	# Gates	Model Evaluation time (mS)			Speedup ($\times$)		Error (%) for energy estimation	
		HSPICE	[4]	Proposed	[4]	Proposed	[4]	Proposed
C17	6	24.80	0.053	0.093	468	266	10.29	2.79
C432	261	325.69	0.054	0.093	6031	3502	8.66	2.56
C880	383	401.20	0.055	0.093	7295	4314	13.78	2.95
C1908	972	660.10	0.056	0.091	11787	6948	8.79	3.28
C2670	1211	1025.71	0.052	0.091	19725	11271	15.33	3.36
C6288	2351	1505.67	0.053	0.090	28409	16729	13.13	4.08
C7552	3624	2188.56	0.050	0.095	43771	23037	17.45	3.69

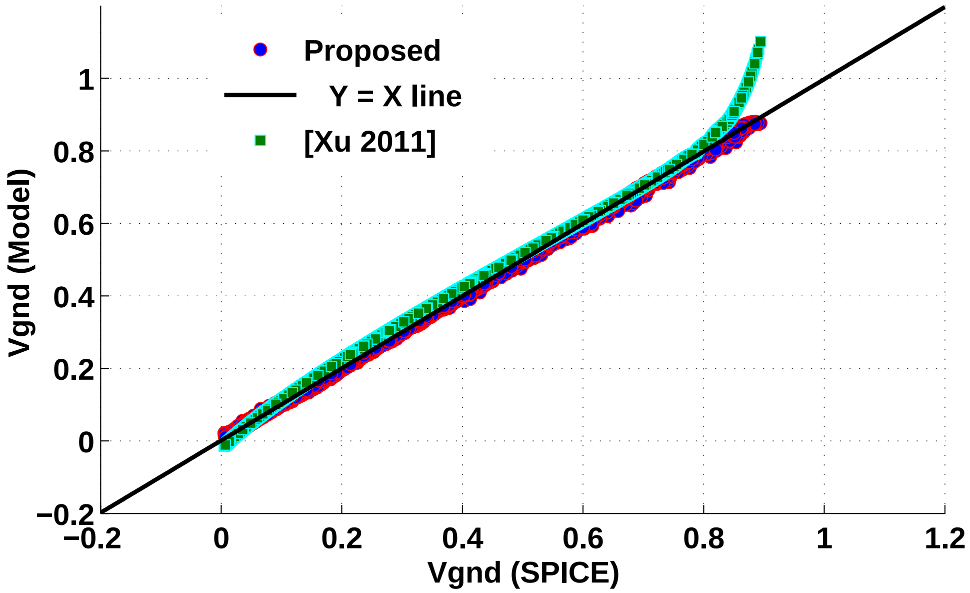


Figure 5.25: Correlation curve of C7552 benchmark circuit V_{gnd} model; 5000 test samples from 100 different input vectors

samples and variables in the training data. Both parameters are same in all circuits. Model has maximum runtime improvement of 23037× on largest benchmark circuit i.e. C7552 circuit, which is likely to be improved on larger circuits. Figure 5.25 shows that V_{gnd} values calculated from our model highly matches with the SPICE results. for EL model, error in V_{gnd} estimation is larger for higher V_{gnd} due to incorrect assumption of applying V_{gnd} to the inputs of CMOS gates in the circuit. Accuracy of the EL model is higher for large circuits but less for small circuits like C17 ISCAS benchmark circuit. Thus, it can be said that EL model's accuracy is highly circuit dependent.In our case, it is circuit size independent. EL model has advantage of better runtime (∼2× improvement) compared to our model, however the accuracy of our model is higher than EL model.

Authors in [87] provided a first order formula to calculate wake-up energy consumption. If total capacitance at V_{gnd} node due to both logic circuit and footer transistors is denoted by C_{ircuit}, then wake-up energy can be estimated as in Equation (5.39):

$$E_{wake-up} = \frac{1}{2} C_{circuit} V_{gnd}^2 \tag{5.39}$$

Accuracy of V_{gnd} model directly affects the accuracy of trade-off analysis between leakage saving in sleep mode and wake-up energy consumption. In our work, we have applied our V_{gnd} model and earlier model to calculate $E_{wake-up}$. Last column in Table 5.5 shows the error in energy estimation. Our model results in average 3.5% error, compared to 13% in previous work. In this work, we are only developing accurate V_{gnd} model, thus error in energy estimation in our case is due to first order energy model and inaccurate capacitance (capacitance will also change with V_{gnd}, can not be defined by a single value).

Since, BSIM4 models in $28nm$ technology are used to relate footer transistor parameters to V_{gnd} voltage. Our proposed model is technology dependent because fitting parameters are extracted using the regression data generated in that technology using SPICE simulations. However, the model development methodology is technology independent and can be used in any technology node with any device model i.e. BSIM4, PSP etc. In more advanced technology nodes than $28nm$, non-linearity related to V_{gnd} model in parameter variation space may be higher, which may require more number of samples than the model in $28nm$ technology node for the same accuracy. If the number of samples are higher, then the runtime of the model will also be higher due to increased number of computations required to evaluate the model. One important advantage of our model is that it does not presume any kind of underlying model like exponential linear or poly3 model which is highly required to increase the accuracy of the model. Our model is a kind of dynamic model whose runtime and accuracy depends upon the training data and has the ability to model highly non-linear performance parameters in the input parameter space.

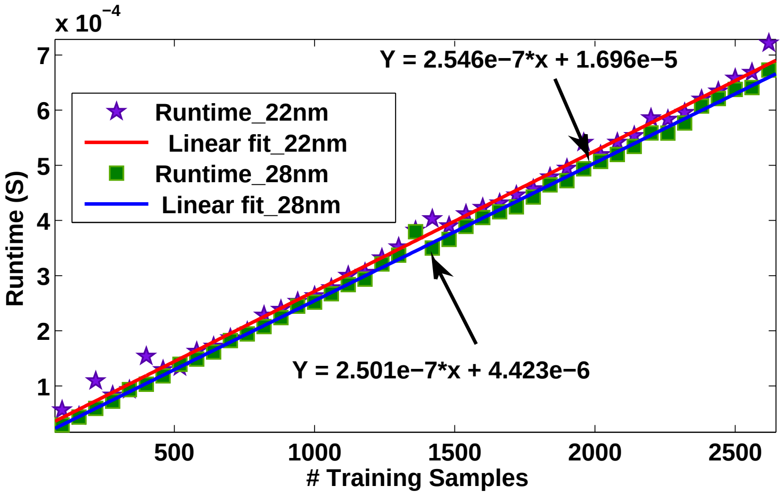


Figure 5.26: Runtime of Static V_{gnd} model in $22nm$ and $28nm$ technologies for varying number of training samples

In the development process of SVM models, we want to use minimum possible number of training samples to reduce the runtime of the models for V_{gnd} estimation. Equation (5.26) and Equation (4.11) are used to evaluate the value of unknown samples in SVC and SVR models respectively. The number of α_k values is directly proportional to the number of training samples. These α_k values affect the computations required to evaluate the newly generated samples (i.e. other than training data).

Figure 5.26 shows the variation in the runtime of static V_{gnd} models in $22nm$ and $28nm$ technologies for increasing value of training samples. Linear curve fitting is used in MATLAB toolbox for both models. Runtime linearly increases with the increasing number of training samples to develop the model. Runtime difference between the models is almost same between both technologies. Runtime complexity for both models is $O(n)$, however, the runtime of $22nm$ model is a little bit higher than $28nm$ model due to the higher value of fitting parameters. Furthermore, in advanced technologies, complexity will also be $O(n)$ but there may be variations in the fitting parameters. The above experiment clearly shows the our SVM model's runtime is mainly dependent on the number of training samples, not on the number of variables and process technology nodes. In the proposed methodology, maximum sampling points is the key for accuracy and efficiency. We have used adaptive sampling based methodology both in classification and regression to iteratively generate the samples for training of the model. Since the dependence of V_{gnd} on footer transistor parameters is same for all circuit, using the same number of samples for all circuits provides us the sufficient accuracy and efficiency of the model. If the complexity of the model is increased by increasing the number of model parameters, then it will require more samples to prepare the model. The proposed methodology is consistent with the generally used regression based analytical equation models [21], which will also require higher number of samples to increased complexity. Thus, there is no specific rule in selecting the maximum samples to train model. However, the adaptive sample generation methodology is more accurate and efficient than fixed samples based methodology.

5.8.4 Effects of SVC model on Accuracy, Runtime and Modeling time of V_{gnd} model

We conducted three set of experiments to analyze the effects of accuracy of Support Vector Classification (SVC) model on accuracy and modeling time of the proposed

V_{gnd} model. For this purpose, we assume 0% to 100% misclassification in the step size of 5%. Misclassification of 0% corresponds to our proposed methodology.

Case (I) is defined as the training samples from $V_{gnd} < V_p$ space are misclassified as $V_{gnd} > V_p$. In this case, more number of training samples need to be simulated through SPICE instead of proposed leakage models to obtain V_{gnd} voltage. It increases the accuracy of V_{gnd} model but at the cost of increased modeling due to the increased use of SPICE simulations. Error reduction and modeling time increment for case (I) can be verified from Figures 5.27.(a) and 5.27.(b).

Case (II) is when the training samples from $V_{gnd} > V_p$ space are misclassified as $V_{gnd} < V_p$. As suggested by SVC model that we should use our leakage models with algorithm in Figure 5.1. But, our leakage models are only valid for $V_{gnd} < V_p$ range and can generate only samples with $V_{gnd} < V_p$ values. To avoid this problem, we initially compare leakage of logic circuit with leakage of footer transistors at $V_{gnd} = V_p$. Evaluation of this condition results in negligible increase in modeling time. If the leakage of logic circuit is higher than footer transistor then we assume that the sample is misclassified and we keep it in $V_{gnd} > V_p$ range and simulate it through SPICE to obtain V_{gnd} voltage. This condition is checked for each training sample. There is a negligible increase in both modeling time and error for case (II), as can be verified from Figures 5.27.(a) and 5.27.(b).

In **Case (III),** training samples from $V_{gnd} < V_p$ space are misclassified as $V_{gnd} > V_p$ and vice-versa. Error and modeling time in this case is approximately equal to case (I) because these parameters in case (II) are same for all misclassification rates, as shown in Figures 5.27.(a) and 5.27.(b).

To clarify the effectiveness of SVC on proposed approach, we use separate sub-columns as without SVC and with SVC under both modeling time and Error column in Table 5.5. Model generation time for proposed approach is also divided into SVC and SVR model generation time. Total modeling time of the proposed approach will be

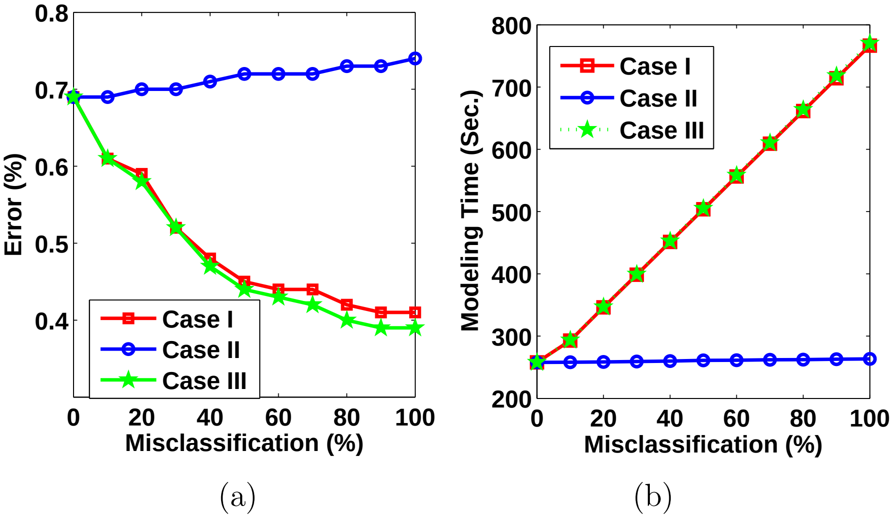


Figure 5.27: Effect of SVC misclassification on (a) Accuracy (b) Modeling time of V_{gnd} model

the summation of SVC and SVR model generation time. Addition of SVC model significantly reduces modeling time without degrading the accuracy.

SVC model is generally given as in Equation (5.40):

$$y_k = sign\left[\sum_{k=1}^{N} \alpha_k K(x_k, x) + b\right] \tag{5.40}$$

Here, $K(x_k, x)$ is the kernel function and α_k and b are solution of the linear systems. Total modeling time of the proposed approach is also affected by runtime of SVC model. If SVC model is trained with higher number of samples than our approach (100 samples), then SVC model generation time and time required to classify the training sample is also high because more number of α_k in Equation (5.40) are used to classify a sample. Figure 5.28 shows the effect of number of training samples used to build the SVC model. Modeling time linearly increases w.r.t. number of training samples. Thus, our overall experiments related to SVC model imply that the runtime and accuracy of the proposed SVC model impacts modeling time without degrading the accuracy of V_{gnd} model.

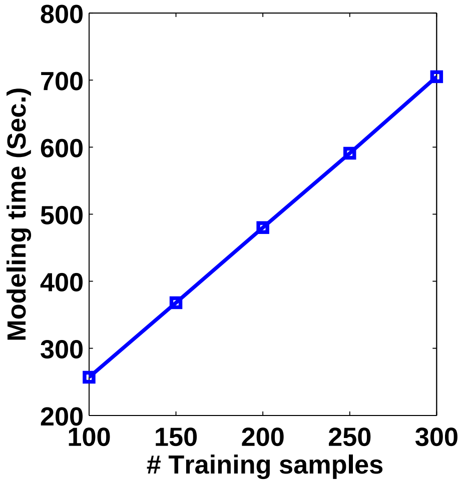


Figure 5.28: Effect of SVC training samples on modeling time of V_{gnd} model

Total modeling time of the proposed approach is also affected by runtime of SVC model. If SVC model is trained with higher number of samples (100 in this work), then SVC model generation time and time required to classify training sample is also high because more number of α_k in Equation (5.40) are used to classify a sample. Figure 5.27.(c) shows the effect of number of SVC model training samples. Modeling time linearly increases w.r.t. number of training samples. Thus, our overall experiments related to SVC model imply that the runtime and accuracy of the proposed SVC model impacts modeling time without degrading the accuracy of V_{gnd} model.

The value of V_p used to build SVC model also affects the accuracy and modeling time of the proposed approach. If $V_p > 0.7\,V$, then less number of SPICE simulation will be required due to the reduction of $V_{gnd} > V_p$ space and maximum samples can be evaluated using proposed leakage models. It will have the advantage of reduced modeling time but there will be negligible reduction in the accuracy due to the use of accurate leakage models in larger $V_{gnd} < V_p$ space. If $V_p < 0.7\,V$, then SPICE simulations will be used for larger number of input samples. It will increase the modeling time but also improve the accuracy of the model.

5.9 Summary

In this Chapter, SVM based regression models are discussed for leakage characterization of transistor stacks in power gated circuits. These surrogate models remove the inaccurate assumption of exponential linear and polynomial functional dependency of leakage current of CMOS gates as a function of gate input voltages and V_{gnd}. This kind of model is more appropriate because the input gate voltages, which are the outputs of preceding gates may take any value from $0\,V$ to V_{dd} for different values of V_{gnd}. To develop SVM models, our methodology uses the accurate value of leakage for all CMOS gates, whether it is primary or non-primary for complete input range varying from $0\,V < V_{gnd} < V_{dd}$. However, previous models had neglected the leakage of the non-primary gates for higher V_{gnd} voltages. Adaptive training sample selection scheme used to prepare SVM models is shown to be more accurate than random sampling and Latin Hypercube Sampling (LHS) based technique. SVM based regression methodology is used to develop static V_{gnd} models for higher accuracy and efficient computation. However, for regression based models, computations required to generate training data using SPICE simulation may take longer time for larger CMOS circuits. We proposed a methodology, which results in substantial saving in model generation time and develop a partition based V_{gnd} data generation algorithm, which uses SPICE simulation for higher values of V_{gnd} and our leakage current models with bisection search algorithm for lower values of V_{gnd}. SVM classifier model is proposed to partition the V_{gnd} values depending on footer transistor parameters. It divides the input parameter space into two partitions. For one partition, SPICE simulation is used whereas for other partition, our leakage current models with bisection search algorithm are used. SVM regression based models are proposed to estimate static V_{gnd} value with high accuracy providing average mean square error of the order 10^{-6} and average correlation coefficient higher than 0.9996. SVM regression based models are proven to be more accurate than analytical models due to use of actual device equations without neglecting any term. SVM surrogate models for leakage current and SVC save 3× simulation time for sample data generation with high accuracy for final static V_{gnd} model. Our static V_{gnd} model is 23000× faster than SPICE for largest C7552 ISCAS'85 benchmark circuit. The proposed methodology is also applicable to other logic styles like Pass-transistor logic (PTL), Dynamic logic, Domino logic.

Chapter 6

Surrogate Models for Dynamic Virtual Ground Voltage Estimation

The characteristics of ground gated circuit during mode transition is shown in Figure 6.1. Voltage at virtual ground node increases from lower value to higher steady state value. The shape and steady state voltages of dynamic V_{gnd} depends on the parameters of footer transistors. In previous Chapter, our aim was to calculate V_{gnd} in steady state mode.

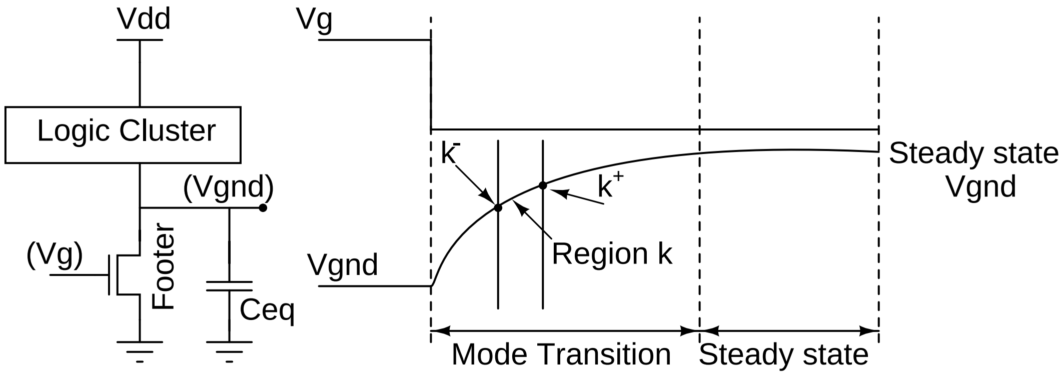


Figure 6.1: Dynamic V_{gnd} in ground gated circuits

In this Chapter, we describe that how effectively our leakage models and static V_{gnd} model in Chapter 5 can be used for dynamic V_{gnd} estimation. Our contributions are as follows.

- SVM based regression models are developed for capacitance estimation at V_{gnd} node i) due to CMOS gates as a function of input voltage and V_{gnd} voltage ii) due to footer transistor as function of input voltage (V_g), width (W_{footer}), threshold voltage ($V_{th,footer}$) and V_{gnd} voltage.

- Piecewise simulation based methodology is developed for dynamic V_{gnd} estimation that efficiently uses the pre-developed leakage and static V_{gnd} models along with extra capacitance models.

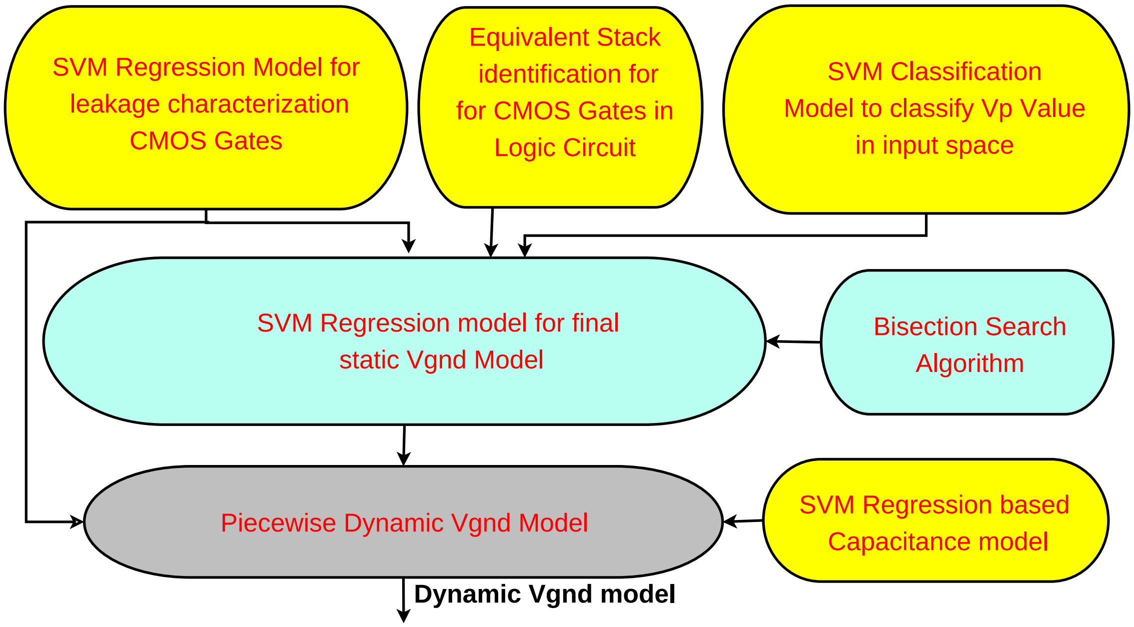

SVM Regression Model for leakage characterization CMOS Gates
Equivalent Stack identification for for CMOS Gates in Logic Circuit
SVM Classification Model to classify Vp Value in input space
SVM Regression model for final static Vgnd Model
Bisection Search Algorithm
Piecewise Dynamic Vgnd Model
SVM Regression based Capacitance model
Dynamic Vgnd model

Figure 6.2: Proposed methodology for dynamic V_{gnd} estimation

Figure 6.2 shows the flowchart describing our proposed methodology for dynamic V_{gnd} estimation. A piecewise simulation based dynamic V_{gnd} estimation methodology is developed which needs the capacitance models to calculate capacitance at virtual ground node along with leakage models and static V_{gnd} model developed in Chapter 5.

6.1 Capacitance Models and Piecewise Simulation Method for Dynamic V_{gnd} Estimation

Virtual ground node capacitance affects the timing characteristics of V_{gnd}, thereby changes the energy consumed in mode transition of power gated circuits. Capacitance at virtual ground node is a strong function of input vector applied to logic circuit and footer transistor parameters. Thus, accurate equivalent capacitance (C_{eq}) as shown in Figure 6.1, due to both footer transistors and CMOS gates in logic circuit needs to be considered at virtual ground node to obtain accurate dynamic V_{gnd} characteristics. Capacitance C_{eq} can be calculated by summing capacitance due to logic cluster (C_{LC}) and footer transistor (C_{footer}). The charging current across C_{eq} can be given as in Equation (6.1).

$$I_{C_{eq}} = C_{eq}\frac{d_{V_{gnd}}}{dt} = I_{LC} - I_{footer} \tag{6.1}$$

Here, I_{LC} and $I_{footter}$ are the leakage current components across logic cluster and footer transistor. To calculate V_{gnd} from 6.1, C_{eq}, I_{LC} and $I_{footter}$ must be modeled in terms of V_{gnd}. I_{LC} and $I_{footter}$ models can be directly imported from Section 5.2 but capacitance models are need to be developed. Since, capacitance at output nodes of CMOS gates affect the virtual ground node capacitance, hence we can not remove the PDN or PUN and C_{eq} is modeled using two types of model i.e. one for CMOS gates and other for footer transistor as shown in Figure 6.3. CMOS gate capacitance C_{gate} at virtual ground node is function of input voltages, Width and V_{gnd} whileC_{footer} is modeled as function of footer transistor parameters (V_g, $V_{th,footer}$, W_{footer}) and V_{gnd}. Total $N+1$ models are required. Where, N represents the number of CMOS gates in a standard cell library composed of basic gates such as NAND2 NAND3, NOR2, NOR3 etc. (Ref. Table 6.1) and CMOS gates consisting parallel or stack of parallel transistors such as AOI22, AOI32, OAI22, OAI23. One model is required for footer transistor. Capacitance models are a continuous function of input voltages like leakage models and hence, cover all input pattern cases which can appear at the input of CMOS gates and footer transistor. Capacitance at V_{gnd} node is a weak non-linear function of input terminal voltages and V_{gnd} voltage and it does not depend on 'ON' or 'OFF' condition of transistors, instead it depends only on the potential difference across two terminals. Thus, we can vary input and V_{gnd} voltages from $0\,V$ to V_{dd} in single model. SVM based regression models are used for capacitance models. Previously, capacitance at any node was assumed as linear function of gate to source voltage V_{gs} [126] but it is not true. In actual, C_{gate} model will be a non-linear function of input voltages, Width and V_{gnd}. C_{footer} will be non-linear function in terms of footer transistor parameters. SVM models are able to capture this non-linearity with high accuracy. Since, SVM models are black-box models and we do not know the equations relating C_{eq}, I_{LC} and $I_{footter}$ in terms of V_{gnd}. So, we develop piecewise simulation based model for dynamic V_{gnd} estimation. As explained in Section 5.3, during mode transition input vector is constant because of standby mode of operation. We derive our methodology in the following steps.

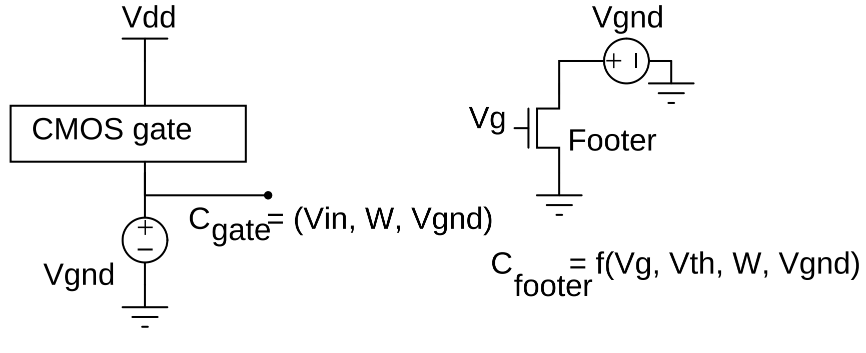


Figure 6.3: Models for capacitance modeling at virtual ground node

Step (1) - Divide V_{gnd} from $0\,V$ to V_p into k regions as shown in Figure 6.1. k determines the trade-off between the runtime and accuracy of the model.

Step (2) - Find Capacitance C_{eq}^{k} at virtual ground node for a given input vector

(i) with N CMOS gates in logic cluster and M footer transistors (Consider DSTN method) at lower (k^-) and higher (k^+) value of region k as shown in Figure 6.1.

$$C_{eq}^{k} = C_{LC}^{k} + C_{footer}^{k} \tag{6.2}$$

Here,

$$C_{LC}^{k} = C_{gate,1}^{k} + C_{gate,2}^{k} + ... + C_{gate,N}^{k} \tag{6.3}$$

$$C_{footer}^{k} = C_{footer,1}^{k} + C_{footer,2}^{k} + ... + C_{footer,M}^{k} \tag{6.4}$$

Calculate average $C_{eq}^{k,avg}$ as in Equation (6.5).

$$C_{eq}^{k,avg} = (C_{eq}^{k^-} + C_{eq}^{k^+})/2 \tag{6.5}$$

Step (3) - Calculate average charging current, $I_{C_{eq}}^{k,avg}$ using Equation (6.6).

$$I_{C_{eq}}^{k,avg} = I_{LC}^{k,avg} - I_{footer}^{k,avg} \tag{6.6}$$

Here,

$$I_{LC}^{k} = I_{gate,1}^{k} + I_{gate,2}^{k} + ... + I_{gate,N}^{k} \tag{6.7}$$

$$I_{LC}^{k,avg} = (I_{LC}^{k^-} + I_{LC}^{k^+})/2 \tag{6.8}$$

$$I_{footer}^{k} = I_{footer,1}^{k} + I_{footer,2}^{k} + ... + I_{footer,M}^{k} \tag{6.9}$$

$$I_{footer}^{k,avg} = (I_{footer}^{k^-} + I_{footer}^{k^+})/2 \tag{6.10}$$

Step (4) - In each region k, we consider a constant leakage across logic cluster and footer transistor. From Equation (6.1), average charging current in region k can be given as follows.

$$I_{C_{eq}}^{k,avg} = C_{eq}^{k,avg}\frac{dV_{gnd}}{dt} = I_{LC}^{k,avg} - I_{footer}^{k,avg} \tag{6.11}$$

$$\frac{V_{gnd} - V_{gnd}}{T^{k^+} - T^{k^-}} = \frac{I_{LC}^{k,avg} - I_{footer}^{k,avg}}{C_{eq}^{k,avg}} \tag{6.12}$$

$$T^{k^+} - T^{k^-} = (V_{gnd} - V_{gnd}) . \frac{C_{eq}^{k,avg}}{I_{LC}^{k,avg} - I_{footer}^{k,avg}} = \alpha \tag{6.13}$$

$$T^{k^+} = T^{k^-} + \alpha \tag{6.14}$$

Equation (6.14) estimates the time duration that circuit takes in charging virtual ground node from V_{gnd} to V_{gnd}. T^{k^-} for first region will be 0 sec. T^{k^+} of each region will be T^{k^-} for the region $k + 1$.

Step (5) - In step (1), we only partition the V_{gnd} range from $0\,V$ to V_p because we do not know about the inputs of CMOS gates in the circuit and hence, we are unable to find capacitances and currents during $V_{gnd} > V_p$. Here, our static V_{gnd} model plays an important role. At $V_{gnd} = V_p$, we know the capacitance and currents at virtual ground node and at steady state V_{gnd}, the current across C_{eq} will be zero. The current across logic cluster will be same as footer transistor. Thus, leakage across logic cluster can be calculated without simulating it. From Figure 6.4, it can be said that the current and the capacitance of logic cluster is linear for $V_p < V_{gnd} < V_{steady,state}$. Using these two points, linear equation is obtained in MATLAB curve fitting toolbox. Correlation coefficient (ρ) of greater than 0.995 is obtained with linear fitting. This linear equation can now be used in V_{gnd} estimation for remaining range of V_{gnd} and steps (1) to (4) are repeated until $V_{steady,state}$ is reached.

Our approach for dynamic V_{gnd} estimation is more accurate than that proposed in [4] due to accurate use of leakage and capacitance models for $V_{gnd} < V_p$ without any assumption and neglecting any term in the device model equations. Linear form of logic cluster current and capacitance is assumed Only for $V_p < V_{gnd} < V_{steady,state}$ is also highly accurate assumption. In [4], a piecewise exponential linear model is developed, in which complete range is divided into regions and linear equations for logic cluster current and footer transistor are obtained through fitting in these regions.

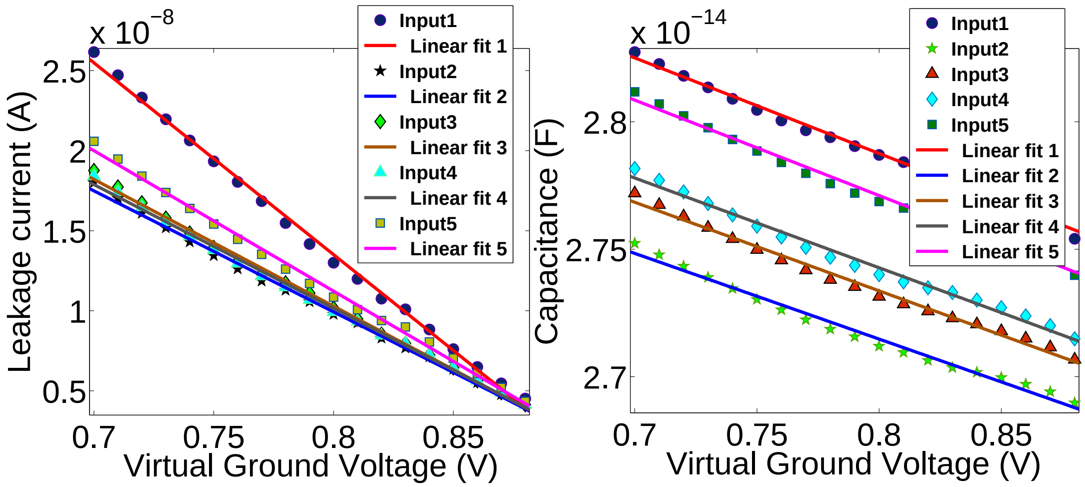


Figure 6.4: Leakage current and capacitance at virtual ground node during $V_p < V_{gnd} < V_{steady,state}$; different input vector combinations of C880 (ISCAS'85)

These fitted equations are used in Equation (6.1) to find out the resulting expressions for V_{gnd} for the considered region. Capacitance variation in the partitioned region is considered as constant, incurs high error in models [4]. While, in our model, we divide it sufficiently in large number of regions such that capacitance can be treated as a constant in each region. Multiple fitting points in each region are considered separately which gives us more accurate capacitance in complete range of V_{gnd}. Another important limitation and source of error of model in [4] is the unknown input voltages of CMOS gates for $V_p < V_{gnd} < V_{steady,state}$ and hence, linear current equations and capacitances can not be evaluated.

6.2 Experimental Results

According to our piecewise simulation based dynamic V_{gnd} estimation methodology, capacitance models are required with leakage and static V_{gnd} model. The virtual ground capacitance is modeled for each CMOS gate and footer transistor. The number of capacitance models to be characterized, depends upon the gates in the cell library used for synthesizing the logic circuit. Figure 6.5 plots the variation in the capacitance of inverter for change in input voltage and V_{gnd} voltage simultaneously. Same adaptive sampling is used for capacitance models as used for leakage models. Maximum of 1000 training samples are used to develop model and another disjoint 5000 samples are used to evaluate the model. Simulated data points lie on the actual curve and hence, verifying the higher accuracy of our model. For higher V_{gnd} and increasing values of input voltage, change in the capacitance is very less and almost linear curve is obtained. But, lower V_{gnd} introduces non-linearity for the same input voltage range. Our SVM models can easily handle this non-linearity with high accuracy. Same results can also be verified from the capacitance variation of NAND2 gate as shown in Figure 6.6.

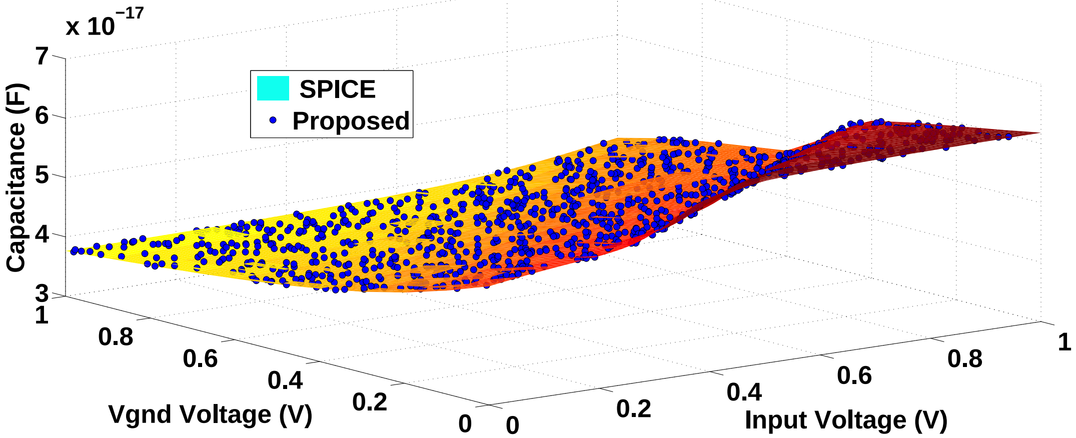


Figure 6.5: Capacitance variation at V_{gnd} node due to Inverter

Table 6.1 shows the error in capacitance modeling for basic CMOS gates of a standard cell library. We consider a CMOS gate with maximum 4 inputs, as higher input gate will have a larger delay due to increased logical effort. Error for OR family of gates is a little bit higher than AND family of gates because of directly connecting the higher number of transistors to the V_{gnd} node. However, capacitance model for each gate has $<$1% average error and maximum error is also between 1% to 2%. Low MSE and high ρ proves the high accuracy of our capacitance models. Since the runtime for

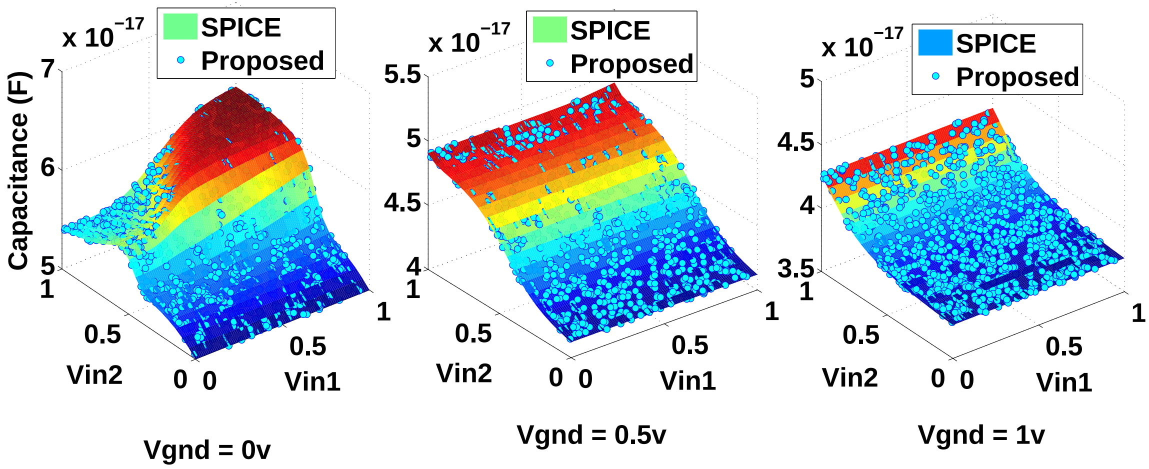


Figure 6.6: Capacitance variation at V_{gnd} node for NAND2 gate for different values of V_{gnd}

all models is of same order describing the less dependence of SVM based models on number of varying inputs. Total 11 models are required to calculate capacitance at V_{gnd} node due to all CMOS gates and footer transistor.

Table 6.1: Evaluating capacitance models; Maximum training samples = 1000, Testing samples = 5000

Gate	*Samples*	*Mean Square Error*	*Correlation Coefficient*	*% Error*		*Model Simulation Time (mS)*
				Max	*Avg*	
NAND2	850	3.11e-37	0.9968	1.05	0.109	0.2162
NAND3	850	3.45e-37	0.9968	1.16	0.314	0.2168
NAND4	900	4.22e-37	0.9961	1.20	0.368	0.2267
NOR2	900	3.23e-37	0.9969	1.05	0.239	0.2269
NOR3	1000	5.14e-37	0.9960	1.26	0.365	0.2560
NOR4	1000	5.20e-37	0.9961	1.29	0.430	0.2564
INV	800	3.04e-37	0.9970	1.02	0.106	0.2089

Dynamic V_{gnd} model is verified on ISCAS'85 benchmark circuits. For each circuit, we consider 100 input vectors and for every input vector, we divide V_{gnd} voltage from $0\,V$ to V_p into 90 regions and 10 regions are chosen for $V_p < V_{gnd} < V_{steady,state}$. for any circuit, if $V_{steady,state} < V_p$ then all 100 points are selected within $0\,V < V_{gnd} < V_p$. We compare our model with the piecewise linear model in [4], where V_{gnd} voltage from $0\,V$ to V_{dd} was divided in 18 regions. Runtime of our SVM model is given as in Equation (6.15).

$$\begin{gathered} T_{ours} = \underbrace{[T_{I,LC} + T_{cap,LC}]_{SVM,90}}_{1^{st}Term} + \underbrace{T_{fit-cap,LC,1}}_{2^{nd}Term} + \underbrace{T_{fit-I,LC,1}}_{3^{nd}Term} + \underbrace{[T_{I,LC} + T_{cap,LC}]_{fit,10}}_{4^{rd}Term} + \ldots\ldots \\ \ldots\ldots + \underbrace{[T_{I,footer} + T_{cap,footer}]_{SVM,100}}_{5^{th}Term} + \underbrace{T_{dyn-vgnd,100}}_{6^{th}Term} \end{gathered} \tag{6.15}$$

Here, 1^{st} term denotes the total runtime of logic cluster for leakage current estimation ($T_{I,LC}$) and capacitance calculation ($T_{cap,LC}$). Runtime is estimated by simulating SVM models 90 times, as 1^{st} term denotes the timing analysis during $0\,V < V_{gnd} < V_p$. $T_{fit-cap,LC}$ and $T_{fit-I,LC}$ determine the runtime to fit the linear model for capacitance and leakage current estimation of logic cluster during $V_p < V_{gnd} < V_{steady,state}$. Only one time fitting is required for each one of two models. 4^{th} term shows the runtime of the fitted models in 2^{nd} and 3^{rd} term and these models are simulated for 10 times. Next, the 5^{th} term evaluates the runtime of leakage and capacitance models for footer transistor and for all 100 regions, SVM models are used. Last term calculates the runtime of the Equation (6.14) for each region. Runtime of the model in [4] can be given as follows.

$$T_{[4]} = [T_{fit-I,LC} + T_{I,LC} + T_{fit-I,footer} + T_{cap,LC} + T_{I,footer} + T_{cap,footer} + T_{dyn-vgnd}]_{18} \tag{6.16}$$

Each term in Equation (6.16) is evaluated 18 times i.e. one time for each region. It should be noted that linear fitting models are used only for leakage current of logic cluster and footer transistor. In each region, capacitance is considered to be a constant value. The maximum and average error with average runtime across all input patters is shown in Table 6.2. Our model has less than 1% error while model in [4] gives approx 5% average error with respect to the SPICE results. Figure 6.7 shows the dynamic V_{gnd} characteristics of C880 ISCAS'85 benchmark circuit after the ground is gated.

This shows the high accuracy of our model when compared to the SPICE output. We have implemented the consumed energy estimation methodology in [4] during mode transition for a dynamic V_{gnd} characteristics of benchmark circuits. Percentage error in energy estimation is given in Table 6.3. Less error in energy estimation shows the higher accuracy of our proposed of dynamic V_{gnd} estimation.

Table 6.2: Circuit level Dynamic V_{gnd} model (ISCAS'85 benchmark circuits)

Circuit	*Error (%)*				*Runtime (sec.)*			*Speedup(×)*	
	Maximum		*Average*		*SPICE*	*[4]*	*Ours*	*[4]*	*Ours*
	[4]	*Ours*	*[4]*	*Ours*					
C17	7.62	0.67	3.66	0.12	12.66	1.08	1.46	12	8
C432	6.64	0.68	5.10	0.11	60.45	1.62	2.56	37	23
C499	7.40	0.79	4.12	0.15	117.89	3.49	5.22	33	22
C880	5.30	1.02	3.66	0.56	72.70	2.73	4.02	26	18
C1355	8.33	1.10	4.83	0.49	110.23	2.90	5.11	38	21
C1908	6.42	0.98	4.95	0.72	136.56	3.89	5.29	35	25
C2670	7.93	0.75	4.36	0.67	143.78	3.45	5.48	41	26
C3540	5.31	1.17	3.22	0.88	162.33	4.32	4.92	37	33
C5315	6.88	1.16	4.37	0.80	201.34	4.11	6.24	49	32
C6288	5.66	0.92	2.92	0.13	228.30	2.01	2.23	113	102
C7552	9.41	1.20	4.99	0.83	300.69	4.65	6.93	64	43
Overall	6.99	0.95	4.20	0.45	140.63	3.11	4.50	45	31

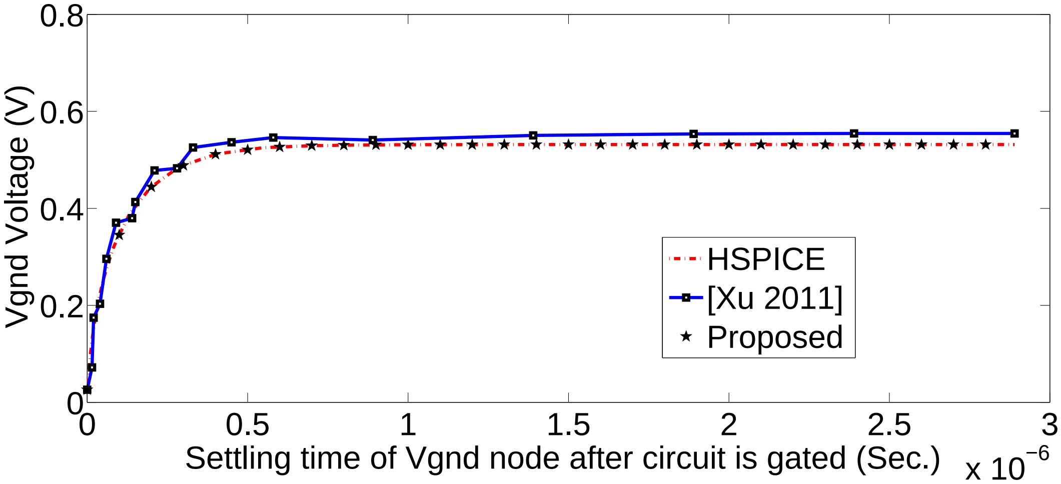


Figure 6.7: Dynamic V_{gnd} voltage during model transition for C880 circuit

Our model is 31× times faster than SPICE but little bit slower than model in [4].

Table 6.3: Evaluating methods for energy estimation during mode transition (ISCAS'85 benchmark circuits)

Circuit	*Error (%)*			
	Maximum		*Average*	
	[4]	*Proposed*	*[4]*	*Proposed*
C17	11.60	1.69	7.65	1.31
C432	9.78	1.68	6.11	1.12
C499	10.46	1.78	6.10	1.15
C880	10.45	2.19	7.54	1.55
C1355	11.01	2.16	6.80	1.50
C1908	8.40	1.79	5.95	0.89
C2670	10.89	1.79	7.34	1.71
C3540	9.10	1.11	6.20	0.98
C5315	10.81	1.16	5.45	1.10
C6288	7.12	0.99	4.21	0.79
C7552	12.48	2.45	7.90	1.96
Overall	10.19	1.71	6.48	1.28

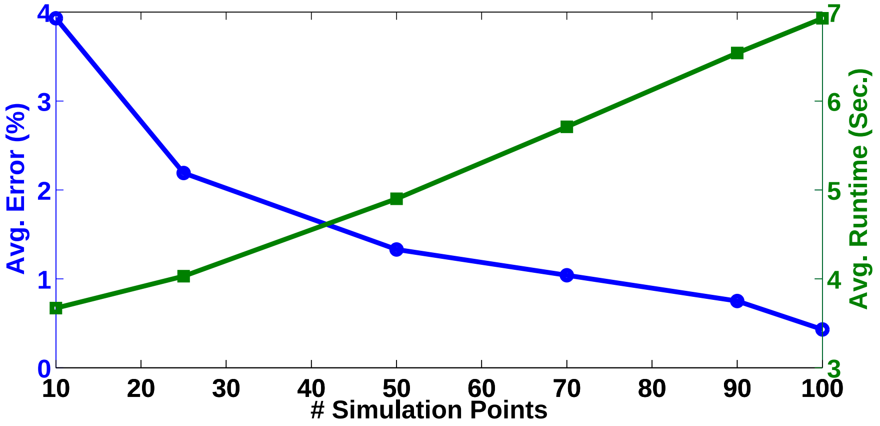


Figure 6.8: Effect of number of simulation points on accuracy and runtime of the model

This is because of the more number of points taken in range from $0\,V$ to $V_{steady,state}$. Maximum runtime improvement is for C6288 benchmark circuit because this circuit consists of maximum NAND2 and NOR2 gates. We require only a few models to be simulated for leakage current estimation of logic cluster and it saves the runtime of complete model. This shows the importance of reducing the number of leakage models in Section 5.2. However, runtime of our models can be further reduced by taking less number of simulation points between $0\,V$ and $V_{steady,state}$. Figure 6.8 plots the accuracy - runtime trade-off with respect to the number of simulation points for C7552 ISCAS'85 benchmark circuit. From Table 6.2, number of simulation points for our models should be less than 43 to make runtime less than in [4]. At this number, the error incurred by our method is ~1.59% which is less than 4.99% of the model proposed in [4].

6.3 Summary

For dynamic V_{gnd} estimation during mode transition, SVM regression models are developed for - (i) capacitance estimation of CMOS gates at virtual ground node as a function of input voltages and V_{gnd} voltage (ii) capacitance estimation of footer transistor at virtual ground node as a function of input sleep voltage, width, threshold voltage and V_{gnd} voltage. Piecewise simulation based model is thus proposed for dynamic V_{gnd} estimation, which efficiently uses the pre-characterized leakage and static V_{gnd} models along with aforesaid extra capacitance models. Our proposed models are highly correlated to the SPICE evaluation with average mean square error of 0.45% on ISCAS'85 benchmark circuits and 31× times faster than SPICE.

Chapter 7

Conclusions and Future Work

Aggressive increase of leakage and process variations in deep sub-micron technologies are a big obstacles in further scaling of the transistors. This leakage is further affected by supply and temperature variations. The effect of these variations on leakage must be accurately analyzed through models such that leakage optimization techniques can be applied in effective manner. Unlike process and environmental variations, input vectors in simple CMOS gate circuits can take only discrete values such as logic high or low. The number of models will be very large if a model is developed for each input vector, thus increasing the model characterization time. Previous models based on scaling the leakage of a single transistor, exponential quadratic models results in increased model characterization time and/or large error in leakage estimation.

In the present work, we have proposed the improved transistor stack models with effective width computation to extract common stack models across 20 gate standard cell library with 176 possible input vector combinations. We have shown that the subthreshold leakage models can be extracted by neglecting 'ON' transistors from the 'OFF' transistor network, only if the subthreshold leakage of that stack lies in the low leakage region. Gate tunneling models are extracted by breaking down the CMOS gate into stacks and single transistors. Thus, only a small subset of models are required for leakage estimation of considered standard cell library.

We have used kernel based SVM for leakage modeling of each stack. SVM models are found to be more accurate than analytical as well as ANN models while capturing highly non-linear functional form of leakage in PVTW space. Various kernels have been explored during SVM model generation for each stack. Optimal tuning parameters of each of the kernels for an SVM model have been obtained through simultaneous use of grid based method and global optimization techniques, resulting in significant less time to find out the best kernel with optimal tuning parameters compared to previous approaches. Active sample selection scheme has been employed to adaptively select training samples to generate SVM models with minimum number of samples, thus resulting in quite smaller simulation time for the model. To further speed-up the model, SVM models have been made Sparse by identifying and excluding the non-significant training samples. Proposed SVM models have been able to predict the leakage with very small error in mean and standard deviation as compared to earlier models proposed in literature. SVM models are shown to be faster than SPICE simulation, however slightly slower than the previous approaches. Proposed SVM models are black box models, thus can be used in full-chip leakage estimation using sampling based methods in place of SPICE simulation. We have also shown that the stack based methodology can be applied to standard cells in post CMOS devices such as FINFET, CNTFET. These SVM models for leakage estimation are illustrated to be applicable even if process parameters take non-Gaussian form of distribution. These models remove the inaccurate log-normal assumption on leakage when process parameters are large in number and vary with generalized distributions. This situation is more likely to appear in future scaled down technologies.

We have also proposed the stack based SVM models for leakage estimation of CMOS gates in power gated circuits. It has been observed that the earlier models in literature are only accurate in a small range of input voltage space. Our proposed leakage models have been developed by considering accurate input voltage conditions for all CMOS gates. These leakage models are then used to develop SVM model for static virtual ground voltage. Support Vector Classification model is used to efficiently generate training data sample with respect to footer transistor parameters, resulting in saving of significant time during static virtual ground model development. To further enhance the accuracy, we have also developed SVM models for capacitance estimation at virtual ground node due to CMOS gates under consideration and footer transistor.

Static virtual ground model, leakage models and capacitance models have been jointly used in proposed piecewise simulation based models for dynamic virtual ground voltage estimation. Proposed methodology has resulted in significantly less error than reported by earlier approaches, Though it is faster than SPICE, but slower than analytical approaches in previous work.

We hope and believe that SVM models can successfully be used in CAD tools for future technologies to accurately model the non-linear effect on different performance parameters of large digital circuits, especially full-chips and SOCs.

Future Work

Since, proposed SVM models can only handle complex CMOS gates consisting parallel or series of parallel transistors. It will be interesting to extend/modify the proposed models for CMOS gates without any restriction on the structure. SVM models accurately capture the non-linear effects but are always slower than the analytical models. Thus, it is highly desirable to increase the simulation speed. In future, methodologies will be required to reduce the runtime of SVM models. Finding out new kernels for accurate and faster SVM models is also a good area of future research. New efficient sampling based full-chip leakage estimation methodologies could be developed, which can use proposed reduced set of models for a given technology and standard cell library. Proposed virtual ground models can be used in exploring trade-off analysis among performance parameters of power gated circuits such as leakage saving, delay overhead, mode transition energy dissipation, reliability effects.

List of Publications out of thesis work

Accepted / Published

- Garg, L.; Sahula, V., "Efficient CMOS subthreshold leakage analysis with improved stack based models in presence of parameter variations," Electronics Letters , vol.49, no.10, pp.644,646, May 9 2013. DOI: 10.1049/el.2012.4311

- Garg, L.; Sahula, V., "Variability aware support vector machine based macromodels for statistical estimation of subthreshold leakage power," Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), 2012 International Conference on , vol., no., pp.253,256, 19-21 Sept. 2012. DOI: 10.1109/SMACD.2012.6339387

- L. Garg and V. Sahula, "Accurate and Efficient Estimation of Dynamic Virtual Ground Voltage in Power Gated Circuits," 2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID), Kolkata, India, 2016, pp. 252-257. doi: 10.1109/VLSID.2016.64.

- L. Garg; V. Sahula, "Macromodels for Static Virtual Ground Voltage Estimation in Power Gated Circuits," in IEEE Transactions on Circuits and Systems II: Express Briefs , vol.PP, no.99, pp.1-1 doi: 10.1109/TCSII.2015.2504270.

Submitted (Under Review)

- Variability Aware Transistor Stack Based Regression Surrogate models for Accurate and Efficient Statistical Leakage Estimation, in Elesevier, Microelectronics Journal, September 2015.

References

[1] S. Sapatnekar, “Overcoming variations in nanometer-scale technologies,” *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on*, vol. 1, no. 1, pp. 5–18, March 2011.

[2] J. Viraraghavan, B. Das, and B. Amrutur, “Voltage and temperature scalable standard cell leakage models based on stacks for statistical leakage characterization,” in *VLSI Design, 2008. VLSID 2008. 21st International Conference on*, jan. 2008, pp. 667 –672.

[3] H. Al-Hertani, D. Al-Khalili, and C. Rozon, “Gate level static power estimation in udsm processes,” in *Microelectronics, 2008. ICM 2008. International Conference on*, dec. 2008, pp. 212 –215.

[4] H. Xu, R. Vemuri, and W.-B. Jone, “Dynamic characteristics of power gating during mode transition,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 19, no. 2, pp. 237–249, 2011.

[5] T. Vivek, O. Sentieys, and S. Derrien, “A semiempirical model for wakeup time estimation in power-gated logic clusters,” in *Design Automation Conference, 2012 49th ACM/EDAC/IEEE*, 2012, pp. 48–55.

[6] D. Lee, D. Blaauw, and D. Sylvester, “Gate oxide leakage current analysis and reduction for VLSI circuits,” vol. 12, no. 2, feb. 2004, pp. 155 –166.

[7] S. Yang, W. Wolf, N. Vijaykrishnan, Y. Xie, and W. Wang, “Accurate stacking effect macro-modeling of leakage power in sub-100 nm circuits,” in *VLSI Design, 2005. 18th International Conference on*, jan. 2005, pp. 165 – 170.

[8] G. Moore, “Cramming more components onto integrated circuits,” *Proceedings of the IEEE*, vol. 86, no. 1, pp. 82–85, Jan 1998.

[9] S. Borkar, “Design challenges of technology scaling,” *Micro, IEEE*, vol. 19, no. 4, pp. 23–29, Jul 1999.

[10] B. Chatterjee, M. Sachdev, S. Hsu, R. Krishnamurthy, and S. Borkar, “Effectiveness and scaling trends of leakage control techniques for sub-130 nm cmos technologies,” in *Low Power Electronics and Design, 2003. ISLPED ’03. Proceedings of the 2003 International Symposium on*, Aug 2003, pp. 122–127.

[11] S. G. Narendra and A. Chandrakasan, *Leakage in Nanometer CMOS Technologies.* New York, NY: Springer, 2005. [Online]. Available: https://cds.cern.ch/record/1084684

[12] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand, “Leakage current mechanisms and leakage reduction techniques in deep-submicrometer cmos circuits,” *Proceedings of the IEEE*, vol. 91, no. 2, pp. 305–327, Feb 2003.

[13] Y. Taur and T. H. Ning, *Fundamentals of Modern VLSI Devices*, 2nd ed. New York, NY, USA: Cambridge University Press, 2009.

[14] W. Henson, N. Yang, S. Kubicek, E. M. Vogel, J. J. Wortman, K. De Meyer, and A. Naem, “Analysis of leakage currents and impact on off-state power consumption for cmos technology in the 100-nm regime,” *Electron Devices, IEEE Transactions on*, vol. 47, no. 7, pp. 1393–1400, Jul 2000.

[15] B. Streetman and S. Banerjee, *Solid State Electronic Devices*, ser. Prentice-Hall series in solid state physical electronics. Pearson Prentice Hall, 2006. [Online]. Available: https://books.google.co.in/books?id=xHS8QgAACAAJ

[16] A. P. Chandrakasan, W. J. Bowhill, and F. Fox, *Design of High-Performance Microprocessor Circuits*, 1st ed. Wiley-IEEE Press, 2000.

[17] H. Chang and S. Sapatnekar, "Statistical timing analysis considering spatial correlations using a single pert-like traversal," in *Computer Aided Design, 2003. ICCAD-2003. International Conference on*, Nov 2003, pp. 621–625.

[18] A. Agarwal, D. Blaauw, and V. Zolotov, "Statistical timing analysis for intra-die process variations with spatial correlations," in *Computer Aided Design, 2003. ICCAD-2003. International Conference on*, Nov 2003, pp. 900–907.

[19] S. Ozdemir, D. Sinha, G. Memik, J. Adams, and H. Zhou, "Yield-aware cache architectures," in *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO 39. Washington, DC, USA: IEEE Computer Society, 2006, pp. 15–25. [Online]. Available: http://dx.doi.org/10.1109/MICRO.2006.52

[20] S. Duvall, "Statistical circuit modeling and optimization," in *Statistical Metrology, 2000 5th International Workshop on*, 2000, pp. 56–63.

[21] H. Chang and S. S. Sapatnekar, "Prediction of leakage power under process uncertainties," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 12, no. 2, Apr. 2007.

[22] G. Roy, A. Brown, F. Adamu-Lema, S. Roy, and A. Asenov, "Simulation study of individual and combined sources of intrinsic parameter fluctuations in conventional nano-mosfets," *Electron Devices, IEEE Transactions on*, vol. 53, no. 12, pp. 3063–3070, Dec 2006.

[23] W. Liu, X. Jin, J. Chen, M.-C. Jeng, Z. Liu, Y. Cheng, K. Chen, M. Chan, K. Hui, J. Huang, R. Tu, P. Ko, and C. Hu, "Bsim3v3.2.2 mosfet model users' manual," EECS Department, University of California, Berkeley, Tech. Rep. UCB/ERL M99/18, 1999. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/1999/3616.html

[24] H. Su, F. Liu, A. Devgan, E. Acar, and S. Nassif, "Full chip leakage-estimation considering power supply and temperature variations," in *Low Power Electronics and Design, 2003. ISLPED '03. Proceedings of the 2003 International Symposium on*, Aug 2003, pp. 78–83.

[25] R. Rao, A. Srivastava, D. Blaauw, and D. Sylvester, "Statistical estimation of leakage current considering inter- and intra-die process variation," in *Low Power Electronics and Design, 2003. ISLPED '03. Proceedings of the 2003 International Symposium on*, Aug 2003, pp. 84–89.

[26] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, "Parameter variations and impact on circuits and microarchitecture," in *Design Automation Conference, 2003. Proceedings*, June 2003, pp. 338–342.

[27] D. Boolchandani, L. Garg, S. Khandelwal, and V. Sahula, "Variability aware svm macromodel based design centering of analog circuits," *Analog Integrated Circuits and Signal Processing*, vol. 73, no. 1, pp. 77–87, 2012. [Online]. Available: http://dx.doi.org/10.1007/s10470-011-9799-y

[28] A. Gupta, N. Dutt, F. Kurdahi, K. Khouri, and M. Abadir, "Stefal: A system level temperature- and floorplan-aware leakage power estimator for socs," in *VLSI Design, 2007. Held jointly with 6th International Conference on Embedded Systems., 20th International Conference on*, Jan 2007, pp. 559–564.

[29] R. Gu and M. Elmasry, “Power dissipation analysis and optimization of deep submicron CMOS digital circuits,” *Solid-State Circuits, IEEE Journal of*, vol. 31, no. 5, pp. 707 –713, may 1996.

[30] Z. Chen, M. Johnson, L. Wei, and W. Roy, “Estimation of standby leakage power in CMOS circuit considering accurate modeling of transistor stacks,” in *Low Power Electronics and Design, 1998. Proceedings. 1998 International Symposium on*, aug. 1998, pp. 239 –244.

[31] R. Rao, J. Burns, A. Devgan, and R. Brown, “Efficient techniques for gate leakage estimation,” in *Low Power Electronics and Design, 2003. ISLPED ’03. Proceedings of the 2003 International Symposium on*, aug. 2003, pp. 100 – 103.

[32] S. Mukhopadhyay, A. Raychowdhury, and K. Roy, “Accurate estimation of total leakage current in scaled CMOS logic circuits based on compact current modeling,” in *Design Automation Conference, 2003. Proceedings*, june 2003, pp. 169 – 174.

[33] M. Dunga, W. Yang, and X. e. a. Xi, *BSIM4.6.1 MOSFET model, users manual*, University of California, Berkeley, 2007.

[34] R. Shen, S. X.-D. Tan, H. Wang, and J. Xiong, “Fast statistical full-chip leakage analysis for nanometer VLSI systems,” *ACM Trans. Des. Autom. Electron. Syst.*, vol. 17, no. 4, pp. 51:1–51:19, Oct. 2012.

[35] K. Heloue, N. Azizi, and F. Najm, “Full-chip model for leakage-current estimation considering within-die correlation,” *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 28, no. 6, pp. 874–887, June 2009.

[36] S. Bhardwaj and S. Vrudhula, “A fast and accurate approach for full chip leakage analysis of nano-scale circuits considering intra-die correlations,” in *VLSI*

Design, 2007. Held jointly with 6th International Conference on Embedded Systems., 20th International Conference on, Jan 2007, pp. 589–594.

[37] J. Kim, W. Kim, and Y.-H. Kim, "Evaluation of the state-of-the art statistical leakage estimation methods using the bsim4 transistor model," in *Integrated Circuits, ISIC '09. Proceedings of the 2009 12th International Symposium on*, 2009, pp. 409–412.

[38] N. Beaulieu, A. Abu-Dayya, and P. McLane, "Comparison of methods of computing lognormal sum distributions and outages for digital wireless applications," in *Communications, 1994. ICC '94, SUPERCOMM/ICC '94, Conference Record, 'Serving Humanity Through Communications.' IEEE International Conference on*, 1994, pp. 1270–1275 vol.3.

[39] A. Srivastava, S. Shah, K. Agarwal, D. Sylvester, D. Blaauw, and S. Director, "Accurate and efficient gate-level parametric yield estimation considering correlated variations in leakage power and performance," in *Design Automation Conference, 2005. Proceedings. 42nd*, 2005, pp. 535–540.

[40] W. Liu, X. Jin, and X. e. a. Xi, *Users Manual, BSIM3v3. 3 MOSFET model*, University of California, Berkeley, 2005.

[41] H. Agarwal, S. Venugopalan, M.-A. Chalkiadaki, N. Paydavosi, J. Duarte, S. Agnihotri, C. Yadav, P. Kushwaha, Y. Chauhan, C. Enz, A. Niknejad, and C. Hu, "Recent enhancements in bsim6 bulk mosfet model," in *Simulation of Semiconductor Processes and Devices (SISPAD), 2013 International Conference on*, Sept 2013, pp. 53–56.

[42] H. Su, F. Liu, A. Devgan, E. Acar, and S. Nassif, "Full chip leakage-estimation considering power supply and temperature variations," in *Low Power Electron-*

ics and Design, 2003. ISLPED '03. Proceedings of the 2003 International Symposium on, Aug 2003, pp. 78–83.

[43] J. Jaffari and M. Anis, "Statistical thermal profile considering process variations: Analysis and applications," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 27, no. 6, pp. 1027 –1040, june 2008.

[44] Z. Abbas, A. Mastrandrea, and M. Olivieri, "A voltage-based leakage current calculation scheme and its application to nanoscale MOSFET and FINFET standard-cell designs," pp. 2549–2560, Dec 2014.

[45] R. Y. Rubinstein, *Simulation and the Monte Carlo Method*, 1st ed. New York, NY, USA: John Wiley & Sons, Inc., 1981.

[46] I. Sobol, "On the distribution of points in a cube and the approximate evaluation of integrals," *USSR Computational Mathematics and Mathematical Physics*, vol. 7, no. 4, pp. 86 – 112, 1967.

[47] V. Veetil, D. Sylvester, D. Blaauw, S. Shah, and S. Rochel, "Efficient smart sampling based full-chip leakage analysis for intra-die variation considering state dependence," in *Design Automation Conference, 2009. DAC '09. 46th ACM/IEEE*, 2009, pp. 154–159.

[48] R. Guindi and F. Najm, "Design techniques for gate-leakage reduction in CMOS circuits," in *Quality Electronic Design, 2003. Proceedings. Fourth International Symposium on*, march 2003, pp. 61 – 65.

[49] L. Cheng, P. Gupta, and L. He, "Efficient additive statistical leakage estimation," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 28, no. 11, pp. 1777–1781, 2009.

[50] O. Garitselov, S. P. Mohanty, and E. Kougianos, “A comparative study of metamodels for fast and accurate simulation of nano-cmos circuits,” *IEEE Transactions on Semiconductor Manufacturing*, vol. 25, no. 1, pp. 26–36, Feb 2012.

[51] S. P. Mohanty and E. Kougianos, “Polynomial metamodel based fast optimization of nano-cmos oscillator circuits,” *Analog Integrated Circuits and Signal Processing*, vol. 79, no. 3, pp. 437–453, 2014.

[52] S. P. Mohanty, E. Kougianos, O. Garitselov, and J. M. Molina, “Polynomial-metamodel assisted fast power optimization of nano-cmos pll components,” in *Specification and Design Languages (FDL), 2012 Forum on*, Sept 2012, pp. 233–238.

[53] J. Wang, X. Li, and L. T. Pileggi, “Parameterized macromodeling for analog system-level design exploration,” in *Design Automation Conference, 2007. DAC ’07. 44th ACM/IEEE*, June 2007, pp. 940–943.

[54] L. Garg and V. Sahula, “Variability aware support vector machine based macromodels for statistical estimation of subthreshold leakage power,” in *Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), 2012 International Conference on*, 2012, pp. 253–256.

[55] G. Zheng, S. P. Mohanty, E. Kougianos, and O. Okobiah, “ivams: Intelligent metamodel-integrated verilog-ams for circuit-accurate system-level mixed-signal design exploration,” in *Application-Specific Systems, Architectures and Processors (ASAP), 2013 IEEE 24th International Conference on*, June 2013, pp. 75–78.

[56] R. Samanta, J. Hu, and P. Li, “Discrete buffer and wire sizing for link-based non-tree clock networks,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 7, pp. 1025–1035, July 2010.

[57] A. Lamecki, L. Balewski, and M. Mrozowski, "Towards automated full-wave design of microwave circuits," in *Microwaves, Radar and Wireless Communications, 2008. MIKON 2008. 17th International Conference on*, May 2008, pp. 1–2.

[58] J. L. Wong, A. Davoodi, V. Khandelwal, A. Srivastava, and M. Potkonjak, "A statistical methodology for wire-length prediction," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 7, pp. 1327–1336, July 2006.

[59] B. Liu, D. Zhao, P. Reynaert, and G. G. E. Gielen, "Synthesis of integrated passive components for high-frequency rf ics based on evolutionary computation and machine learning techniques," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 10, pp. 1458–1468, Oct 2011.

[60] B. Liu, N. Deferm, D. Zhao, P. Reynaert, and G. G. E. Gielen, "An efficient high-frequency linear rf amplifier synthesis method based on evolutionary computation and machine learning techniques," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 7, pp. 981–993, July 2012.

[61] A. Singhee and R. A. Rutenbar, "Statistical blockade: Very fast statistical simulation and modeling of rare circuit events and its application to memory design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 8, pp. 1176–1189, Aug 2009.

[62] H. Shin, S. X. D. Tan, G. Shi, and E. Tlelo-Cuautle, "Rare event diagnosis by iterative failure region locating and elite learning sample selection," in *Test Symposium (LATS), 2015 16th Latin-American*, March 2015, pp. 1–5.

[63] S. Sun, X. Li, H. Liu, K. Luo, and B. Gu, “Fast statistical analysis of rare circuit failure events via scaled-sigma sampling for high-dimensional variation space,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 7, pp. 1096–1109, July 2015.

[64] G. Wolfe and R. Vemuri, “Extraction and use of neural network models in automated synthesis of operational amplifiers,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 2, pp. 198–212, Feb 2003.

[65] L. Wang, *Support Vector Machines: Theory and Applications*, ser. Studies in Fuzziness and Soft Computing. Heidelberg, Germany: Springer Berlin, 2005, vol. 177.

[66] V. N. Vapnik, *Statistical learning theory*, 1st ed. Wiley, 1998.

[67] J. Suykens and J. Vandewalle, “Least squares support vector machine classifiers,” *Neural Processing Letters*, vol. 9, no. 3, pp. 293–300, 1999.

[68] G. Rubio, H. Pomares, I. Rojas, and L. J. Herrera, “A heuristic method for parameter selection in ls-svm: Application to time series prediction,” *International Journal of Forecasting*, vol. 27, no. 3, pp. 725 – 739, 2011.

[69] S. Pandit, C. Mandal, and A. Patra, “Systematic methodology for high-level performance modeling of analog systems,” in *VLSI Design, 2009 22nd International Conference on*, jan. 2009, pp. 361 –366.

[70] J. A. K. Suykens, T. V. Gestel, J. D. Brabanter, B. D. Moor, and J. Vandewalle, *Least Squares Support Vector Machines*, 2002.

[71] S. Pandit, C. Mandal, and A. Patra, “A methodology for generation of performance models for the sizing of analog high-level topologies,” *VLSI Des.*, vol. 2011, pp. 4:1–4:17, Jan. 2011.

[72] R. Storn and K. Price, “Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces,” *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[73] B. Settles, “Active learning literature survey,” University of Wisconsin–Madison, Tech. Rep., 2010.

[74] S. Tong and D. Koller, “Support vector machine active learning with applications to text classification,” *The Journal of Machine Learning Research*, vol. 2, pp. 45–66, Mar. 2002.

[75] N. Roy and A. McCallum, “Toward optimal active learning through sampling estimation of error reduction,” in *ICML*, 2001, pp. 441–448.

[76] E. Maricau, D. De Jonghe, and G. Gielen, “Hierarchical analog circuit reliability analysis using multivariate nonlinear regression and active learning sample selection,” in *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, march 2012, pp. 745 –750.

[77] J. Valyon and G. Horvath, “A sparse least squares support vector machine classifier,” in *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, vol. 1, july 2004, pp. 4 vol. (xlvii+3302).

[78] J. Wang, W. Wu, N. Gong, L. Zuo, L. Hou, S. Geng, W. Zhang, and D. Gao, “Switching and leakage power modeling for multiple-supply dynamic gate with delay constraining based on wavelet neural networks,” in *Neural Networks, 2009. International Joint Conference on*, June 2009, pp. 3240–3243.

[79] X. Dong, N. P. Jouppi, and Y. Xie, "A circuit-architecture co-optimization framework for exploring nonvolatile memory hierarchies," *ACM Trans. Archit. Code Optim.*, vol. 10, no. 4, pp. 23:1–23:22, Dec. 2013. [Online]. Available: http://doi.acm.org/10.1145/2541228.2541230

[80] F. Ye, Z. Zhang, K. Chakrabarty, and X. Gu, "Board-level functional fault diagnosis using artificial neural networks, support-vector machines, and weighted-majority voting," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 32, no. 5, pp. 723–736, May 2013.

[81] ——, "Board-level functional fault diagnosis using multikernel support vector machines and incremental learning," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 33, no. 2, pp. 279–290, Feb 2014.

[82] A. B. Kahng, B. Lin, and S. Nath, "Enhanced metamodeling techniques for high-dimensional ic design estimation problems," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2013*, March 2013, pp. 1861–1866.

[83] K.-K. Kim, Y.-B. Kim, and K. Choi, "Hybrid CMOS and CNFET power gating in ultralow voltage design," *Nanotechnology, IEEE Transactions on*, vol. 10, no. 6, pp. 1439–1448, Nov 2011.

[84] M. B. Henry and L. Nazhandali, "From transistors to nems: Highly efficient power-gating of CMOS circuits," *J. Emerg. Technol. Comput. Syst.*, vol. 8, no. 1, pp. 2:1–2:18, Feb. 2012. [Online]. Available: http://doi.acm.org/10.1145/2093145.2093147

[85] V. Janakiraman, A. Bharadwaj, and V. Visvanathan, "Voltage and temperature aware statistical leakage analysis framework using artificial neural networks,"

Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol. 29, no. 7, pp. 1056–1069, July 2010.

[86] O. Garitselov, S. Mohanty, and E. Kougianos, "A comparative study of metamodels for fast and accurate simulation of nano-CMOS circuits," *Semiconductor Manufacturing, IEEE Transactions on*, vol. 25, no. 1, pp. 26–36, Feb 2012.

[87] H. Singh, K. Agarwal, D. Sylvester, and K. Nowka, "Enhanced leakage reduction techniques using intermediate strength power gating," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 15, no. 11, pp. 1215–1224, 2007.

[88] S. Kim, S. Kosonocky, D. Knebel, K. Stawiasz, and M. Papaefthymiou, "A multi-mode power gating structure for low-voltage deep-submicron CMOS ics," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 54, no. 7, pp. 586–590, 2007.

[89] A. Sinkar, T. Park, and N. S. Kim, "Clamping virtual supply voltage of power-gated circuits for active leakage reduction and gate-oxide reliability," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 21, no. 3, pp. 580–584, 2013.

[90] T. Vivek, O. Sentieys, and S. Derrien, "Wakeup time and wakeup energy estimation in power-gated logic clusters," in *VLSI Design (VLSI Design), 2011 24th International Conference on*, 2011, pp. 340–345.

[91] V. D. Tovinakere, O. Sentieys, and S. Derrien, "A polynomial based approach to wakeup time and energy estimation in power-gated logic clusters," *Journal of Low Power Electronics*, vol. 7, no. 4, pp. 482–489, 2011-12-01T00:00:00.

[92] L. Yuan and G. Qu, "A combined gate replacement and input vector control approach for leakage current reduction," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 14, no. 2, pp. 173–182, 2006.

[93] N. Jayakumar and S. Khatri, "An algorithm to minimize leakage through simultaneous input vector control and circuit modification," in *Design, Automation Test in Europe Conference Exhibition, 2007. DATE '07*, 2007, pp. 1–6.

[94] S. Pendyala and S. Katkoori, "Interval arithmetic based input vector control for rtl subthreshold leakage minimization," in *VLSI and System-on-Chip (VLSI-SoC), 2012 IEEE/IFIP 20th International Conference on*, 2012, pp. 141–146.

[95] A. Sathanur, A. Pullini, L. Benini, A. Macii, E. Macii, and M. Poncino, "Optimal sleep transistor synthesis under timing and area constraints," in *Proceedings of the 18th ACM Great Lakes Symposium on VLSI*, ser. GLSVLSI '08. New York, NY, USA: ACM, 2008, pp. 177–182.

[96] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand, "Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits," *Proceedings of the IEEE*, vol. 91, no. 2, pp. 305–327, 2003.

[97] J.-H. Choi, Y. Xu, and T. Sakurai, "Statistical leakage current reduction in high-leakage environments using locality of block activation in time domain," *Solid-State Circuits, IEEE Journal of*, vol. 39, no. 9, pp. 1497–1503, Sept 2004.

[98] B. Yu and M. Bushnell, "A novel dynamic power cutoff technique (dpct) for active leakage reduction in deep submicron CMOS circuits," in *Low Power Electronics and Design, 2006. ISLPED'06. Proceedings of the 2006 International Symposium on*, Oct 2006, pp. 214–219.

[99] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson, and P. Bose, "Microarchitectural techniques for power gating of execution units," in *Low*

Power Electronics and Design, 2004. ISLPED '04. Proceedings of the 2004 International Symposium on, Aug 2004, pp. 32–37.

[100] K. Usami and N. Ohkubo, "A design approach for fine-grained run-time power gating using locally extracted sleep signals," in *Computer Design, 2006. ICCD 2006. International Conference on*, Oct 2006, pp. 155–161.

[101] S. P. Mohanty and E. Kougianos, "Modeling and reduction of gate leakage during behavioral synthesis of nanocmos circuits," in *VLSI Design, 2006. Held jointly with 5th International Conference on Embedded Systems and Design., 19th International Conference on*, Jan 2006, pp. 6 pp.–.

[102] V. Mukherjee, S. P. Mohanty, and E. Kougianos, "A dual dielectric approach for performance aware gate tunneling reduction in combinational circuits," in *Computer Design: VLSI in Computers and Processors, 2005. ICCD 2005. Proceedings. 2005 IEEE International Conference on*, Oct 2005, pp. 431–436.

[103] S. P. Mohanty, R. Velagapudi, and E. Kougianos, "Physical-aware simulated annealing optimization of gate leakage in nanoscale datapath circuits," in *Design, Automation and Test in Europe, 2006. DATE '06. Proceedings*, vol. 1, March 2006, pp. 6 pp.–.

[104] Y.-S. Lin and D. Sylvester, "Runtime leakage power estimation technique for combinational circuits," in *Design Automation Conference, 2007. ASP-DAC '07. Asia and South Pacific*, 2007, pp. 660–665.

[105] R. Rao, A. Srivastava, D. Blaauw, and D. Sylvester, "Statistical analysis of subthreshold leakage current for vlsi circuits," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 12, no. 2, pp. 131–139, 2004.

[106] P. B. Nair, A. Choudhury, and A. J. Keane, “Some greedy learning algorithms for sparse regression and classification with mercer kernels,” *J. Mach. Learn. Res.*, vol. 3, pp. 781–801, Mar. 2003.

[107] J. Tropp and A. Gilbert, “Signal recovery from random measurements via orthogonal matching pursuit,” *Information Theory, IEEE Transactions on*, vol. 53, no. 12, pp. 4655–4666, Dec 2007.

[108] R. Mazumder, J. H. Friedman, and T. Hastie, “Sparsenet: Coordinate descent with nonconvex penalties,” *Journal of the American Statistical Association*, vol. 106, no. 495, pp. 1125–1138, 2011.

[109] S. Bhardwaj and S. Vrudhula, “A fast and accurate approach for full chip leakage analysis of nano-scale circuits considering intra-die correlations,” in *VLSI Design, 2007., 20th International Conference on*, 2007, pp. 589–594.

[110] S. P. Mohanty and E. Kougianos, “Incorporating manufacturing process variation awareness in fast design optimization of nanoscale cmos vcos,” *IEEE Transactions on Semiconductor Manufacturing*, vol. 27, no. 1, pp. 22–31, Feb 2014.

[111] R. Kanj and R. Joshi, “A novel sample reuse methodology for fast statistical simulations with applications to manufacturing variability,” in *Quality Electronic Design (ISQED), 2012 13th International Symposium on*, 2012, pp. 672–678.

[112] S. Liu, Q. Qiu, and Q. Wu, “Full-chip leakage current estimation based on statistical sampling techniques,” in *Proceedings of the 18th ACM Great Lakes symposium on VLSI*, ser. GLSVLSI ’08. New York, NY, USA: ACM, 2008, pp. 391–394.

[113] G. Asadi and M. Tahoori, “An accurate ser estimation method based on propagation probability [soft error rate],” in *Design, Automation and Test in Europe, 2005. Proceedings*, March 2005, pp. 306–307 Vol. 1.

[114] Y. Yang and N. Jha, "Finprin: Finfet logic circuit analysis and optimization under pvt variations," pp. 1–1, 2013.

[115] N. Paydavosi, S. Venugopalan, Y. Chauhan, J. Duarte, S. Jandhyala, A. Niknejad, and C. Hu, "$bsim - spice$ models enable finfet and utb ic designs," *Access, IEEE*, vol. 1, pp. 201–215, 2013.

[116] L. Garg and V. Sahula, "Efficient CMOS subthreshold leakage analysis with improved stack based models in presence of parameter variations," *Electronics Letters*, vol. 49, no. 10, pp. 644–646, May 2013.

[117] S. e. a. Natarajan, "A 32nm logic technology featuring 2nd-generation high-k + metal-gate transistors, enhanced channel strain and 0.171 um2 sram cell size in a 291mb array," in *Electron Devices Meeting, 2008. IEDM 2008. IEEE International*, Dec 2008, pp. 1–3.

[118] A. Abdollahi, F. Fallah, and M. Pedram, "A robust power gating structure and power mode transition strategy for mtcmos design," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 15, no. 1, pp. 80–89, Jan 2007.

[119] D.-C. Juan, Y.-T. Chen, M.-C. Lee, and S.-C. Chang, "An efficient wake-up strategy considering spurious glitches phenomenon for power gating designs," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 18, no. 2, pp. 246–255, Feb 2010.

[120] A. Calimera, E. Macii, and M. Poncino, "Nbti-aware clustered power gating," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 16, no. 1, pp. 3:1–3:25, Nov. 2010.

[121] A. B. Kahng, S. Kang, and B. Park, “Active-mode leakage reduction with data-retained power gating,” in *Design, Automation Test in Europe Conference Exhibition (DATE), 2013*, March 2013, pp. 1209–1214.

[122] H. Li, M. Mansour, S. Maturi, and L.-C. Wang, “A new sampling method for analog behavioral modeling,” in *Circuits and Systems (ISCAS), Proc. of 2010 IEEE International Symposium on*, 2010, pp. 2908–2911.

[123] J. Kao, S. Narendra, and A. Chandrakasan, “MtCMOS hierarchical sizing based on mutual exclusive discharge patterns,” in *Design Automation Conference, 1998. Proceedings*, 1998, pp. 495–500.

[124] M. Anis, S. Areibi, M. Mahmoud, and M. Elmasry, “Dynamic and leakage power reduction in mtCMOS circuits using an automated efficient gate clustering technique,” in *Design Automation Conference, 2002. Proceedings. 39th*, 2002, pp. 480–485.

[125] C. Long and L. He, “Distributed sleep transistor network for power reduction,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 12, no. 9, pp. 937–946, 2004.

[126] K. Nose, S.-I. Chae, and T. Sakurai, “Voltage dependent gate capacitance and its impact in estimating power and delay of CMOS digital circuits with low supply voltage,” in *Low Power Electronics and Design, 2000. ISLPED ’00. Proceedings of the 2000 International Symposium on*, 2000, pp. 228–230.

Brief Curriculum-Vitae

Lokesh Garg obtained his Bachelor of Engineering (B. E.) degree in Electronics & Communication from Seedling Academy of Design, Technology & Management, Jaipur in 2007. He obtained his Master of Technology (M. Tech.) degree in VLSI Design from the Malaviya National Institute of Technology, Jaipur in 2010. He is currently pursuing his Ph.D. in the field of CAD-VLSI in the department of Electronics and Communication Engineering at Malaviya National Institute of Technology Jaipur, India. His current research interests are into the areas of macromodeling of analog and digital circuits.