



University of
Massachusetts
Amherst

Comprehensive Analysis of Leakage Current in Ultra Deep Sub-micron (udsm) Cmos Circuits

Item Type	Thesis (Open Access)
Authors	Rastogi, Ashesh
DOI	10.7275/343882
Download date	2025-06-09 03:52:35
Link to Item	https://hdl.handle.net/20.500.14394/47346

COMPREHENSIVE ANALYSIS OF LEAKAGE CURRENT IN ULTRA
DEEP SUB-MICRON (UDSM) CMOS CIRCUITS

A Thesis Presented

by

ASHESH RASTOGI

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

September 2007

Department of Electrical and Computer Engineering

COMPREHENSIVE ANALYSIS OF LEAKAGE CURRENT IN ULTRA
DEEP SUB-MICRON (UDSM) CMOS CIRCUITS

A Thesis Presented

by

ASHESH RASTOGI

Approved as to style and content by:

Sandip Kundu, Chair

Wayne P. Burlison, Member

Massimo V. Fischetti, Member

C.V. Hollot, Department Head
Department of Electrical and Computer Engineering

To my parents

ACKNOWLEDGMENTS

I express my sincere gratitude to Professor Sandip Kundu for his constant support, encouragement and guidance throughout this thesis research. He always motivated me to develop a deeper understanding in VLSI research and strive for the best. I will always remember his invaluable advice and generosity over the past two years.

I would like thank my thesis faculty committee members, Professor Wayne Burleson and Professor Massimo Fischetti, who even generously took time off from his vacation, to provide valuable feedback. Professor Wayne Burleson's course teachings inspired me to pursue my masters in the field of VLSI research for which I am grateful. Special thanks to Professor Russell Tessier for introducing me to the field of reconfigurable computing that helped me formulate a part of thesis work related to FPGAs.

I thank the Semiconductor Research Corporation (SRC) for supporting this research work. I would like to thank Spencer Gold (AMD), Ruchir Puri (IBM) and Suriyaprakash Natarajan (INTEL), who were the industrial liaisons for SRC Task ID 1420. In particular, I would like to express my sincere gratitude to Suriyaprakash Natarajan for his valuable guidance and constant encouragement during the course of my internship at INTEL.

A special thanks to Wei Chen, Kunal Ganeshpure and Alodeep Sanyal for their worthy contributions in this thesis work and to all my colleagues in our research group and elsewhere for providing me such an exciting learning environment over the past six years and helped made my experience in UMASS a great one.

Finally, I thank my parents for their greatest support and guidance that I have been receiving all these years. Their support has helped me accomplish this achievement.

ABSTRACT

COMPREHENSIVE ANALYSIS OF LEAKAGE CURRENT IN ULTRA DEEP SUB-MICRON (UDSM) CMOS CIRCUITS

SEPTEMBER 2007

ASHESH RASTOGI

B.S., UNIVERSITY OF MASSACHUSETTS AMHERST

M.S.E.C.E., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Sandip Kundu

Aggressive scaling of CMOS circuits in recent times has led to dramatic increase in leakage currents. Previously, sub-threshold leakage current was the only leakage current taken into account in power estimation. But now gate leakage and reverse biased junction band-to-band-tunneling leakage currents have also become significant. Together all the three types of leakages namely sub-threshold leakage, gate leakage and reverse bias junction band-to-band tunneling leakage currents contribute to more than 25% of power consumption in the current generation of leading edge designs. Different sources of leakage can affect each other by interacting through resultant intermediate node voltages. This is called loading effect and it leads to further increase in leakage current. On the other hand, sub-threshold leakage current decreases as more number of transistors is stacked in series. This is called stack effect. Previous works have been done that analyze each type of leakage current and its effect in detail but independent of each other. In this

work, a pattern dependent steady state leakage estimation technique was developed that incorporates loading effect and accounts for all three major leakage components, namely the gate leakage, band to band tunneling leakage and sub-threshold leakage. It also considers transistor stack effect when estimating sub-threshold leakage. As a result, a coherent leakage current estimator tool was developed. The estimation technique was implemented on 65nm and 45nm CMOS circuits and was shown to attain a speed up of more than 10,000X compared to HSPICE. This work also extends the leakage current estimation technique in Field Programmable Gate Arrays (FPGAs). A different version of the leakage estimator tool was developed and incorporated into the Versatile Place & Route CAD tool to enable leakage estimation of design after placement and routing.

Leakage current is highly dependent on the steady state terminal voltage of the transistor, which depends on the logic state of the CMOS circuit as determined by the input pattern. Consequently, there exists a pattern that will produce the highest leakage current. This work considers all leakage sources together and tries to find an input pattern(s) that will maximize the composite leakage current made up of all three components.

This work also analyzes leakage power in presence of dynamic power in a unique way. Current method of estimating total power is to sum dynamic power which is $\frac{1}{2}\alpha C_L V_{DD}^2 f$ and sub-threshold leakage power. The dynamic power in this case is probabilistic and pattern independent. On the other hand sub-threshold leakage is pattern dependent. This makes the current method very inaccurate for calculating total power. In this work, it is shown that leakage current can vary by more than 8% in time in presence of switching current.

TABLE OF CONTENTS

	Page
ACKNOWLEDGMENTS	iii
ABSTRACT	v
LIST OF TABLES.....	ix
LIST OF FIGURES	x
CHAPTER	
1. INTRODUCTION	1
1.1 Leakage Sources	3
1.2 Impact of transistor logic values on leakage.....	8
1.3 Loading Effect	10
1.4 Leakage Estimation in Field Programmable Gate Arrays (FPGA)	12
1.5 Leakage Maximization.....	13
2. EFFICIENT AND ACCURATE LEAKAGE ESTIMATION	15
2.1 Compact Model Formulation.....	15
2.2 Pattern Dependent Leakage Estimation Considering Loading Effect	18
2.3 Leakage Current Estimation Algorithm.....	22
2.4 Results.....	25
2.5 Validation.....	27
2.6 Leakage Estimation incorporating Loading Effect using Newton Raphson Method	30
2.7 Results with Newton Raphson Method.....	34
2.8 Conclusions.....	38
2.9 Leakage Estimation in Field Programmable Gate Arrays (FPGA)	38
2.9.1 Efficient Leakage Estimation in VPR CAD Tool.....	41
2.9.2 2-input Multiplexer Leakage Power Computation	42
2.9.3 Inverter Leakage Power Computation	44
2.9.4 SRAM Leakage Power Computation	45
2.9.5 4-input LUT Leakage Power Computation	47
2.9.6 4-input Input Multiplexer Leakage Power Computation.....	48
2.9.7 Logic Block Leakage Power Computation.....	49
2.9.8 Clock Network Leakage Power Computation	50
2.9.9 Routing Leakage Power Computation.....	50

2.9.10	Results.....	52
2.9.10.1	Impact of technology scaling.....	53
2.9.10.2	Leakage Power Vs FPGA Resource Utilization ...	54
2.9.10.3	Leakage Power Vs FPGA Resource Type	56
2.9.10.4	Effect of Number of Clusters N on Leakage Power.....	57
2.9.10.5	Effect of Number of Clusters Inputs I on Leakage Power.....	59
2.9.10.6	Effect of Number of LUT Inputs K on Leakage Power.....	60
2.9.10.7	Effect of Connection Block Flexibility F_c on Leakage Power.....	61
2.9.11	Conclusions.....	62
3.	COMPOSITE LEAKAGE CURRENT MAXIMIZATION	63
3.1	Leakage Characterization of Cell Library.....	66
3.2	Circuit Modification.....	68
3.3	Pattern Search for Maximum Leakage	70
3.3	Results.....	72
3.4	Conclusions.....	77
4.	IMPACT OF DYNAMIC POWER ON LEAKAGE CURRENT	78
4.1	Switching Current Estimation.....	79
4.2	Total Current Estimation.....	80
4.3	Results.....	83
4.4	Conclusions.....	88
5.	CONCLUSIONS AND FUTURE WORK.....	89
5.1	Conclusions.....	89
5.2	Future Work.....	90
	BIBLIOGRAPHY	91

LIST OF TABLES

Table	Page
2.1. Gate and BTBT leakage for different bias states for 65nm and 45nm NMOS device.....	17
2.2. Maximum gate leakage current and loading voltage obtained after 100 random pattern simulations by STABLE.....	26
2.3. Total leakage current in micro-amperes obtained after 100 random pattern simulations with and without loading effect.....	26
2.4. Gate Leakage and BTBT Leakage Values for 65nm and 45nm NMOS Device obtained from NGSPICE.....	28
2.5. Average Total Leakage Current in Micro-Amperes and CPU Runtimes in seconds after 100 Random Pattern Simulation.....	29
2.6. CPU Runtimes in seconds after 100 Patterns for NGSPICE (NG), STABLE (ST) and with Newton Method (STN).....	37
2.7. Leakage current values of NMOS device in 65nm used for leakage estimation in FPGA.....	41
2.8. Inverter Leakage Power	45
2.9. SRAM Leakage Power	46
2.10. 2-input Multiplexer Leakage Power for all possible states.....	46
2.11. Total leakage power vs. FPGA resource utilization	55
2.12. Leakage Power dissipated by each resource type.....	56
3.1. Total leakage corresponding to each input combination in a NAND gate	63
3.2. Sub-threshold leakage maximization bounds in ISCAS 85 benchmark circuits ..	73
3.3. Gate leakage maximization bounds in ISCAS 85 benchmark circuits	74
3.4. BTBT leakage maximization bounds in ISCAS 85 benchmark circuits.....	74
3.5. Total leakage maximization bounds in ISCAS 85 benchmark circuits	75
4.1. Comparison of average values of total and leakage current in mA for ISCAS benchmarks circuits	84
4.2. Comparison of peak switching current in mA for different delay models for ISCAS benchmark circuits.....	85
4.3. Comparison of average total current obtained from HSPICE and our technique with and without calibration.....	86

LIST OF FIGURES

Figure	Page
1.1. ITRS Roadmap showing Static Power surpassing Dynamic Power.....	2
1.2. Illustration of sub-threshold leakage in a NMOS	4
1.3. Illustration of gate leakage in a NMOS device (left) and the tunneling mechanism in band diagram (right) adapted from [9]	6
1.4. Illustration of BTBT leakage in a NMOS.....	7
1.5. Illustration of GIDL leakage in a NMOS	8
1.6. Illustration showing Leakage Dependence on States in a NOR2 cell	9
1.7. Illustration showing Stacking Effect in a NOR2 cell.....	10
1.8. Illustration of Loading Effect on c17.....	11
2.1. Leakage current components in six steady states in a NMOS device.....	16
2.2. Gate leakage (left) and sub-threshold leakage (right) sensitivity versus loading effect in 45nm NMOS device	18
2.3. Effect of loading gate illustrated at gate level (left) and at transistor level (right) showing the bias states in the fan-out gate.....	19
2.4. Method to compute loading voltage in a cell using SPICE	21
2.5. Loading voltage in NOR2 cell when output is 0 (left) and when output is 1 (right) in 65nm and 45nm.....	22
2.6. STABLE algorithm for estimating overall leakage in a circuit including loading effect	24
2.7. Frequency distribution of loading voltage in 45nm circuits	27
2.8. Illustration showing possible terminal voltages and logic state of transistors in a 2-input NOR cell.....	31
2.9. STABLE algorithm with Newton-Raphson Method. Part of algorithm outside the dashed box corresponds to the baseline algorithm.....	33
2.10. Comparison of maximum gate leakage current with and without Newton Raphson Method	34
2.11. Comparison of maximum loading voltage with and without Newton Raphson Method	35
2.12. Comparison of average total leakage current with and without Newton Raphson Method	35
2.13. Frequency Distribution of Loading Voltage in 45nm circuits after Newton Raphson Method	36
2.14. Island Style FPGA architecture (adapted from [17]).....	39

2.15.	BLE (top) and Logic Cluster (bottom) adapted from [38].....	40
2.16.	2-Input multiplexer with inputs = ‘01’	43
2.17.	Transistor states in an Inverter with input = 0 and input = 1	44
2.18.	1-bit SRAM with Data = 0 (left) and Data = 1 (right).....	45
2.19.	2-input LUT implemented using 2-input multiplexers (adapted from [17]).....	47
2.20.	4-input multiplexer implemented using 2-input multiplexer tree (adapted from [17]).....	48
2.21.	Combinational Logic Block (CLB) or logic cluster (adapted from [17]).....	49
2.22.	Connection block (left) and Switch block (right)	51
2.23.	Routing Segment (adapted from [19])	52
2.24.	Comparison of Leakage Power estimated from original and modified Power Model for each resource type in s298 design	53
2.25.	Sub-threshold and Total Leakage Power Comparison.....	54
2.26.	Logic block leakage power for designs placed & routed with different cluster sizes	57
2.27.	Routing leakage power for designs placed & routed with different cluster sizes	58
2.28.	Routing leakage power for designs placed & routed with cluster of size N = 4 and of different number of inputs	59
2.29.	Logic block leakage power for designs placed & routed with cluster of N = 4, I = 10 and of different number of LUT inputs.....	60
2.30.	Routing leakage power for designs placed & routed on fabric with different connection block flexibilities.....	61
3.1.	Illustration showing weighted max-satisfiability problem	65
3.2.	Histogram showing leakage weights of three different gates for different input vectors.....	68
3.3.	Circuit modification.....	69
3.4.	Circuit modification for peak leakage estimation.....	71
3.5.	Variation of maximum absolute ETA in random pattern logic simulation	73
3.6.	Improvement in $\eta_{relative}$ from CMPG algorithm over preprocessing for sub-threshold leakage.....	76
4.1.	Plot of total current with and without leakage for c7552 with time	82
4.2.	Plot of total leakage current for c7552 with time	82
4.3.	Leakage current in presence and absence of switching activity	84
4.4.	Total current obtained from our technique and HSPICE for c3540	87

CHAPTER 1

INTRODUCTION

The demand for greater integration, higher performance and lower dynamic power dissipation drives scaling of CMOS devices. In recent times, as we approach atomic scale devices, leakage currents have increased dramatically, leading to higher static power dissipation. In nano-scaled CMOS devices, there are many leakage sources such as gate oxide tunneling based leakage, sub-threshold leakage, band-to-band tunneling (BTBT) based leakage, gate induced drain leakage (GIDL), drain induced bulk leakage (DIBL) etc. [5]. The magnitude of each leakage component depends on the process technology used. Use of high-K dielectric gate helps reduce gate oxide leakage current. Use of SiGe layer to strain Si for improving carrier mobility to increase performance causes an increase in sub-threshold and BTBT leakage current [40]. This work uses Berkeley Predictive Technology (BPTM) process models for all experiments performed. For 65 and 45nm BPTM technology nodes, in terms of scale, the most important sources of leakage are: sub-threshold leakage, gate leakage and the reversed bias junction band-to-band tunneling (BTBT) leakage. Sub-threshold current rises due to lowering of threshold voltage which is scaled to maintain transistor ON current on the face of falling power supply voltage. Gate leakage current density is increasing due to scaling of oxide thickness resulting in rising tunneling current. In fact, gate leakage is expected to increase at least by 10X for each of the future generations [1]. Reverse-biased tunneling band-to-band leakage is increasing due to reduction in junction depletion width that is necessary to contain transistor short channel effects (SCE). In previous CMOS technologies,

dynamic power easily wins over leakage power but as shown in Figure 1.1, for 65nm the ITRS roadmap predicts that this trend is coming to an end.

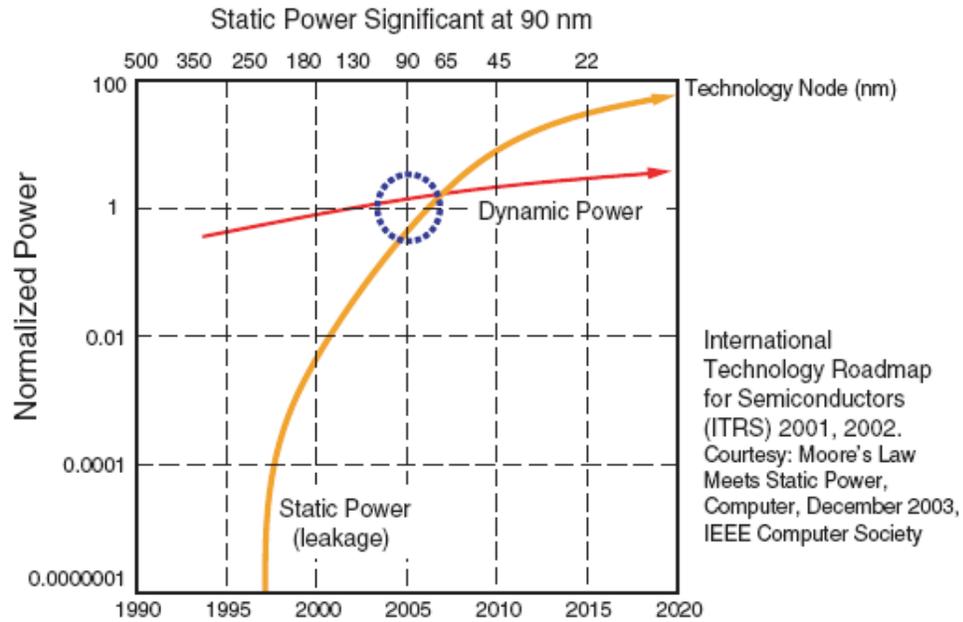


Figure 1.1. ITRS Roadmap showing Static Power surpassing Dynamic Power

Leakage power is a major concern not only in ASICs but also in field programmable gate arrays (FPGA). FPGA has more transistors per logic function than ASIC, making it more susceptible to leakage power. FPGA consists of large number of logic blocks (BLE), switch blocks and connection blocks over its entire array. Each of these blocks consists of 1-bit SRAM cells, pass transistors, tri-state drivers and multiplexers. All these devices in idle state conditions dissipate power. Altera Stratix II EP2S180 FPGA built on 90nm process technology contains about 180k 4-input LUTs, 9Mbit memory and 384 18x18 multipliers and can dissipate static power of about 5W [13]. In fact, more than 25% of the total static power consumption comes from routing [14]. Therefore, this work targets to develop an efficient and accurate estimation of total leakage current in FPGAs

1.1 Leakage Sources

This section briefly describes each type of leakage current.

1) Sub-Threshold Leakage Current (I_{SUB}) – Sub-threshold current is the most dominant among all sources of leakages. It is caused by minority carriers drifting across the channel from drain to source due to presence of weak inversion layer when the transistor is operating in cut-off region ($V_{GS} < V_{TH}$). The minority carrier concentration rises exponentially with gate voltage V_G and so the plot of $\log(I_{SUB})$ versus V_G is a linear curve with typical slopes of 60-80mV per decade. I_{SUB} depends on the substrate doping concentration and halo implant, which modifies the threshold voltage V_{TH} . I_{SUB} also rises exponentially with temperature.

$$I_{SUB} = I_o \left(1 - \exp\left(\frac{-qV_{DS}}{kT}\right) \right) \exp\left(q \frac{V_{GS} - V_{TH} - V_{OFF}}{nkT}\right) \quad (1.1)$$

Where, V_{OFF} is the offset voltage in sub-threshold region and I_o is given as:

$$I_o = \mu \frac{W_{eff}}{L_{eff}} \left(\frac{kT}{q}\right)^2 \sqrt{\frac{q \epsilon_{si} NDEP}{2\phi_s}} \quad (1.2)$$

Condition for I_{SUB} to occur in an NMOS transistor is shown below.

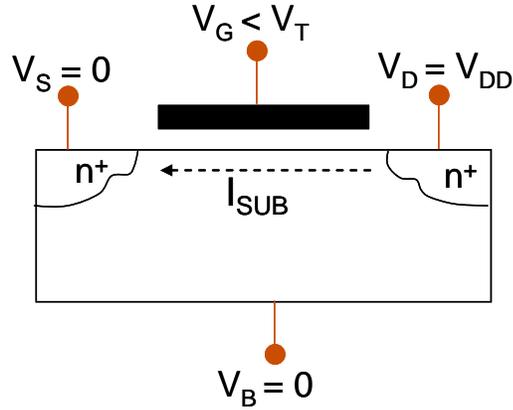


Figure 1.2. Illustration of sub-threshold leakage in a NMOS

2) Gate Leakage Current (I_G) – Gate leakage is a serious concern at gate oxide thicknesses below 2nm. With such thin gate oxide, fairly small potential difference across the gate oxide can induce high electric field, causing electrons to easily tunnel through the oxide. This process is called Fowler-Nordheim Tunneling. Gate leakage consists of three components: gate-to-channel (I_{GC}), gate-to-bulk (I_{GB}) and gate-to-source/drain diffusion (I_{GS} / I_{GD}) leakage. In NMOS, I_{GC} occurs due to electron conduction-band tunneling mechanism (ECB). Similarly, in PMOS, hole valence-band tunneling mechanism (HVB) causes I_{GC} . I_{GC} can further be split into two components, one from gate to source via the channel (I_{GCS}) and other to drain (I_{GCD}) [7]. I_{GC} occurs only when device operates in inversion region. The current densities are given as:

$$J_{GCD} = J_{GCO} \frac{PIGCD V_{DSEFF} \exp(-PIGCD V_{DSEFF}) - 1 + 10^{-4}}{(PIGCD V_{DSEFF})^2 + 2 \times 10^{-4}} \quad (1.3)$$

$$J_{GCS} = J_{GCO} \frac{1 - (PIGCD V_{DSEFF} + 1) \exp(-PIGCD V_{DSEFF}) + 10^{-4}}{(PIGCD V_{DSEFF})^2 + 2 \times 10^{-4}} \quad (1.4)$$

Where, $PIGCD$ is a constant parameter. J_{GC0} is J_{GC} at $V_{DS} = 0$ and is modeled as:

$$J_{GC0} = A \times T_{OXRATIO} \times V_{GSE} \times V_{AUX} \times \exp\left(\frac{-B \times TOXE \times (AIGC - BIGC)}{\times V_{OXDEPINV} (1 + CIGC \times V_{OXDEPINV})}\right) \quad (1.5)$$

V_{AUX} is an auxiliary function that models density of tunneling carriers and depends on region of operation and $V_{OXDEPINV}$ is the voltage across the oxide during inversion.

I_{GB} consists of two components I_{GBACC} and I_{GBINV} . I_{GBACC} occurs because of ECB and is significant in accumulation region of operation ($V_G > 0$). Electrons tunneling from valence-band (EVB) cause I_{GBINV} , which is dominant in the inversion region of operation ($V_G < 0$). The current densities are given as:

$$J_{GBACC} = A \times T_{OXRATIO} \times V_{GB} \times V_{AUX} \times \exp\left(\frac{-B \times TOXE \times (AIGBACC - BIGBACC)}{\times V_{OXACC} (1 + CIGBACC \times V_{OXACC})}\right) \quad (1.6)$$

$$J_{GBINV} = A \times T_{OXRATIO} \times V_{GB} \times V_{AUX} \times \exp\left(\frac{-B \times TOXE \times (AIGBINV - BIGBINV)}{\times V_{OXDEPINV} (1 + CIGBINV \times V_{OXDEPINV})}\right) \quad (1.7)$$

I_{GSO} and I_{GDO} are parasitic leakage currents that pass through gate to source-drain extension overlap region. I_{GDO} in off-state ($V_G = 0$) NMOS device is also known as edge directed tunneling current (EDL) [8] and is higher than its on-state counterpart. PMOS devices have less gate leakage compared to NMOS devices as holes have higher barrier of 4.5eV compared to 3.1eV for electron. Total gate leakage current is given as:

$$I_G = I_{GC} + I_{GB} + I_{GS} + I_{GD} \quad (1.8)$$

A bias condition at which I_{GC} , I_{GB} and I_{EDL} occurs is shown below.

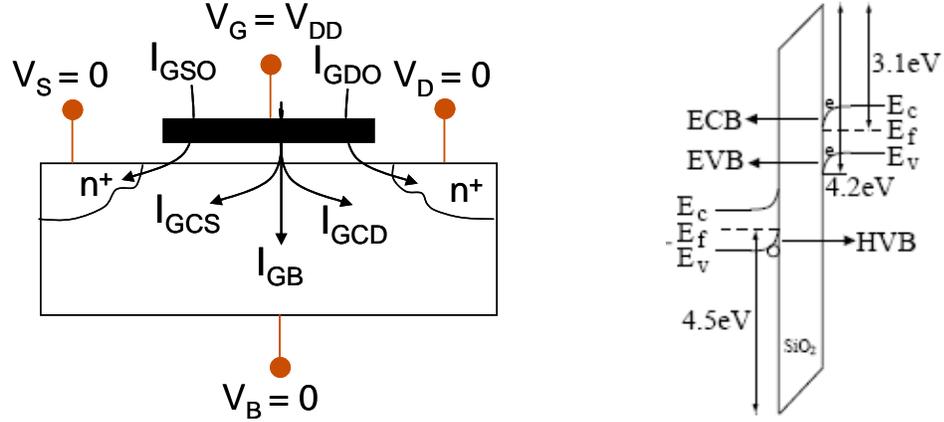


Figure 1.3. Illustration of gate leakage in a NMOS device (left) and the tunneling mechanism in band diagram (right) adapted from [9]

3) Band-to-Band Tunneling (BTBT) Current (I_{BTBT}) – This represents leakage from drain/source to the bulk and depend on the substrate doping profiles. BTBT leakage current tends to be significant in 65nm and 45nm process technologies and is caused by tunneling of electrons from n-type source/drain to p-type substrate in NMOS device in presence of very high electric field ($> 1MV/cm$) I_{BTBT} is modeled for a rectangular junction. BTBT leakage happens from both the bottom and side of the junction [2] and is given as:

$$I_{BTBT_ij} = W_{eff} A \frac{E_j}{\sqrt{E_g}} V_{ib} \exp\left(\frac{-B E_g^{3/2}}{E_j}\right) \quad (1.9)$$

Where i is drain/source, E_g is the band-gap, V_{iB} is the applied potential on source/drain with respect to bulk and E_j is the average electric field on the side/bottom of the junction are given as:

$$E_{side} = \sqrt{\frac{2q NDEP NSD (V_{iB} + V_{BISIDE})}{\epsilon_{si} (NDEP + NSD)}} \quad (1.10)$$

$$E_{bottom} = \sqrt{\frac{2q NSUB NSD (V_{iB} + V_{BIBOTTOM})}{\epsilon_{si} (NSUB + NSD)}} \quad (1.11)$$

Parameters NDEP, NSUB and NSD are the channel doping concentration at depletion edge, substrate doping and source/drain diffusion doping concentration respectively and $V_{BISIDE/BOTTOM}$ is the built-in potential. In the above evaluation of electric fields, uniform doping is assumed to avoid complexity. Total band-to-band tunneling current is given by:

$$I_{BTBT} = \sum_{i=source, drain} \sum_{j=side, bottom} I_{BTBT_ij} ; \text{if } (V_{iB} + V_{Bij}) > E_g \quad (1.12)$$

Bias condition for I_{BTBT} to occur is shown below.

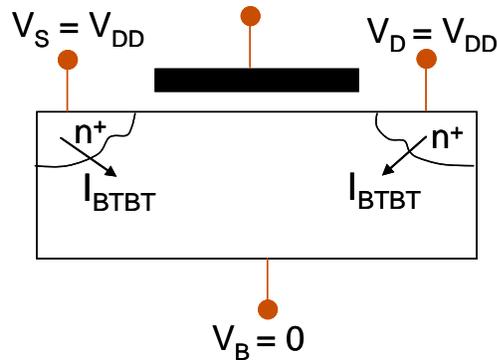


Figure 1.4. Illustration of BTBT leakage in a NMOS

4) Gate Induced Drain Leakage Current (I_{GIDL}) – Is not a major source of leakage. During accumulation region additional holes are created at the oxide surface, which results in narrower depletion region at the drain leading to additional leakage.

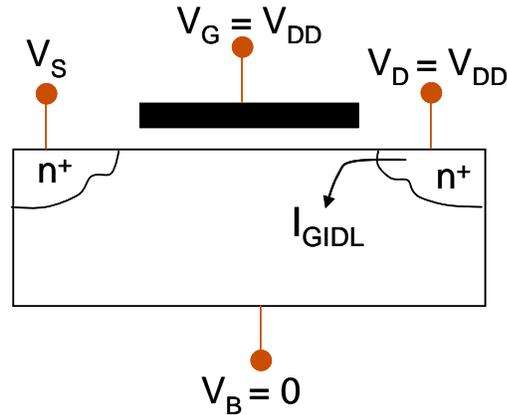


Figure 1.5. Illustration of GIDL leakage in a NMOS

1.2 Impact of transistor logic values on leakage

From the previous section, it is evident that all leakage currents depend on the bias voltage applied at the terminals. The bias voltages applied depend on the logic (Boolean) values at the transistor terminals. The logic values on device terminals in turn are determined by the input patterns applied to a circuit and the location of the device in that circuit. To illustrate how leakage currents vary with the logic states at the terminals of a transistor, a 2-input NOR gate example is shown in Figure 1.6.

a) Sub-threshold Leakage: In Figure 1.6, the transistors are annotated with their respective input and output logic values. Transistors P1 and N2 are ON, so we do not consider sub-threshold leakage through these devices. Transistor P2 is in OFF state and is the main contributor for sub-threshold leakage. P2 leaks to the ground through the

conducting device N2. N1 has identical logic values on its source and drain terminals, so there is no leakage through N1.

b) Gate Leakage: In P1, the gate has a logic value 0, while the source and drain terminals are at logic value 1. Hence there is gate leakage from drain and source terminals of this transistor. P2 has lesser gate leakage than P1 as leakage happens from gate to drain terminal only. On the other hand, N2 has high gate leakage from gate to both drain and source terminals.

c) Band-to-Band Tunneling Leakage: There is no BTBT leakage from both source and drain terminals of P1 as PMOS bulk is tied to V_{DD} . P2 has some BTBT leakage form its source terminal and N1 and N2 exhibit no leakage of this type as none of the NMOS transistors have a field from source/drain to the bulk which is tied to ground.

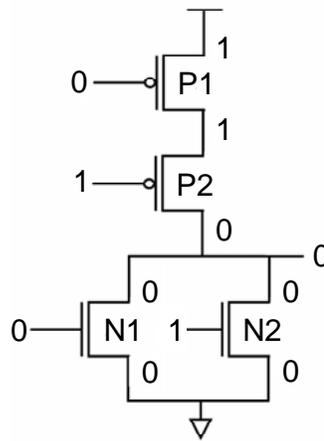


Figure 1.6. Illustration showing Leakage Dependence on States in a NOR2 cell

This example illustrates strong dependency of all leakage sources on the logic values at the terminals of a transistor. The present logic states of a device sometimes depend on logic values under the previous pattern. Consider the example in Figure 1.7 when P1 is

also turned off. In this case, the state of node *int* is unknown. If P1 was previously in a conducting state as in Figure 1.2, the node *int* has a logic value ‘1’. However, if P1 was off and P2 was on in the previous cycle, *int* value will be ‘0’. If *int* value is close to V_{DD} , P1 will not have negligible sub-threshold leakage. On the other hand, if *int* is closer to 0, P2 will have negative gate to source condition, which will reduce sub-threshold leakage. This is known as the stacking effect [10][11]. Thus, the leakage currents not only depend on the present logic states but also on the previous logic states. This thesis work uses this idea as a basis of leakage current estimation at transistor level and at circuit level.

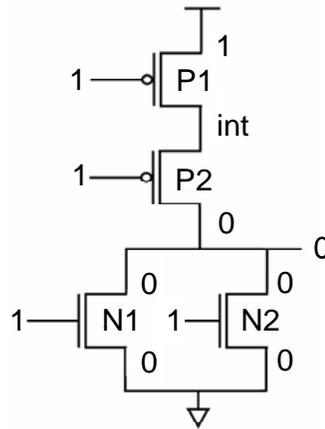


Figure 1.7. Illustration showing Stacking Effect in a NOR2 cell

1.3 Loading Effect

Consider the case of leakages in c17 benchmark circuit as shown in Figure 1.8. The gate leakage currents from input of gates G5 and G6 enter the output node of G3 causing a small increase in its output voltage. Now the gate bias V_{GS} on devices in gates G5 and G6 is greater than zero. This in turn increases the sub-threshold leakage in gates G5 and G6. This is known as loading effect and it depends on the number of fan-out gates and input pattern applied.

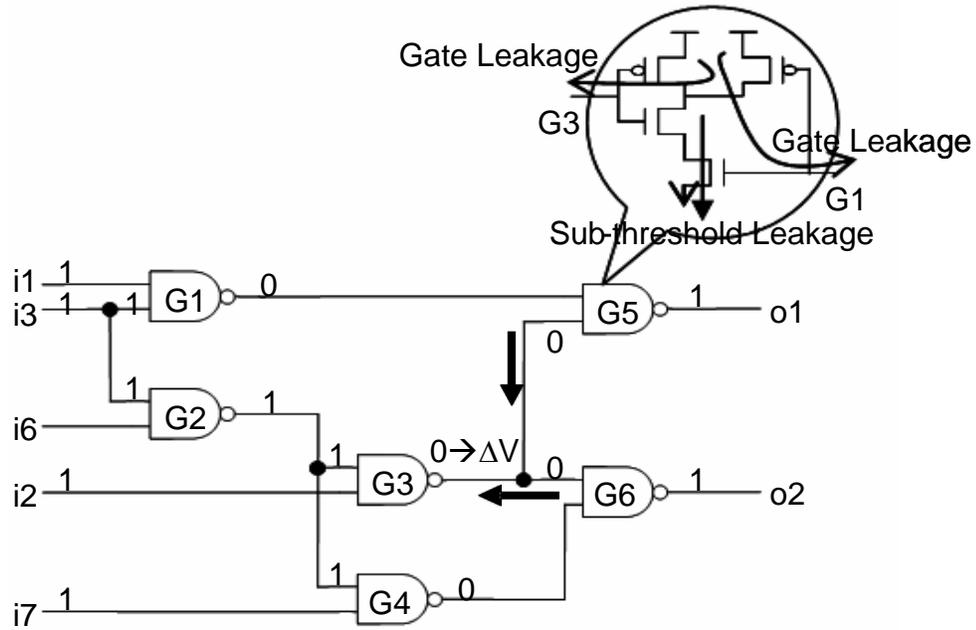


Figure 1.8. Illustration of Loading Effect on c17

Studies on estimation of total leakage at both gate and circuit levels have been reported in literature to date. Mukhopadhyay et al presented a compact model for estimating total leakage current by solving KCL equations at each node in basic circuits such as INVERTER, NAND and NOR gates [2]. This model did not incorporate loading effect. In a subsequent publication, Mukhopadhyay et al considered the impact of loading effect at circuit level by computing KCL equations at each node of the circuit and minimized the number of equations to be solved by ignoring the impact of gate leakage in driver gate due to gate leakage from fan-out gates and showed that loading effect is significant [3]. However, they do not re-calculate gate leakage in the fan-out gates after considering loading effect at the output of driver gate, which is crucial for obtaining accurate gate leakage. Brown et al [4] reported an efficient technique for estimating gate leakage current by performing a logic state-based analysis of the transistors. The authors

characterized gate leakage of a single transistor based on six possible steady states and use the state-based leakage characterization for estimating gate leakages at cell and circuit level. Their analysis is limited to gate leakage only and loading effect is not considered. Rahman et al [5] proposed a technique for estimating gate leakage and sub-threshold leakage current using state-based leakage characterization technique for only four steady states but did not consider loading effect as well as band-to-band tunneling leakage and stacking effects. This thesis work aims to estimate all major sources of leakage in a given circuit and incorporates loading effect and its effect on leakage current with additional benefit of higher simulation performance than HSPICE.

1.4 Leakage Estimation in Field Programmable Gate Arrays (FPGA)

In FPGAs, most of the work has been done in estimating total power dissipation that consists of three components: (1) dynamic power, (2) short circuit power PSC and (3) static power. High emphasis has been laid on in developing models of dynamic power dissipation that arise due to switching. There are very few literatures that have laid insight into leakage power analysis in FPGAs. Li et al in [15] have done mixed-level power analysis by considering both dynamic and leakage power. The authors have used static power macro-models in their power model analysis. The macro-models were derived by performing a set of SPICE simulations for different LUT and buffer sizes. As to our knowledge there is no SPICE tool that can be used to band-to-band tunneling leakage. SPICE tools have only the capability to estimate sub-threshold leakage current and have recently added the capability for estimating gate leakage. Further the authors have not mention the process technology on which the power analysis is done. Poon et al

in [16][17] have developed a model for estimating total power in FPGAs. They have integrated their power model into widely used Versatile Place and Route (VPR) CAD tool. In this model, the authors have only considered sub-threshold leakage for 180 nm process technology [18]. In this thesis work, a technique was developed and integrated into the widely used Versatile Place and Route (VPR) CAD tool making it easier for designers to calculate leakage power right after place and route.

1.5 Leakage Maximization

A designer must know the maximum or worst case power of the design before the design is imported into silicon. After knowing what input pattern(s) can produce maximum leakage current, a designer can make modifications in the design to reduce leakage. With extreme scaling of transistors, not only accurate leakage power estimation is crucial but knowing the limits of static power dissipation of a design is also vital for a chip/design to succeed in market, especially with the advent of mobile processing. Several techniques have been implemented to maximize leakage current in a circuit but are only limited to sub-threshold leakage. Finding the exact maximum or minimum I_{LEAK} and the vectors causing these extremes is a computationally intractable problem [22] belonging to the class of NP-hard problems [20][21] as for an n-input circuit, 2^n logic simulations are required making it intractable for large value of n. Bobba and Hajj [23] proposed a graph-algorithmic solution in which they form a circuit constraint graph with 2^k nodes for a k-input circuit where each node represents a gate input pattern combination and the associated leakage power weights. The edges in the graph are drawn between conflicting nodes. Leakage maximization problem is mapped to the problem of finding

the clique cover. Since clique covering problem has been known to be intractable, they estimate lower and upper bounds on maximum leakage. However, the constraint graph grows exponentially for large number of inputs and therefore the approach does not scale very much. Ferre and Figueras [24] proposed an ATPG-based hierarchical method for maximizing and minimizing sub-threshold leakage by partitioning the circuit into sub-circuits. They obtain a lower bound which is the maximal possible set of sub-circuits with maximum leakage and the upper bound which is the sum of maximum leakages of all the sub-circuits. They reported favorable results compared to the results obtained from Monte Carlo simulations.

This thesis work, instead of finding an exact solution to the maximizing pattern generation problem, seeks to establish a tight upper and lower bound for the leakage current estimates. This approach reduces the search space and makes a practical solution attainable. The approach has shown to improve accuracy gradually along with computation by gradual tightening of lower and upper bounds. An exact solution is attainable if by chance the lower and upper bounds are found to be the same.

The thesis work is organized as follows. CHAPTER 2 presents the methodology and algorithm for accurate leakage estimation in CMOS circuits in ASIC and similar leakage estimation methodology is proposed for SRAM based FPGA with island-style architecture. In CHAPTER 3, an ATPG technique is proposed to determine an input pattern(s) that can maximize composite leakage in a given circuit. CHAPTER 4 gives an study on impact of switching current on leakage current. Finally CHAPTER 5 concludes the thesis.

CHAPTER 2

EFFICIENT AND ACCURATE LEAKAGE ESTIMATION

In this section an efficient and accurate leakage current estimation technique is presented for ASIC and FPGA. In section 1.2, it was shown that leakage current is highly dependent on the steady state terminal bias of the transistor. Using this concept, the first step is to model all sources of leakage for a single NMOS and PMOS device. For this a compact model is devised. This model forms the basic building block to all leakage current calculations.

2.1 Compact Model Formulation

Since gate leakage, band-to-band tunneling leakage and sub-threshold leakage vary almost linearly with transistor width, look-up tables can be constructed that can compute leakage current for given state values.

A single transistor has 3 terminals: source, drain and gate that can be connected to V_{DD} (logic 1) or Ground (logic 0) in $2^3 = 8$ ways, while the body or bulk is permanently connected to V_{DD} (for a PMOS) or Ground (for NMOS). Logically, source, drain or gate can have value 0 or 1. Out of 8 possible states, 2 states do not represent steady-state. These two states correspond to the cases when a transistor is in a conducting state due to its gate voltage while its source and drain are in different logic states. When a transistor is ON, its source and drain cannot be different logic values under steady state conditions. For example, in steady state, a NMOS (PMOS) device cannot have gate voltage = V_{DD} while source voltage is 0 (V_{DD}) and drain voltage is V_{DD} (0). The basic idea behind using

state based gate leakage estimation was presented in [4]. This idea will be extended for other leakages.

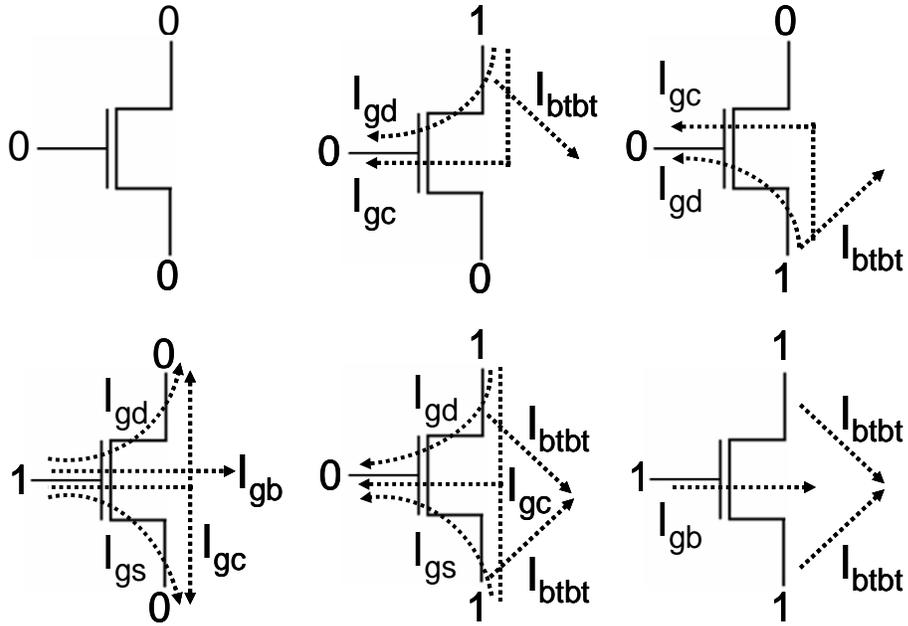


Figure 2.1. Leakage current components in six steady states in a NMOS device

For each possible steady state as shown in Figure 2.1, values of gate leakage current are computed using Berkeley Predictive BPTM models for 65nm and 45nm technologies [6][7][12] and band-to-band tunneling current is computed using model in [2]. Table 2.1 shows the values that were computed using these predictive models. It is noted that the gate and BTBT leakage values are dependent on gate and drain voltages, respectively. In steady-state conditions these values are expected to be at or close to V_{DD} and Ground voltages.

STATES [G][D][S]	I _G Leakage (nA/μm)		I _{BTBT} Leakage (nA/μm)	
	65nm	45nm	65nm	45nm
[0][0][0]	0	0	0	0
[0][1][0]	-8.83	-27.09	54.6	40.2
[0][0][1]	-8.83	-27.09	54.6	40.2
[0][1][1]	-17.65	-54.18	109.3	80.5
[1][0][0]	25.27	67.12	0	0
[1][1][1]	1.6e-7	7.2e-7	109.3	80.5

Table 2.1. Gate and BTBT leakage for different bias states for 65nm and 45nm NMOS device

In Table 2.1, a negative gate leakage current has a direction out of the gate of the transistor and a positive gate leakage points into the gate of the transistor. A similar table exists for PMOS devices. The sub-threshold leakage is also pre-computed but not under steady states bias condition as it depends exponentially on V_{GS} . While estimating leakage on a circuit level, the effect of loading has to be considered. Figure 2.2 shows the sensitivity of gate and sub-threshold leakage per unit width with small change in voltage due to loading effect. We define sensitivity as the derivative of current with respect to voltage. For a NMOS device in [100] state, gate leakage exhibits high sensitivity for smaller drop in gate voltage. The sensitivity decreases exponentially as gate voltage decreases (for higher loading voltage). BTBT leakage does not depend on gate voltage but has linearly increasing sensitivity with increase in drain/source to bulk voltages. However, for [010] NMOS state, sub-threshold leakage is extremely sensitive compared with gate leakage and can change by several micro-amperes for a small increase in gate voltage due to loading effect. Therefore, it becomes very important to estimate the modified sub-threshold leakage current by loading effect. In pre-computing sub-threshold leakage, only the effect on V_{GS} is considered. Sub-threshold leakage also depends on the

drain-to-source voltage (V_{DS}) as in equation (1.1). However, for an inverter the sub-threshold leakage is de-rated for V_{DS} [41]. Same scenario holds for a NOR gate.

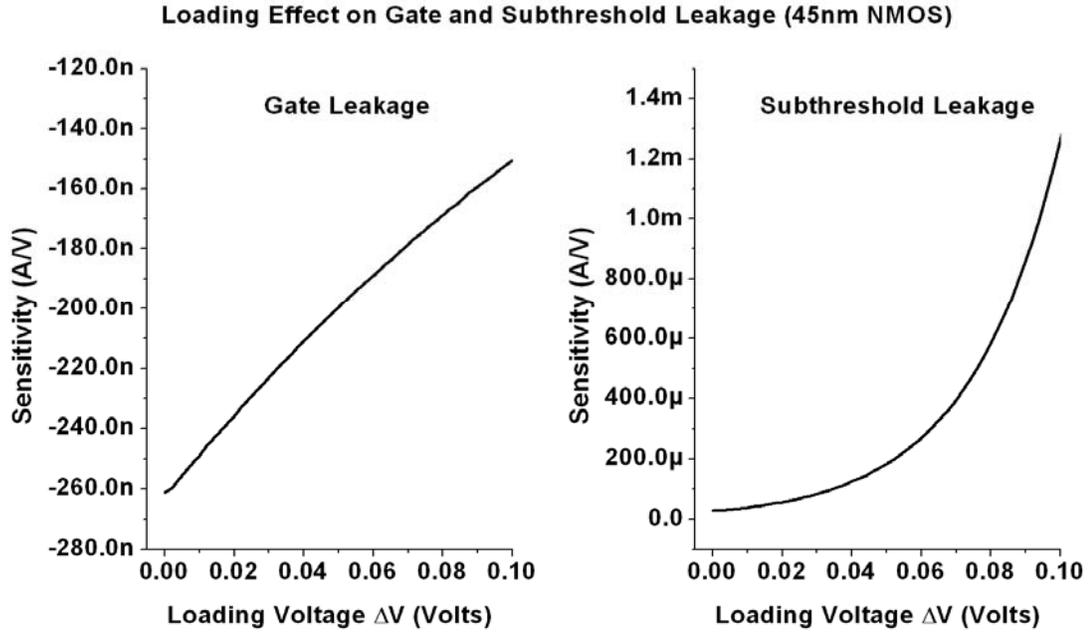


Figure 2.2. Gate leakage (left) and sub-threshold leakage (right) sensitivity versus loading effect in 45nm NMOS device

2.2 Pattern Dependent Leakage Estimation Considering Loading Effect

Leakage estimation is performed on technology mapped circuits. Technology mapped circuits consist of fixed sets of cells. In order to enable fast and efficient estimation, some information is pre-computed for each transistor type while some information is pre-computed for each cell in the cell library. For each transistor, depending on its terminal states, look-up table (Table 2.1) is used for computing gate and band-to-band tunneling leakages. Sub-threshold leakage current is pre-computed from sensitivity plot. For each cell, an output DC current vs. output gate voltage is pre-computed to calculate the

effective drive voltage of a gate for a given load current due to gate leakage from fan-out gates. Consider an example circuit as shown in Figure 2.3 with a NOR gate G1 connected to a number of fan-out gates. Gate leakage from each of the fan-out gates G11-G1j leads to the loading current at the output node of the driver. This increases the gate voltage on the transistors in fan-out gates by ΔV , which causes a change in the sub-threshold and gate leakage current.

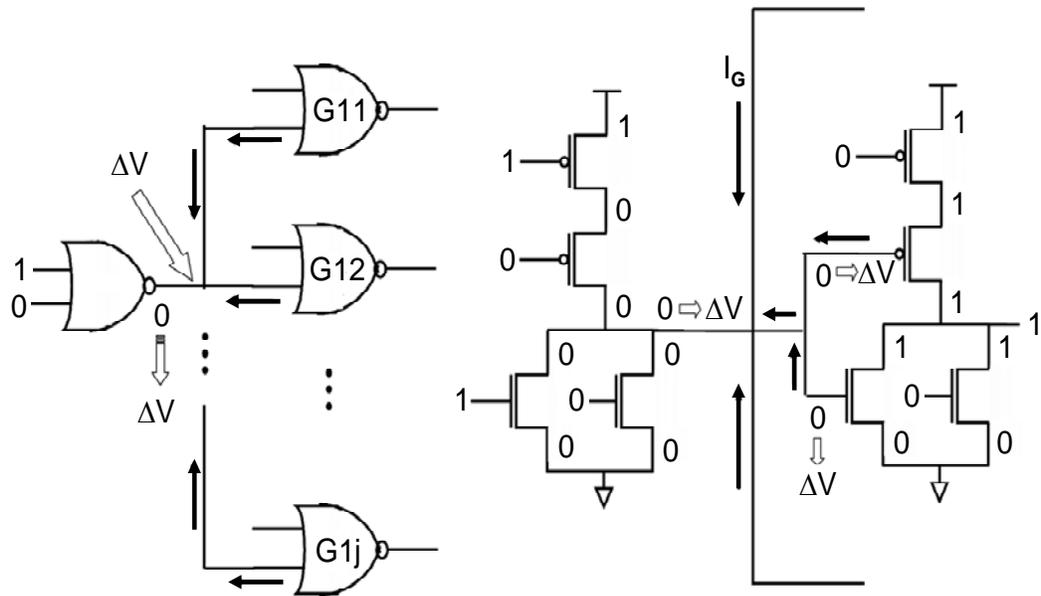


Figure 2.3. Effect of loading gate illustrated at gate level (left) and at transistor level (right) showing the bias states in the fan-out gate

In order to compute ΔV , every cell in the cell library has to be pre-characterized in the following way (Figure 2.4). A set of experiments are performed in HSPICE on a cell driving different loads with output node connected to a current source. If the output of the cell is at logic '0' then the current source supplies current to the output node to force a

small voltage drop ΔV higher than 0V. If cell output is at logic '1' then to force a small voltage drop of $V_{DD} - \Delta V$, the current source draws out current from the output node. For various input combinations of the cell and magnitudes of current source the output voltage is measured and tabulated. Subsequently, a regression analysis is performed on the data and a set of simplified equations are obtained, which quantify the loading voltage as a function of loading current and output load capacitance, which is characterized as number of fan-outs. An example of equations are shown below for a 2-input NOR cell in 45nm with both inputs at logic '1' and logic '0' driving a fixed load.

$$V_L = -8.7302E - 5 + 1854.958 \times I_L \quad (2.1)$$

$$V_L = 0.6009 - 7126.747 \times I_L \quad (2.2)$$

Where I_L is the loading current in micro-amperes and V_L is the output loading voltage in volts. There exist each set of equations for each output load driven by the cell. Several input cell combinations are used as each can create a different strength of conducting path between cell output and its power source and hence more than one equation are needed for each strength [42].

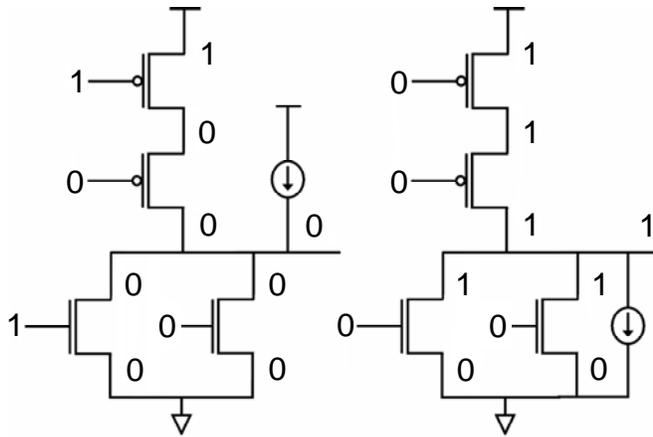


Figure 2.4. Method to compute loading voltage in a cell using SPICE

Figure 2.3 illustrates how the gate voltage in driver gates is driven to ΔV . Once the sink current is computed based on gate leakage it can be translated to ΔV based on pre-computed equations. Figure 2.5 shows the plots of loading voltage for 65 and 45 nm 2-input NOR cell. For 10 μA loading current the change in voltage is at most 60 mV. For larger output load, the loading voltage is lower as larger capacitor can hold the larger amount of charge at a fixed voltage than a smaller capacitor ($Q = CV$). The ΔV computed is used to adjust sub-threshold leakage values of the driven gates. Below we describe our algorithm STABLE (STAtE Based LEakage current estimation) to estimate total leakage after considering loading effect.

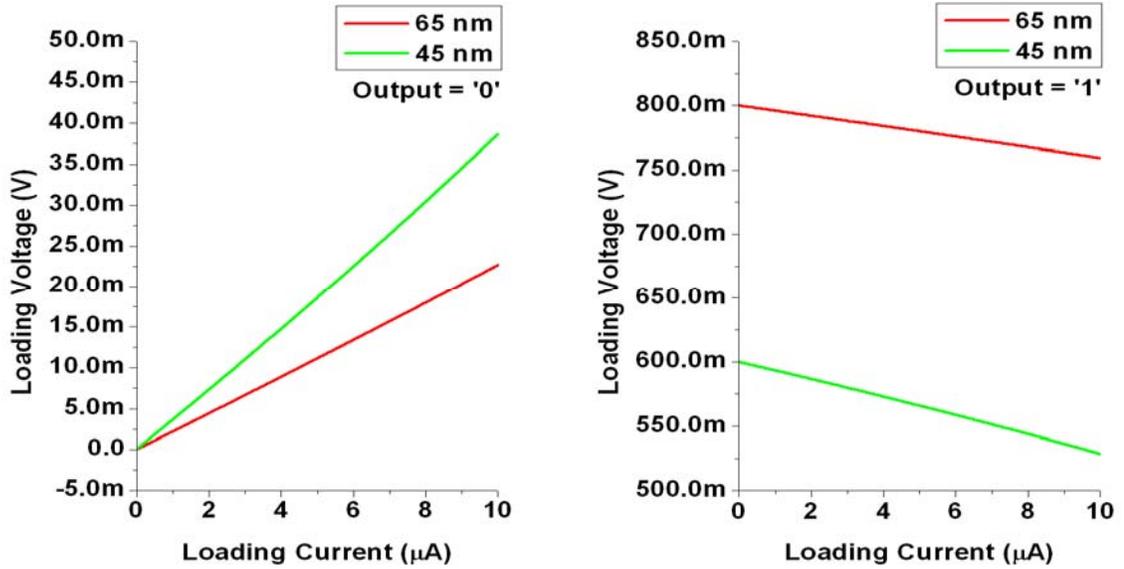


Figure 2.5. Loading voltage in NOR2 cell when output is 0 (left) and when output is 1 (right) in 65nm and 45nm

2.3 Leakage Current Estimation Algorithm

The algorithm has five major steps. Figure 2.6 shows the flowchart for the algorithm with the box numbers indicating the steps mentioned below.

1. Logic simulation: Logic simulation is performed in a levelized order to compute input states of transistors. Information from previous simulation cycles is necessary for correctly resolving the states of internal circuit nodes that have no path to V_{DD} or ground in the current logic simulation cycle.
2. Gate leakage calculation: For each transistor in level n gates, pre-computed gate and BTBT leakage values are used from the state look-up table (Table 2.1). Gate and BTBT leakage from all the transistors in a cell are summed.

3. Driver output voltage calculation: The gate leakages from fan-out gates of a driver are summed to obtain the loading current. Using loading voltage equations (2.1–2.2), the driver output voltage from the loading current is determined.
4. Sub-threshold leakage current calculation: Based on the new gate voltage V_{GS} due to loading at the driver output, sub-threshold leakage is computed and summed for all off devices in the fan-out gates. To account for stack effect, the intermediate node voltage in PMOS stack is calculated by starting with an initial voltage value at $V_{DD}/2$ and then iteratively solving the sub-threshold leakage in the upper and lower PMOS transistors until both are different by 5% or less. This method can be extended for three or more transistors in the stack.
5. Total leakage computation: Sum of sub-threshold, gate and BTBT leakages of each gate is computed.

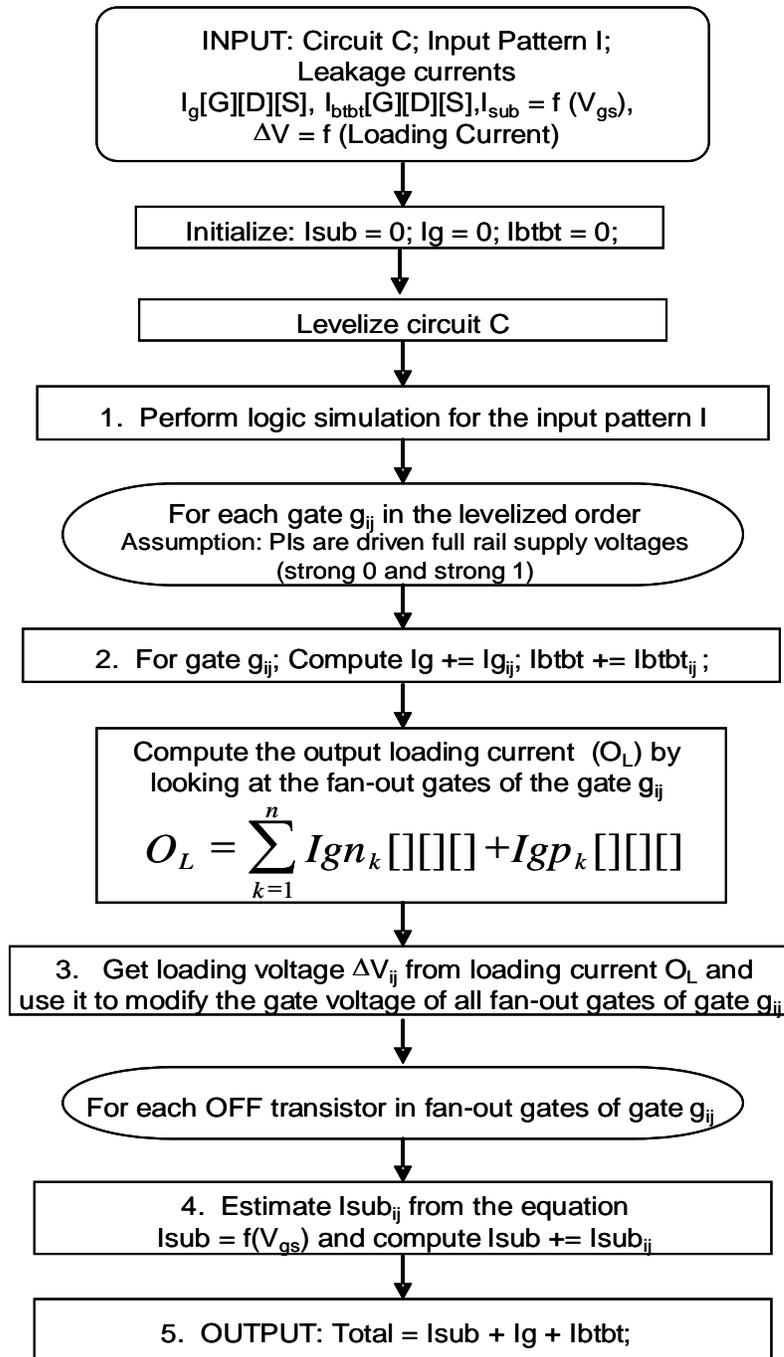


Figure 2.6. STABLE algorithm for estimating overall leakage in a circuit including loading effect

2.4 Results

We conducted experiments on ISCAS85 combinational and ISCAS89 sequential benchmark circuits under nominal temperature $T = 300\text{K}$, $V_{DD} = 0.8\text{V}$ for 65nm and 0.6V for 45nm and nominal channel length. In order to perform leakage calculation on these circuits, it was necessary to perform technology mapping first. To keep it simple, all ISCAS circuits were mapped using a cell library that consisted of two cells, an inverter and a NOR gate. The cells are also made of minimum sized transistors. Technology mapping was performed using SIS [43]. All sequential circuits were simply converted to combinational circuits by changing the inputs to the latch as primary outputs and outputs of the latch to primary inputs. A sequence of 100 random input patterns was used for determining states of internal nodes.

Circuit	I_G max (μA)		ΔV max (mV)	
	65nm	45nm	65nm	45nm
c17	2.18	4.11	0.493	1.150
c432	54.95	103.67	4.699	14.95
c499	98.59	186.07	2.070	6.521
c1355	106.52	200.97	2.241	6.833
c1908	146.31	276.15	4.746	14.97
c2670	213.14	402.33	9.879	32.08
c3540	276.26	521.54	4.704	14.81
c5315	476.31	899.20	5.915	18.84
c6288	364.25	687.21	3.943	12.26
c7552	585.29	1104.91	26.44	87.52
s27	3.91	7.38	0.482	1.545
s298	37.82	71.41	1.768	5.923
s344	40.51	76.48	3.131	10.04
s400	54.99	103.82	1.927	6.179
s510	80.01	151.08	2.168	6.952
s838	137.04	258.74	3.225	10.77
s1196	173.58	327.78	4.336	13.90
s1238	181.56	342.84	4.818	15.45

s1423	156.91	296.27	4.683	15.62
s1494	224.83	424.62	5.323	16.92
s3271	508.39	960.19	6.154	19.31
s3330	322.17	608.44	8.190	26.26
s4863	629.34	1188.55	3.787	11.89
s6669	950.90	1795.93	4.734	14.86
s38417	2667.89	5038.39	16.86	54.06

Table 2.2. Maximum gate leakage current and loading voltage obtained after 100 random pattern simulations by STABLE

Circuit	No loading effect		STABLE Method	
	65nm	45nm	65nm	45nm
c17	4.25	5.77	4.49	6.03
c432	117.29	158.47	121.88	165.38
c499	210.32	283.64	219.11	295.18
c1355	219.68	298.80	245.46	330.26
c1908	321.91	435.07	334.69	453.91
c2670	475.25	637.71	487.85	658.10
c3540	616.02	828.31	657.79	887.10
c5315	1061.62	1424.58	1106.54	1494.08
c6288	746.50	993.12	870.05	1147.11
c7552	1308.73	1754.16	1406.77	2055.81
s27	8.87	12.02	8.75	11.94
s298	87.21	116.25	90.06	120.54
s344	92.87	124.63	93.45	126.52
s400	126.57	169.58	130.61	175.74
s510	186.35	249.85	191.42	258.28
s838	314.58	425.43	323.18	439.34
s1196	404.98	543.67	417.26	565.27
s1238	422.42	566.50	435.06	589.58
s1423	360.39	482.59	373.09	502.06
s1494	528.79	708.84	548.57	750.97
s3330	756.25	1013.32	772.72	1044.78
s3271	1200.02	1613.33	1222.94	1656.72
s4863	1466.24	1969.56	1528.55	2062.51
s6669	2225.61	2988.43	2298.64	3102.81
s38417	6254.45	8348.86	6458.97	8753.49

Table 2.3. Total leakage current in micro-amperes obtained after 100 random pattern simulations with and without loading effect

In Figure 2.7, loading voltage at each circuit node of all the ISCAS circuits over 100 test patterns is shown. From the distribution, loading voltage is usually about 0-1 mV but for some nodes it can go as high as 87.5mV as shown in Table 2.2. In Table 2.3, the total leakage after considering loading effect is higher. For c7552, in 65nm the total leakage is 8% higher and in 45nm total leakage becomes 18% higher. On average, total leakage current after incorporating loading effect increases by 6.9% in 65nm to 7.6% in 45nm due to increase in sub-threshold leakage current. This trend will likely get worse with scaling and therefore it becomes necessary to consider loading effect.

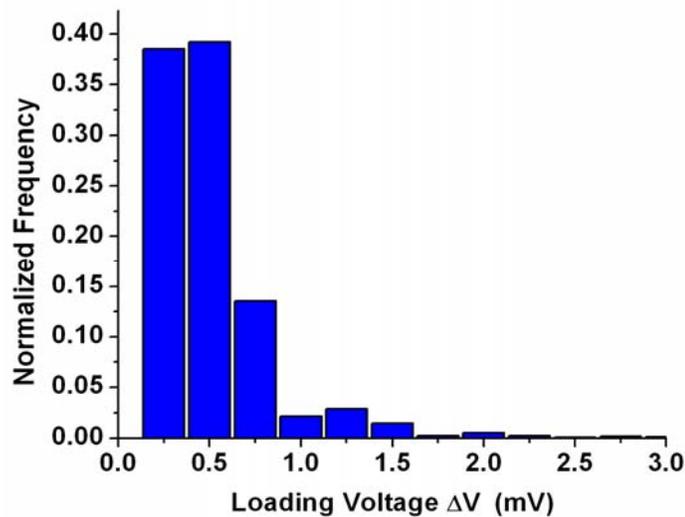


Figure 2.7. Frequency distribution of loading voltage in 45nm circuits

2.5 Validation

Our model was validated using NGSPICE [28]. Since NGSPICE does not model BTBT leakage, our algorithm cannot be directly compared with NGSPICE results. To make validation possible, we re-computed all the information by NGSPICE that are

needed by the algorithm. A state lookup table (Table 2.4) similar to Table 2.1 is created for gate and BTBT leakage by simulating a single NMOS and PMOS device in NGSPICE.

STATES [G][D][S]	I_G Leakage (nA/ μ m)		I_{BTBT} Leakage (nA/ μ m)	
	65nm	45nm	65nm	45nm
[0][0][0]	0	0	0	0
[0][1][0]	-8.43	-25.17	4.4e-03	4.9e-03
[0][0][1]	-8.43	-25.17	4.3e-03	4.8e-03
[0][1][1]	-16.87	-50.34	8.7e-03	9.7e-03
[1][0][0]	24.14	62.38	0	0
[1][1][1]	0	0	8.7e-03	9.7e-03

Table 2.4. Gate Leakage and BTBT Leakage Values for 65nm and 45nm NMOS Device obtained from NGSPICE

For sub-threshold leakage, its sensitivity on loading voltage was used. This was done by measuring the drain current using NGSPICE for small values of gate voltage. The above pre-computed information from NGSPICE was then fed to our STABLE algorithm. The results of the algorithm are then compared with leakage current results obtained by simulating benchmark circuits under NGSPICE. Table 2.5 shows the average total current and the total runtimes obtained after 100 random patterns. Table 2.2 cannot be validated as individual leakage components cannot be extracted in NGSPICE. Also, larger circuits could not be validated because NGSPICE produces a non-converging solution and exhibits extremely slow runtimes.

Circuit	Average Total Leakage Current (μA)		Runtimes (seconds)		% Accuracy	Speedup
	<i>NGSPICE</i>	<i>STABLE</i>	<i>NGSPICE</i>	<i>STABLE</i>		
c17	0.285	0.284	14	0.005	99.65	2800
c432	8.61	7.464	1202	0.040	86.64	30050
c499	15.6	13.57	3479	0.070	87.18	49700
c1355	17.3	13.63	5346	0.075	78.61	71280
c1908	23.9	21.22	7724	0.105	88.70	73562

Table 2.5. Average Total Leakage Current in Micro-Amperes and CPU Runtimes in seconds after 100 Random Pattern Simulation

Performance of STABLE algorithm was compared against NGSPICE for 100 random patterns for some of the benchmark circuits. Both simulations were run on Intel Xeon dual core processor 3.40GHz. For larger circuits such as 16x16 multiplier circuit c6288 having 2400 NOR gates, NGSPICE is not feasible. For smaller circuits STABLE showed 2,000-70,000X speedup over NGSPICE with the additional benefit of estimating all three major sources of leakage with reasonable accuracy.

The algorithm is based on pre-computed transistor leakage tables as well as pre-computed gate output current vs. gate output voltage tables. Avoidance of direct circuit simulation greatly improves speed and capacity of simulation, while accuracy is vastly improved by considering gate leakage, BTBT current and loading effect which are typically not addressed by current generation of commercial spice simulators. Loading effect is shown to increase with scaling.

2.6 Leakage Estimation incorporating Loading Effect using Newton Raphson Method

Gate leakage from fan-out gates was shown to change the driver voltage by ΔV volts. This gate leakage current was overestimated. The new driver output voltage acting on the gate terminal of transistor in the fan-out gates will lower the gate leakage and increase the sub-threshold leakage. Since gate leakage modifies, the loading current at the driver output also gets modified. This way loading effect acts on gate leakage and gate leakage again produces a new loading effect. To model this recursive effect, gate leakage is iteratively computed from new driver voltage and the new driver voltage will be computed from new gate leakage and so on. This will go on until a converging solution is obtained. This iterative computation is called Newton Raphson Method. The above algorithm is modified to include Newton Raphson Method for accurately computing gate and sub-threshold leakage from new driver voltage iteratively.

To account for Newton Raphson Method, the following algorithm is used and the circuit is levelized from output to input to estimate gate leakage only from fan-out gates in level n .

1. Logic simulation: Logic simulation is performed in a reverse levelized order to compute input states of transistors. Information from previous simulation cycles is necessary for correctly resolving the states of internal circuit nodes that have no path to V_{DD} or ground in the current logic simulation cycle (same as in the baseline algorithm).

2. Gate leakage calculation: For each transistor in level n gates, pre-computed gate and BTBT leakage values are used from the state look-up table (Table 2.1). Gate and BTBT leakage from all the transistors in a cell are summed (same as in the baseline algorithm).
3. Driver output voltage calculation: The gate leakages from fan-out gates in level n are summed to obtain the loading current. Using loading voltage equations (2.1–2.2), the driver output voltage from the loading current is determined. For Newton Raphson method, the following steps are added.
4. Gate Leakage Re-computation: For each transistor in the fan-out gate, gate leakage current is computed by using a model that determines the gate leakage as a function of gate, drain and source voltage of the transistor. In this case, steady state gate leakage values cannot be used as voltages can either be at loading voltage ΔV or $V_{DD} - \Delta V$ (Figure 2.8). We create a set of piecewise linear equations for gate current as a function of gate voltage for selected values of drain voltages tuned to deliver highest accuracy for the most frequently encountered ΔV in baseline method as shown in Figure 2.7.

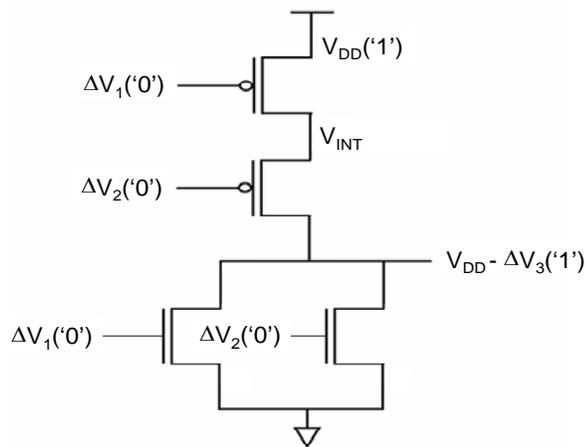


Figure 2.8. Illustration showing possible terminal voltages and logic state of transistors in a 2-input NOR cell

5. Calculating new driver output voltage: New output loading voltage from the loading current is determined from loading current vs. loading voltage table (Figure 2.5). If the difference in loading voltage between two iterations is less than 5%, then the loading voltage from the last iteration is used to estimate sub-threshold current in the next step. Otherwise, steps (a)-(b) are repeated until the solution converges.
6. Gate leakage of driver gate in level $n-1$ is estimated using the new value of loading voltage.
7. Sub-threshold leakage current calculation: Based on the new gate voltage V_{GS} due to new loading at the driver output, sub-threshold leakage is computed and summed for all off devices in the fan-out gates (same as in the baseline algorithm).
8. Total leakage computation: Sum of sub-threshold, gate and BTBT leakages of each gate is computed.

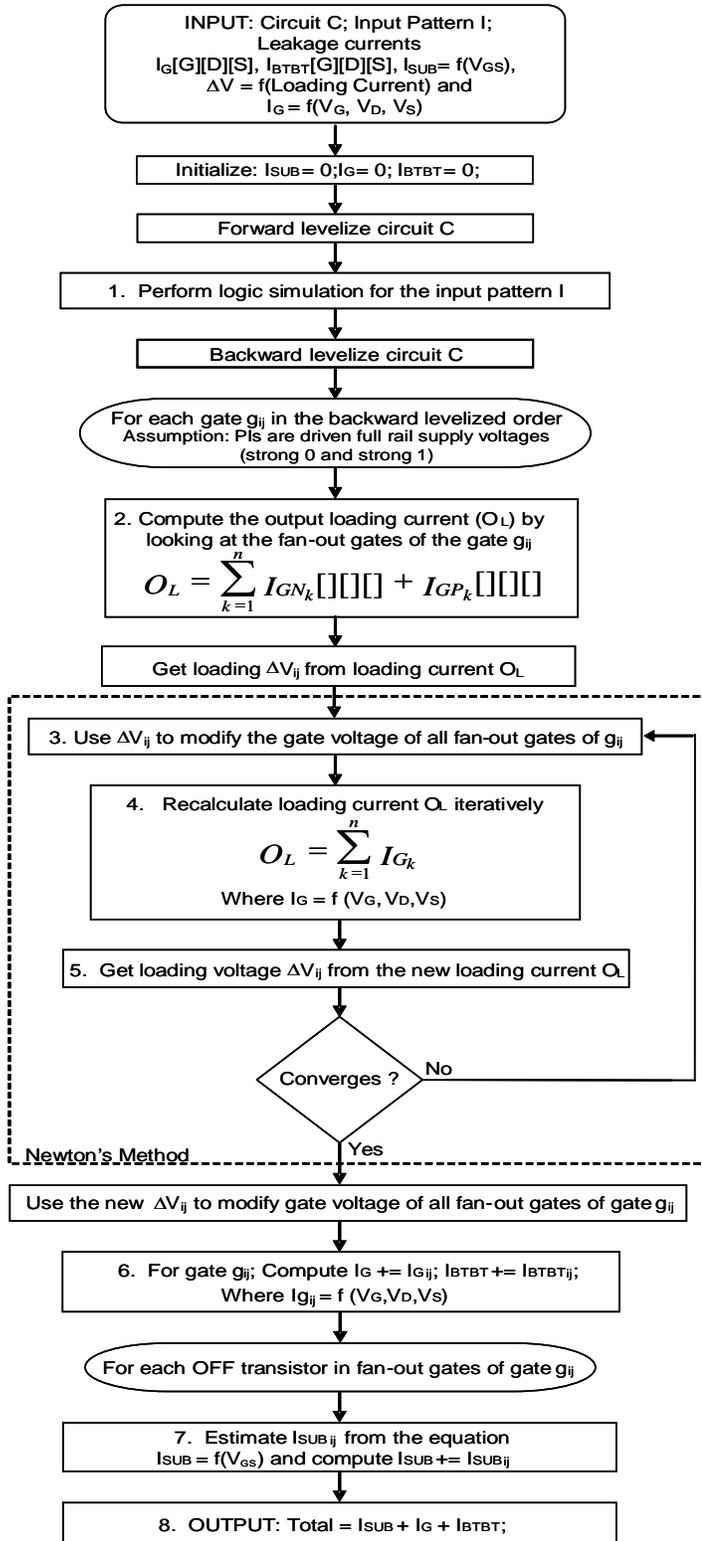


Figure 2.9. STABLE algorithm with Newton-Raphson Method. Part of algorithm outside the dashed box corresponds to the baseline algorithm

2.7 Results with Newton Raphson Method

Experiment was conducted on technology mapped combinational ISCAS85 and sequential ISCAS89 benchmark circuits made of inverter and 2-input NOR cells.

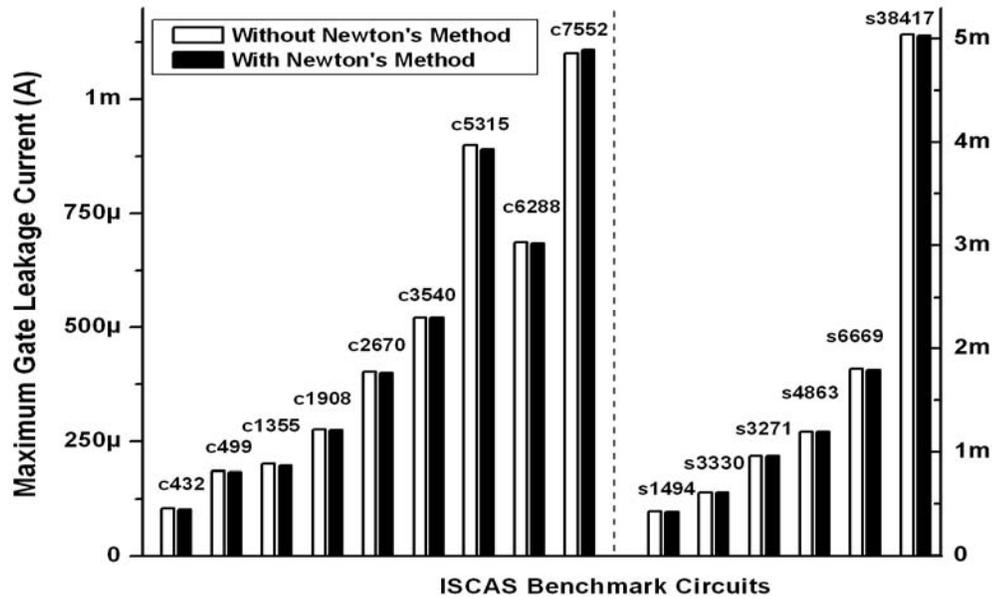


Figure 2.10. Comparison of maximum gate leakage current with and without Newton Raphson Method

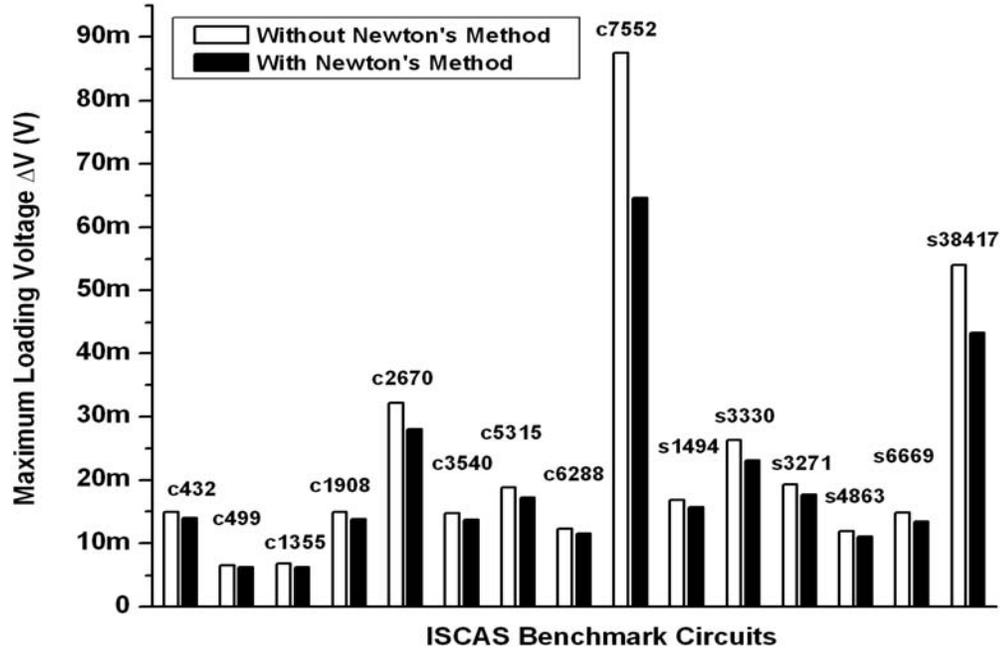


Figure 2.11. Comparison of maximum loading voltage with and without Newton Raphson Method

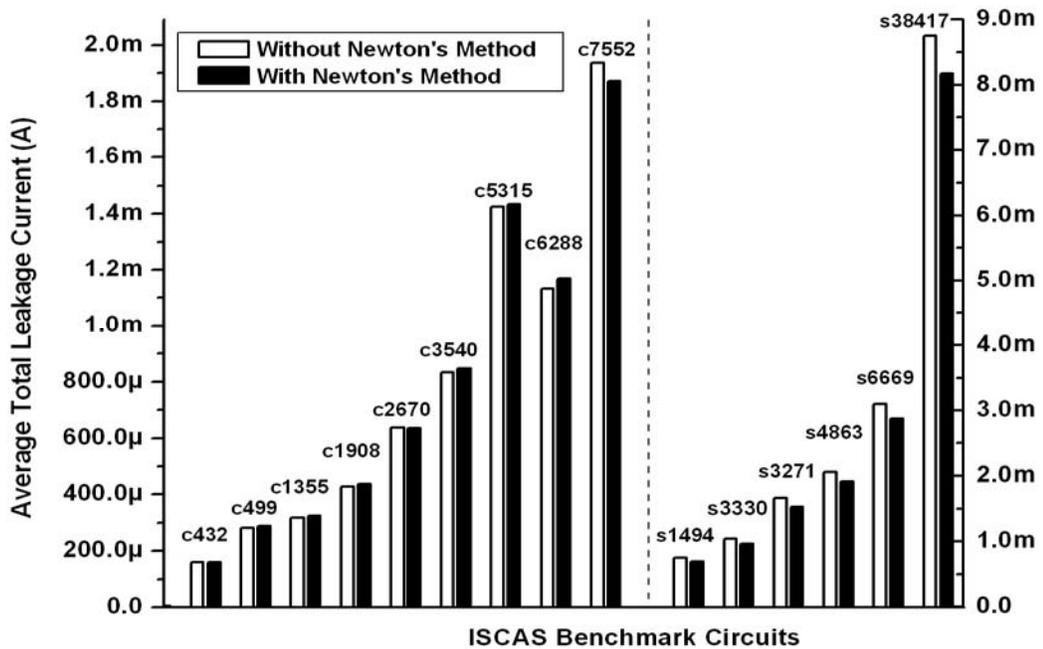


Figure 2.12. Comparison of average total leakage current with and without Newton Raphson Method

Iterative solution of loading effect shows slight decrease in maximum gate leakage current over values obtained from baseline algorithm (Figure 2.10). On average gate leakage decreases in transistors that are driven by ΔV or $V_{DD} - \Delta V$ volts. Lower gate leakage yields to lower loading voltage and eventually a final stabilized solution after Newton Raphson method yields a lower loading voltage. Maximum loading voltage computed using Newton Raphson method shows significant decrease (Figure 2.11), which barely has an implication on total leakage at the circuit level. Figure 2.13 shows the frequency distribution of loading voltages at all the circuit nodes in ISCAS85 and ISCAS89 circuits for 100 random patterns. Most nodes still have 0-1 mV of loading as in the case of loading voltages estimated without Newton Raphson method (Figure 2.7). Hence, average total leakage current with and without Newton Raphson method does show significant differences.

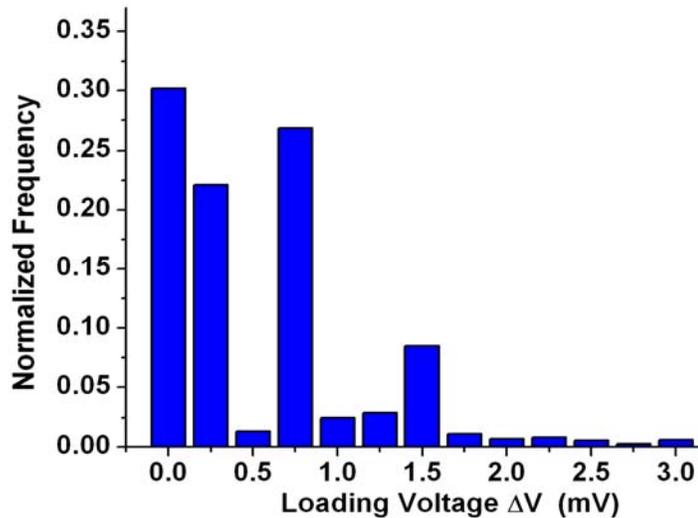


Figure 2.13. Frequency Distribution of Loading Voltage in 45nm circuits after Newton Raphson Method

Circuit	Runtimes			Speedup		Circuit	Runtimes			Speedup	
	NG	ST	STN	ST	STN		NG	ST	STN	ST	STN
c17	14	0.3	3	50	4.7	s344	2475	0.6	7.7	4111	321
c432	1202	7.5	10	161	120	s526	7265	1.2	149	6157	49
c499	3479	14	185	256	19	s820	13326	1.7	228	7885	59
c1355	5346	14	196	392	27	s1196	47061	2.9	324	16007	145
c1908	7724	21	272	364	28	s1269	52776	3.0	357	17890	148

Table 2.6. CPU Runtimes in seconds after 100 Patterns for NGSPICE (NG), STABLE (ST) and with Newton Method (STN)

In this thesis, we have described an algorithm for accurate estimation of total leakage in circuits. The algorithm is based on pre-computed transistor leakage tables. Avoidance of direct circuit simulation greatly improves speed and capacity of simulation, while accuracy is vastly improved by considering gate leakage, BTBT current and loading effect which are typically not addressed by current generation of commercial spice simulators. There are two important points to be noted: (1) Loading effect is shown to increase with scaling. (2) Newton-Raphson method is shown to produce small accuracy improvement with relatively higher cost of computation. The study indicates that Newton-Raphson method is essential for signal integrity (internal node voltage analysis) but is not really necessary for computing overall leakage.

2.8 Conclusions

This thesis work presented the methodology for efficient and accurate leakage current estimation that estimates three major sources of leakage power in 65nm and 45nm technology nodes namely sub-threshold, gate oxide tunneling and band-to-band tunneling leakages. The estimator developed also incorporates loading effect. It considers both sub-threshold leakage loading and gate leakage loading using Newton Raphson Method to accurately estimate leakage power. The estimator was shown to give several thousand factor speed up compared to spice tool and is feasible on fairly large circuits. From the experimental results, loading effect was shown to increase with scaling and can increase the total leakage power by 10-15% for some circuits.

2.9 Leakage Estimation in Field Programmable Gate Arrays (FPGA)

This thesis work develops a technique for leakage estimation in a SRAM based FPGA with island-style architecture shown in Figure 2.14 that is targeted by the Versatile Place and Route (VPR) CAD tool [36][37]. This architecture contains a matrix of logic clusters (logic blocks) connected to each other through programmable switches and channel tracks in between. The logic cluster input and output ports are each connected to channel routing tracks with the help of a programmable connection block. The horizontal and vertical routing channel tracks then connect each other through a programmable switch block. The connection and switch blocks are made up of pass-transistors whose gate terminals are connected to 1-bit SRAM cell. The programmed logic value of the SRAM cells decides the connections. In Figure 2.14, channel tracks with width $W = 4$

tracks are shown. Out of the 4 tracks, 3 tracks can connect to a single logic cluster. This is defined as the flexibility of the connected block, which is $F_c = 3$ in absolute terms and in fractional terms $F_c = \frac{3}{4} = 0.75$. In the switch block, each horizontal track can connect to 3 different vertical tracks and vice-versa. The switch block in Figure 2.14 is said to have a flexibility $F_s = 3$. Depending on the architecture, different flexibilities of connection and switch blocks are possible.

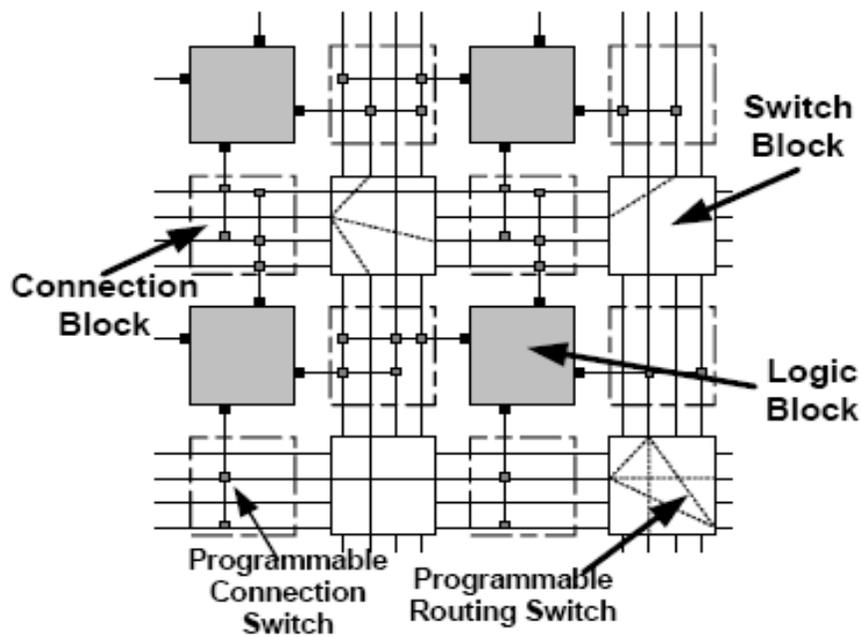


Figure 2.14. Island Style FPGA architecture (adapted from [17])

Each logic cluster is composed of basic logic elements (BLE) as shown in Figure 2.15. A cluster of size N is said to have N number of BLE. The inputs I of a cluster can fan-out to any BLE input within through an input multiplexer. Output of a BLE can also feedback to its input. Cluster size N and inputs I are the two primary parameters that affect logic cluster functionality. Each BLE in this architecture contains one 4-input

lookup table (LUT) and one D flip flop. A 4-input LUT is modeled using a 2-input multiplexer. The inputs to the multiplexer come from SRAM cells. A combinational logic gate can be mapped to a LUT by appropriate programming of these SRAM bits.

Leakage estimation in FPGA reduces to the task of finding leakage current basic components such as 2-input multiplexers, 4-input multiplexers, tri-state buffers and SRAM cells. The compact model developed in section 2.1 can be used for leakage current estimation of each of these components after knowing the states of transistors. Leakage current of a bigger logic block is the sum of leakages of the basic components which make up bigger block. The sum of leakages of all the components gives the total leakage of the design that is mapped, placed and routed on the FPGA.

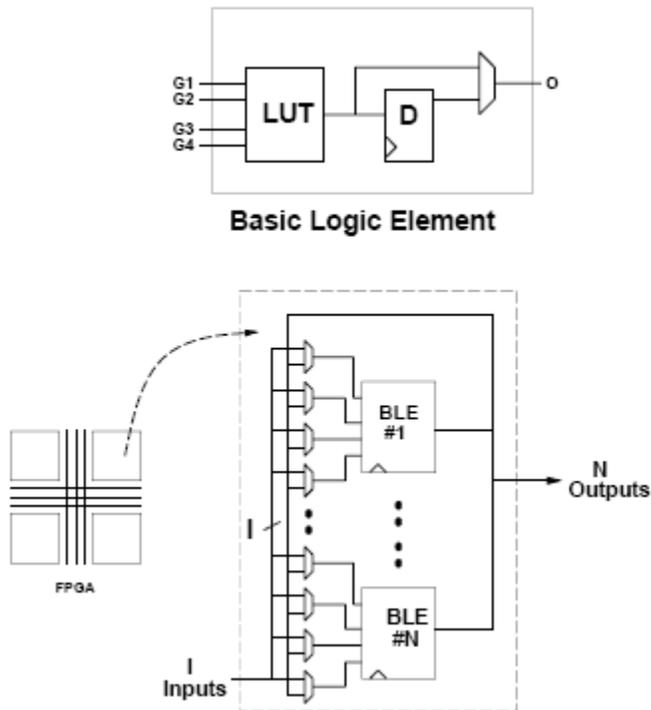


Figure 2.15. BLE (top) and Logic Cluster (bottom) adapted from [38]

Since connection and switch blocks are composed of pass transistors with gate terminal connected to a single SRAM bit cell, gate leakage current from one logic cluster cannot interact with gate leakage currents from other fan-out logic clusters as in ASICs and so there is no gate leakage current flowing into or drawn from the driver cluster output by fan-out clusters. Hence, loading effect cannot happen in FPGA and gate terminal bias at inputs of fan-out clusters will be either at 0 volts or V_{DD} volts. Therefore, sub-threshold leakage current can be pre-computed under steady state bias conditions unlike in section 2.1. Table 2.7 shows leakage current values of a transistor for each state. This table will be used for leakage current computation in FPGAs.

STATES [G][D][S]	I_{SUB} (nA/ μ m)	I_G (nA/ μ m)	I_{BTBT} (nA/ μ m)
[0][0][0]	0	0	0
[0][1][0]	31.93	-8.83	54.6
[0][0][1]	-31.93	-8.83	54.6
[0][1][1]	0	-17.65	109.3
[1][0][0]	0	25.27	0
[1][1][1]	0	1.6E-7	109.3

Table 2.7. Leakage current values of NMOS device in 65nm used for leakage estimation in FPGA

2.9.1 Efficient Leakage Estimation in VPR CAD Tool

In this work, the leakage estimation tool is developed and integrated into VPR CAD tool. VPR already contains a Power Model that computes dynamic and sub-threshold leakage power for island-style SRAM based FPGAs. The Power Model consists of two modules: activity estimator and power estimator [18]. Activity estimator is used to determine the switching activity at each node of the circuit. This switching activity

information is then used by the power estimator module to compute dynamic power. The activity estimator itself employs transition density model to generate transition and static probabilities. To estimate leakage power we will only focus on the static probability. Power Model gives the static probability of each circuit node being at logic high [18]. Since actual logic values are unknown, leakage current will be a statistical quantity. A table for each component is created to store the total leakage current for all the input patterns applied. Each entry of the table is then multiplied by the corresponding static probabilities of the inputs to reflect the fact that for a given input (probability) the following leakage is possible. Sum of leakages of all possible inputs is the total leakage current of that component. Below we discuss the leakage computation in each of the several primary components such as 2-input multiplexers, inverters, tri-state buffers, SRAM cells, 4-input LUT and BLE.

2.9.2 2-input Multiplexer Leakage Power Computation:

In a 2-input multiplexer shown in Figure 2.16, each of the 2 inputs has a static probability $P(x)$ that is the probability of node x being at logic high and $1-P(x)$ of node x being at logic low. For each of $2^2 = 4$ input combinations the states of the transistors are determined. State of SRAM bit is fixed and depends on the design mapped. Once the state of each transistor is known, our leakage compact model shown in Table 2.7 is used to get gate, band-to-band leakage and sub-threshold leakage. For an input combination of '01' and with SRAM bit at logic '0', input 0 will be the output of the 2-input multiplexer. Then the states of transistor A and B will be:

A \Rightarrow [001] and B \Rightarrow [111], which is in format [Gate Drain Source]

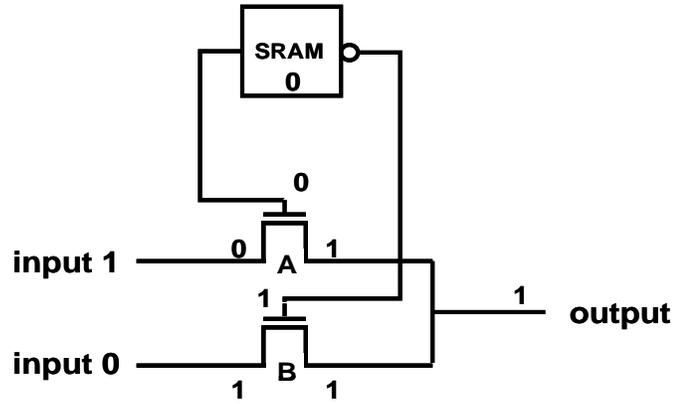


Figure 2.16. 2-Input multiplexer with inputs = '01'

Total gate leakage for the circuit is then multiplied by the static probability to reflect the fact that the gate leakage depends on the state of the 2-input multiplexer. Below is the equation for gate leakage if the inputs are <01>:

$$\begin{aligned}
 I_{mux2}(i_1 = 0; i_0 = 1; s = 0) &= P(i_1 = 0) \times P(i_0 = 1) \\
 &\times \{I_{subA}([001]) + I_{gA}([001]) + I_{bitA}([001]) \\
 &+ I_{subB}([111]) + I_{gB}([111]) + I_{bitB}([111]) + I_{SRAM}(0)\}
 \end{aligned} \tag{2.3}$$

Where $I_{gA}([001])$ and other currents are the values obtained from the leakage compact model (Table 2.7) times the effective transistor width W_{eff} . $I_{SRAM}(0)$ is the total leakage current of SRAM cell with bit '0'. $P(i_0 = 1)$ and $P(i_1 = 0)$ is the probability of input i_0 being at logic '1' and input i_1 being at logic '0' which are determined by the Power Model.

2.9.3 Inverter Leakage Power Computation:

Estimating leakage for an inverter is crucial as it is not only present in SRAM but also as buffers in the routing (switch and connection block) and in the logic block. By knowing the states of the transistors in inverters, our leakage compact model can be used.

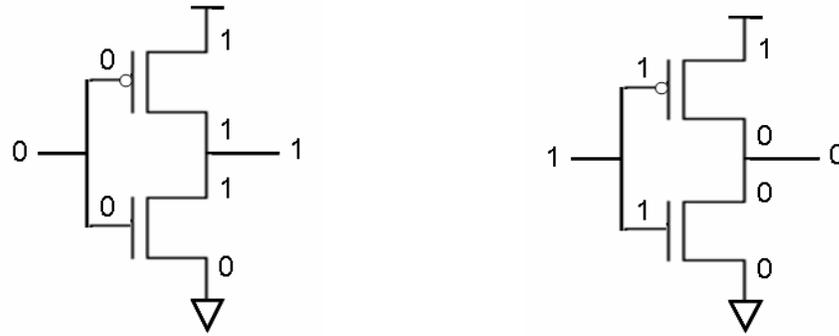


Figure 2.17. Transistor states in an Inverter with input = 0 and input = 1

From Figure 2.17, the following total leakage current equations for input = 0 and 1 are:

$$\begin{aligned}
 I_{inv}(i=0) &= P(i=0) \times \{I_{subN}([010]) + I_{gN}([010]) + I_{btbN}([010]) \\
 &\quad + I_{subP}([011]) + I_{gP}([011]) + I_{btbP}([011])\} \\
 I_{inv}(i=1) &= P(i=1) \times \{I_{subN}([100]) + I_{gN}([100]) + I_{btbN}([100]) \\
 &\quad + I_{subP}([101]) + I_{gP}([101]) + I_{btbP}([101])\}
 \end{aligned} \tag{2.4}$$

Where, the subscript N is NMOS and P is PMOS. All these current values are plugged in from Table 2.7 and multiplied by the effective transistor width W_{eff} . The Power Model makes an assumption of similar NMOS and PMOS device sizing. The same device widths are used for obtaining the total leakage power values.

Input	Total Leakage Power (μW)
0	0.068
1	0.0347

Table 2.8. Inverter Leakage Power

2.9.4 SRAM Leakage Power Computation:

Leakage of SRAM cell is almost independent of its bit value. Most of the time Read signal is asserted. Write signal is asserted initially when programming the SRAM. For any value of Data, the feedback inverter loop will exhibit the same leakage. This is because for either inverter will have input = 0 and output = 1 or input = 1 and output = 0. So leakage of SRAM just depends on the state of the pass transistor. If Data = '0' is written, the state of the pass transistor is [000]. In this state there is no leakage. If Data = '1', then the state is [011]. This is shown in Figure 2.18.

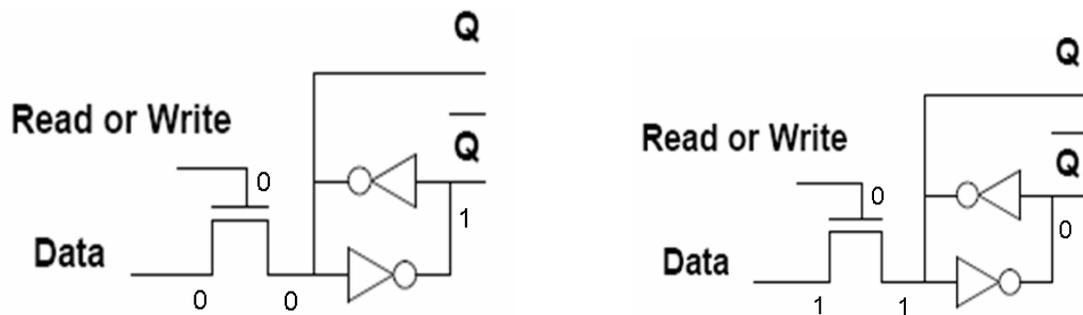


Figure 2.18. 1-bit SRAM with Data = 0 (left) and Data = 1 (right)

Total leakage for a SRAM bit by looking at the transistor states in Figure 2.17 and Figure 2.18 is given as:

$$\begin{aligned}
I_{SRAM}(0) &= I_{inv}(i=0) + I_{inv}(i=1) \\
I_{SRAM}(1) &= I_{subN}([011]) + I_{gN}([011]) + I_{btbN}([011]) \\
&\quad + I_{inv}(i=0) + I_{inv}(i=1)
\end{aligned} \tag{2.5}$$

Leakage power of SRAM cell is tabulated below:

Data	Total Leakage Power (μW)
0	0.1032
1	0.1147

Table 2.9. SRAM Leakage Power

Since the states of SRAM are fixed and determined by the design mapped to FPGA, static probabilities are not considered. The leakage power values for SRAM can be plugged in $I_{SRAM}(0)$ term in leakage power equation (2.3) for 2-input multiplexer. For 4 possible input states and 2 SRAM states, leakage values are computed. These values are tabulated in Table 2.10, which are then multiplied by the corresponding input static probability to get 2-input multiplexer leakage power.

Input	SRAM bit	Total Leakage Power (μW)
00	0	0.105
00	1	0.116
01	0	0.163
01	1	0.166
10	0	0.154
10	1	0.175
11	0	0.125
11	1	0.136

Table 2.10. 2-input Multiplexer Leakage Power for all possible states

2.9.5 4-input LUT Leakage Power Computation:

A 4-input LUT is implemented using 2-input multiplexer tree. Using leakage power values in Table 2.10, leakage power can be calculated for 4-input LUT after knowing the 2-input multiplexer states. Since 4-input LUT has more than 16 2-input multiplexers and is more complex, leakage estimation of 2-input LUT is shown.

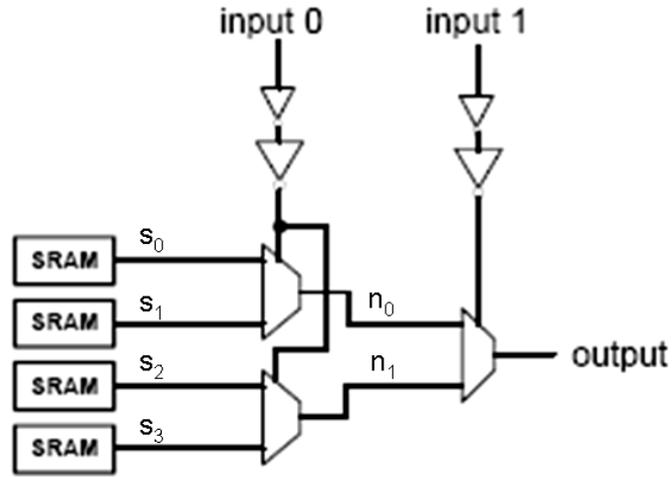


Figure 2.19. 2-input LUT implemented using 2-input multiplexers (adapted from [17])

Power Model calculates the static probability for each multiplexer select signal input 0, input 1 and internal nodes n_1 , n_2 . Consider the case with SRAM configuration $s[3:0] = 0011$ and input = 01, leakage current will be:

$$\begin{aligned}
 I_{LUT2}(s_3 = 0; s_2 = 0; s_1 = 1; s_0 = 1; i_1 = 0; i_0 = 1) = \\
 P(i_0 = 0) \times \{I_{1mux2}(s_1 = 1; s_0 = 1; i_0 = 1) + I_{2mux2}(s_3 = 0; s_2 = 0; i_0 = 1)\} \\
 + P(n_1 = 0) \times P(n_0 = 1) \times P(i_1 = 0) \times I_{3mux2}(n_1 = 0; n_0 = 1; i_1 = 0)
 \end{aligned} \quad (2.6)$$

Where, I_{mux2} is the total leakage of a 2-input multiplexer for a given state. Here a table of total leakage values is not constructed. Based on the SRAM bits and probabilities of select signals the appropriate leakage power of 2-input multiplexer is used from Table 2.10.

2.9.6 4-input Input Multiplexer Leakage Power Computation:

Similar analysis can be done on a 4-input multiplexer shown in Figure 2.20. This 4-input multiplexer is implemented as a 2-input multiplexer tree. For input combination of ‘0001’ and SRAM bits ‘00’, leakage is given as:

$$\begin{aligned}
 I_{mux4}(i_3 = 0; i_2 = 0; i_1 = 0; i_0 = 1; s_1 = 0; s_0 = 0) = \\
 P(i_1 = 0) \times P(i_0 = 1) \times I_{1mux2}(i_1 = 0; i_0 = 1; s_0 = 0) \\
 + P(i_3 = 0) \times P(i_2 = 0) \times I_{2mux2}(i_3 = 0; i_2 = 0; s_0 = 0) \\
 + P(n_1 = 0) \times P(n_0 = 1) \times I_{3mux2}(n_1 = 0; n_0 = 1; s_1 = 0)
 \end{aligned} \tag{2.7}$$

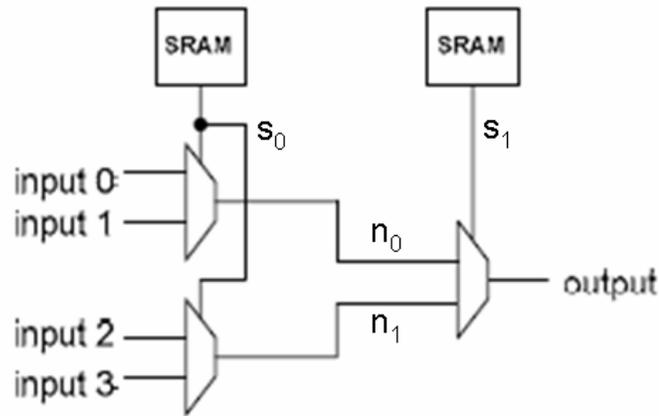


Figure 2.20. 4-input multiplexer implemented using with 2-input multiplexer tree (adapted from [17])

2.9.7 Logic Block Leakage Power Computation:

Finally, the leakage power of a logic block or logic cluster can be computed by summing the leakage power of each component calculated above.

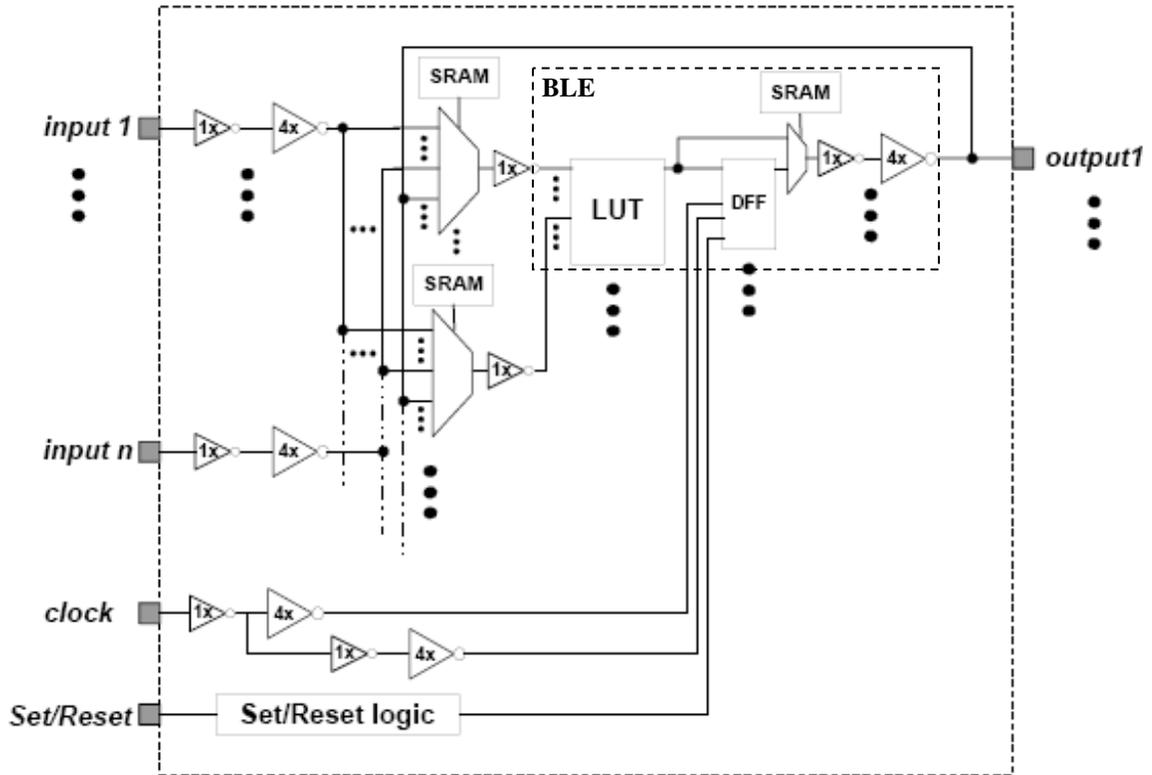


Figure 2.21. Combinational Logic Block (CLB) or logic cluster (adapted from [17])

In a logic block with size $N = 4$, inputs $I = 10$ and with 4-input LUT in a BLE, there are sixteen 4-input multiplexers that connect one of the ten inputs to single input of a BLE and there are a total of four BLEs. The total leakage power of a CLB is given as:

$$\begin{aligned}
I_{clb} = & \{P(i_1 = x) + P(i_{1+1} = x) + \dots + P(i_{10} = x)\} \times (I_{inv1X} + I_{inv4X}) \\
& + 16 \times P(i_{mux4} = x) \times I_{mux4} + 16 \times P(o_{mux4} = x) \times I_{inv1X} \\
& + 4 \times P(i_{LUT4} = x) \times I_{LUT4} + 4 \times I_{mux2} + 4 \times (I_{inv1X} + I_{inv4X}) \quad (2.8)
\end{aligned}$$

$$P_{clb} = I_{clb} \times V_{dd}$$

Where, I_{inv1X} and I_{inv4X} are the total leakage values of minimum sized inverters and inverters 4 times the size of minimum sized inverter, i_{mux4} are the inputs/output of the logic block that are inputs to the 4-input multiplexer, o_{mux4} is output of 4-input multiplexer, I_{mux4} is total leakage current of 4-input multiplexer for a given state, i_{LUT4} are inputs of 4-input LUT, I_{LUT4} is the total leakage current of 4-input LUT for a given state and x is the state can be either logic 0 or 1. This is a simplified equation that does not show the states of each component.

2.9.8 Clock Network Leakage Power Computation:

Clock network and flip flops are not considered in leakage power as clock signal has a toggle rate of 100% and exhibit dynamic power only.

2.9.9 Routing Leakage Power Computation:

Routing fabric in FPGA consists of connection blocks and switch blocks. It is through these blocks power is dissipated due to routing. This work targets Subset type switch block, which is shown in Figure 2.22. The number of pass transistors in the switch block depend on the switch block flexibility F_S . Here subset type switch box with a flexibility

of $F_S = 3$ is considered. Connection block helps connect the logic block to the tracks and number of pass transistors is also sensitive to connection block flexibility F_C .

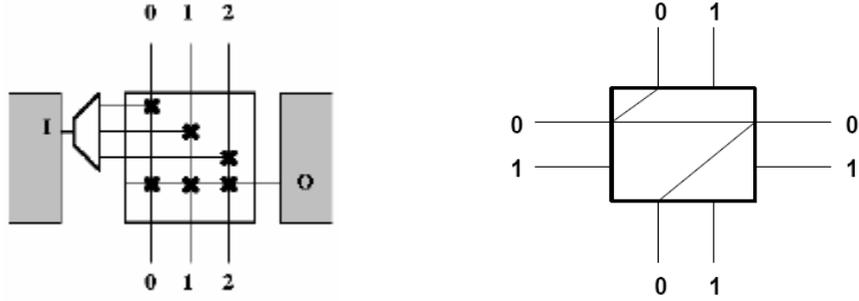


Figure 2.22. Connection block (left) and Switch block (right)

Power Model uses the routing architecture that has 50% pass transistors switched wires and 50% tri-state buffers switched wires. The routing architecture implemented is shown below in Figure 2.23. Each logic block input connecting to the channel has a static probability density of 0.5 assumed by the Power Model. By using the leakage power for pass transistors and buffers, the total leakage power due to routing is:

$$\begin{aligned}
 I_{routing} &= I_{connection_block} + I_{switch_block} \\
 I_{connection_block} &= 0.5 \times N_{pass_trans}^{connection_block} \times P(i = x) \times \{I_{sub} + I_g + I_{btbt}\} \\
 &\quad + 0.5 \times N_{tri-state_buffer}^{connection_block} \times P(i = x) \times I_{inv5X} \\
 I_{switch_block} &= 0.5 \times N_{pass_trans}^{switch_block} \times P(i = x) \times \{I_{sub} + I_g + I_{btbt}\} \\
 &\quad + 0.5 \times N_{tri-state_buffer}^{switch_block} \times P(i = x) \times I_{inv5X}
 \end{aligned} \tag{2.9}$$

$$P_{routing} = I_{routing} \times V_{dd}$$

Where, $P(i=x)$ is the static probability of logic block input = 0.5 assumed by Power Model, $N_{pass_trans}^{connection_block}$ is the number of pass transistors in a connection block, $N_{tri-state_buffer}^{connection_block}$ is the number of tri-state buffers in the connection block. These numbers are computed by the VPR CAD tool. I_{sub} , I_g , I_{btbt} are the leakage currents in a pass transistor and I_{inv5X} is the total leakage current of the tri-state buffer.

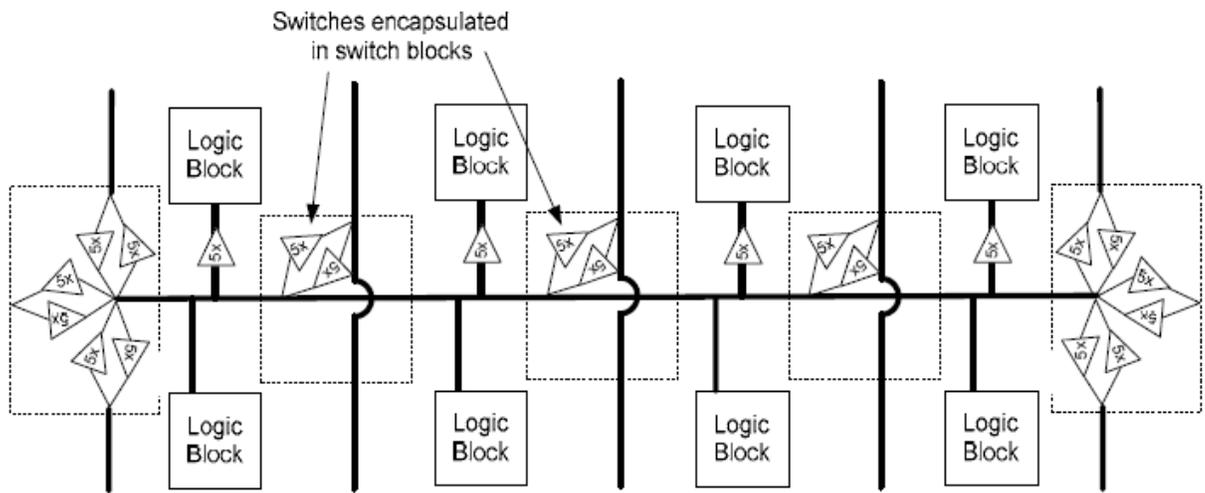


Figure 2.23. Routing Segment (adapted from [19])

2.9.10 Results:

Experiment was conducted on MCNC benchmark circuits. These circuits were mapped, placed & routed on an island style FPGA architectures with different cluster size, number of inputs, connection box flexibilities at $T = 298K$ and $V_{DD} = 0.8V$.

2.9.10.1 Impact of technology scaling:

The original Power Model estimated sub-threshold leakage power only based on device parameters from TSMC 0.18 μ m process and the modified Power Model now takes into account gate and band-to-band tunneling leakage with the modified sub-threshold leakage from BPTM 65nm process technology. Figure 2.24 shows the increase in total leakage power as transistors are scaled from 180nm to 65nm technology. From Figure 2.24, the total leakage power has increased by $\sim 6X$, routing power has increased by $\sim 5X$ and logic block power has increased by $\sim 15X$ by scaling the transistors from 180nm to 65nm.

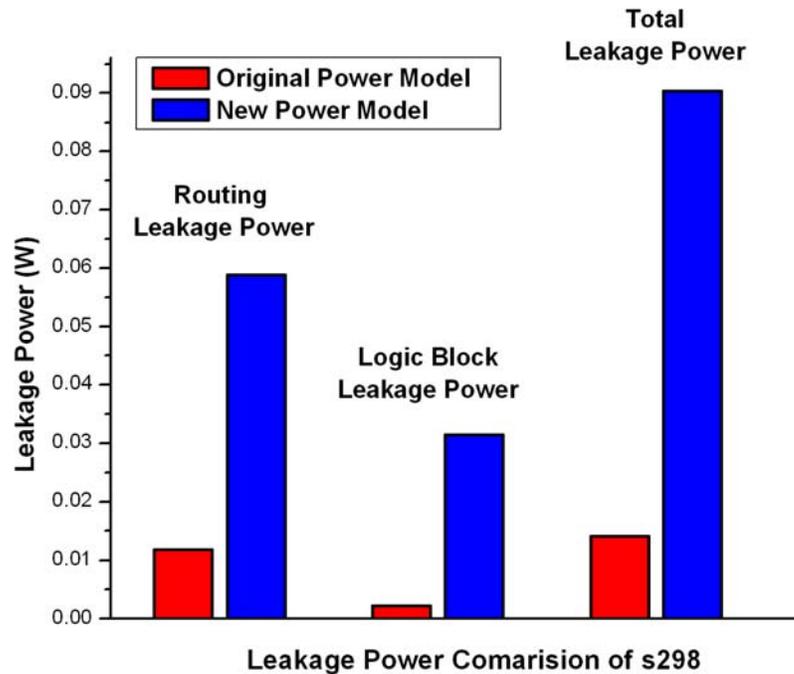


Figure 2.24. Comparison of Leakage Power estimated from original and modified Power Model for each resource type in s298 design

Furthermore, considering sub-threshold leakage power only is not enough. For 65nm, gate leakage and band-to-band leakage have become significant. Figure 2.25 shows comparison between sub-threshold leakage power and total leakage power for five benchmark circuits. It was found that on average the gate and band-to-band tunneling leakage increase leakage power by 54.27%.

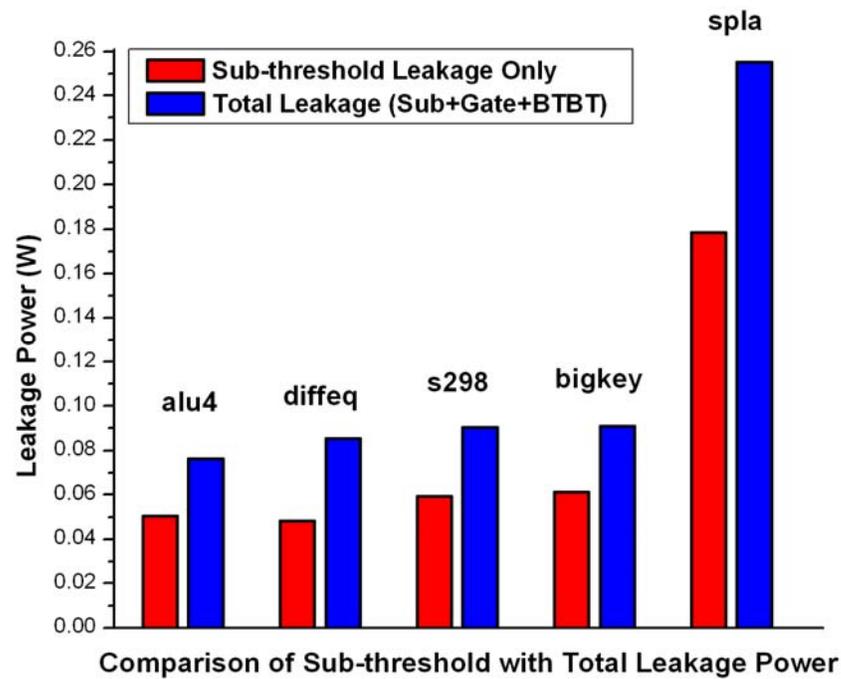


Figure 2.25. Sub-threshold and Total Leakage Power Comparison

2.9.10.2 Leakage Power Vs FPGA Resource Utilization:

The impact of FPGA resource utilization on total leakage power for five benchmark circuits of different sizes are shown. These circuits were mapped, placed and routed by VPR CAD tool with minimum channel width W_{min} , which is the minimum number of

tracks per channel required to successfully route a circuit. The following parameters were used: cluster size $N = 4$, inputs $I = 10$, 4-input LUT, switch block flexibility, $F_s = 3$ and connection block flexibilities for logic cluster inputs and outputs, $F_{c_input} = 0.5$ and $F_{c_output} = 0.25$. These settings ensure close to 98% utilization of FPGA resources [38].

Circuit	Logic Clusters	Nets	Array Size	Transistor Count		W_{min}	Leakage Power (W)		
				Logic	Routing		Logic	Routing	Total
apex4	335	959	19x19	514,895	1,152,312	41	0.0208	0.0589	0.0797
dsip	343	920	27x27	527,191	1,558,602	27	0.022	0.0759	0.0979
alu4	390	1019	20x20	599,430	1,025,600	33	0.0248	0.0513	0.0762
diffeq	379	1180	20x20	582,523	968,800	31	0.024	0.0489	0.0729
apex2	485	1408	23x23	745,445	1,688,568	41	0.3076	0.0825	0.1132
s298	490	1012	23x23	753,130	1,206,120	29	0.0315	0.0589	0.0904
bigkey	427	1037	27x27	656,299	1,351,566	23	0.0275	0.0634	0.0909
spla	955	2539	31x31	1,467,835	3,997,630	55	0.0609	0.1943	0.2552
s38417	1609	5046	41x41	2,473,033	5,127,050	39	0.1028	0.2353	0.3381
clma	2133	6134	47x47	3,278,421	8,829,373	52	0.1373	0.4061	0.5434

Table 2.11. Total leakage power vs. FPGA resource utilization

Table 2.11 shows total number of FPGA logic clusters required to map the entire circuit, number of routing nets, FPGA array size, total number of transistors used in logic and routing and the minimum number of tracks required for connecting all the logic blocks. It can be seen that resource utilization in FPGAs greatly affects the leakage power. Designs with larger number of logic blocks contain for transistors and dissipate more leakage power due to logic. The FPGA array size is the size of the FPGA device needed to implement the design. The device size is usually bigger than the actual design. Larger device dimensions are needed because of design I/O limitations. To map larger number of design I/Os a larger device is required. For example, in bigkey, the device

contains $27 \times 27 = 729$ logic clusters out of which only 427 logic clusters (58.6%) are utilized in bigkey logic. Larger design needs more routing resources (switch and connection blocks) to achieve 100% routability between large numbers of logic blocks. Hence, routing leakage power also depends on the number of logic blocks. For the circuit bigkey, routing leakage power is more than that of s298 even though it requires a smaller W_{\min} . However, bigkey requires a larger device array size in routing, and hence more transistors are used up in routing than s298.

2.9.10.3 Leakage Power Vs FPGA Resource Type:

Table 2.12 shows average total leakage power dissipated by FPGA resource type: Logic element (consisting of 4-input LUTs, input multiplexers), connection block and switch block (buffers and pass transistors).

Resource Type	Total Leakage Power over 10 designs
Logic Block	37.31 %
Switch and connection block (routing)	62.69 %

Table 2.12. Leakage Power dissipated by each resource type

This shows that routing in FPGA contributes to much higher leakage power than logic blocks. The reason for this is that number of transistors in the routing fabric is much higher than in logic blocks.

2.9.10.4 Effect of Number of Clusters N on Leakage Power:

Cluster size N is varied and inputs I to the cluster are selected based on the equation

$$I = \frac{K}{2}(N + 1) \text{ in [38], where } K \text{ is the number of inputs to LUT. This equation has}$$

shown to produce 98% or more utilization of basic logic elements for several designs.

Below shown is the routing leakage power, logic block leakage power for 10 designs that

were placed and routed. Here $K = 4$, $F_{c_input} = 0.5$, $F_{c_output} = 0.25$ and $F_s = 3$ is used.

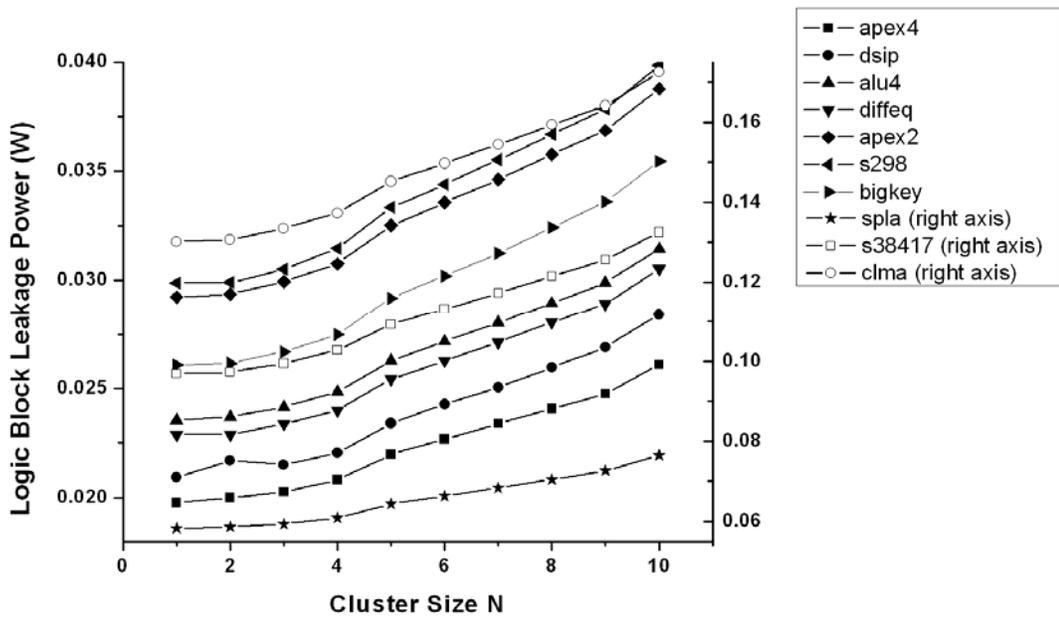


Figure 2.26. Logic block leakage power for designs placed & routed with different cluster sizes

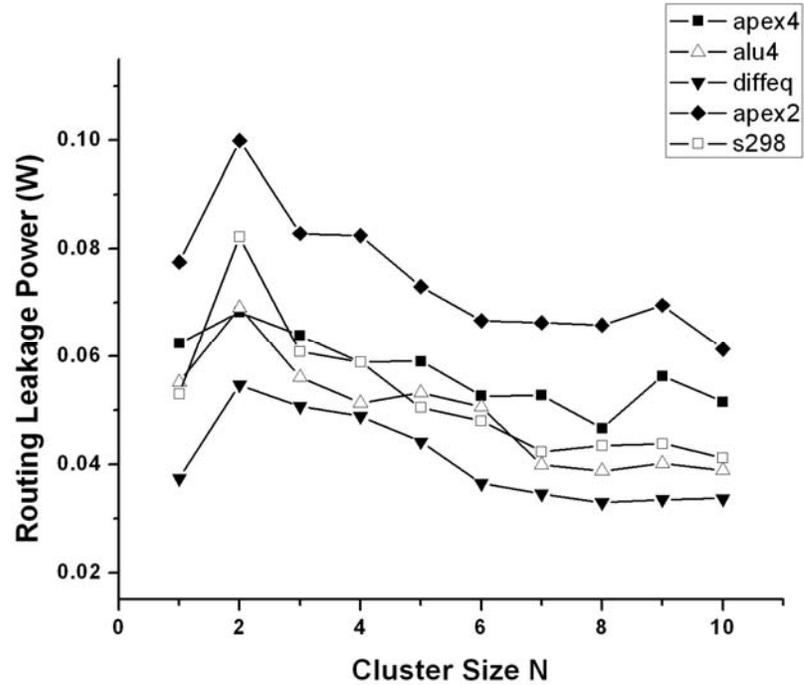


Figure 2.27. Routing leakage power for designs placed & routed with different cluster sizes

From Figure 2.27, the logic block leakage power increases as cluster size is increased. This is primarily due to increase in intra-cluster area. By increasing the cluster size, more transistors are packed inside, which reduces the number of clusters needed for mapping the design. This helps decrease inter-cluster area. However, unused clusters for a given design consumes more transistors in case of bigger clusters and hence the increase in logic power. However, routing leakage power is shown to decrease as cluster size is increased. Increasing cluster size leads to larger W_{\min} as more inputs are used by larger clusters and all these signals have to be routed. Larger W_{\min} requires a bigger switch box which consumes more transistors. Also the connection block flexibility is increased for larger clusters requiring larger connection boxes. But at the same time, device array size is reduced as there are smaller numbers of clusters. Thus, fewer numbers of bigger

connection and switch blocks are needed to achieve full routability. The net result is a decrease in number of transistors used in routing. Hence, there is decrease in routing leakage power. As cluster size is increased from $N=1$ to $N=2$, the number of clusters decrease but not enough to overcome the increase in W_{\min} and the number of transistors consumed in bigger switch/connection blocks.

2.9.10.5 Effect of Number of Clusters Inputs I on Leakage Power:

For a cluster of size $N = 4$, with several inputs, the experiment is conducted. Number of LUT inputs, $K = 4$, connection block $F_{c_input} = 0.5$, $F_{c_output} = 0.25$ and switch block $F_s = 3$ is used.

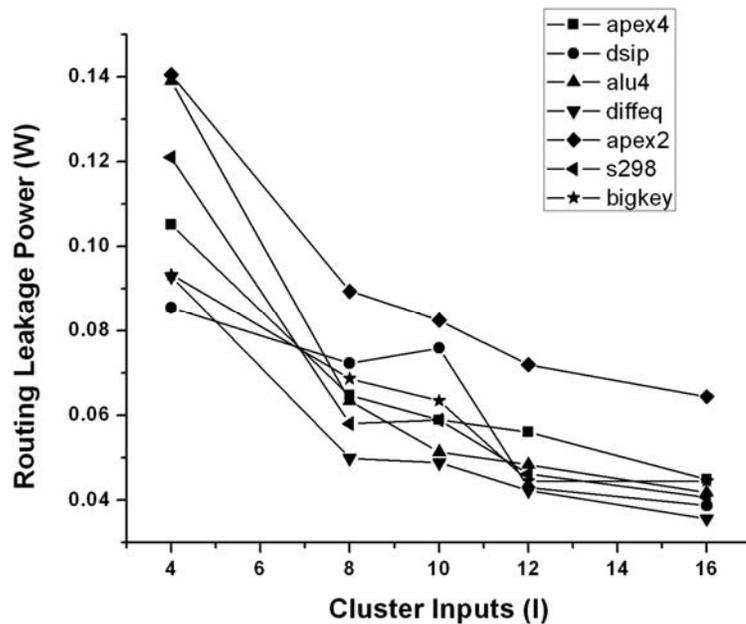


Figure 2.28. Routing leakage power for designs placed & routed with cluster of size $N=4$ and of different number of inputs

It was observed that increasing the number of cluster inputs has no impact on logic leakage power. However, routing leakage power reduces drastically as device array size reduces. It was shown before that often larger devices are required to implement a design whose actual logic only consumes a part of the device. Rest of device is used in mapping the signal pins. By increasing the inputs to the cluster, device size reduces and so does the transistors consumed in routing.

2.9.10.6 Effect of Number of LUT Inputs K on Leakage Power:

For a cluster of size $N = 4$ and $I = 10$, the experiment is conducted with several LUT sizes. Connection block $F_{c_input} = 0.5$, $F_{c_output} = 0.25$ and switch block $F_s = 3$ is used.

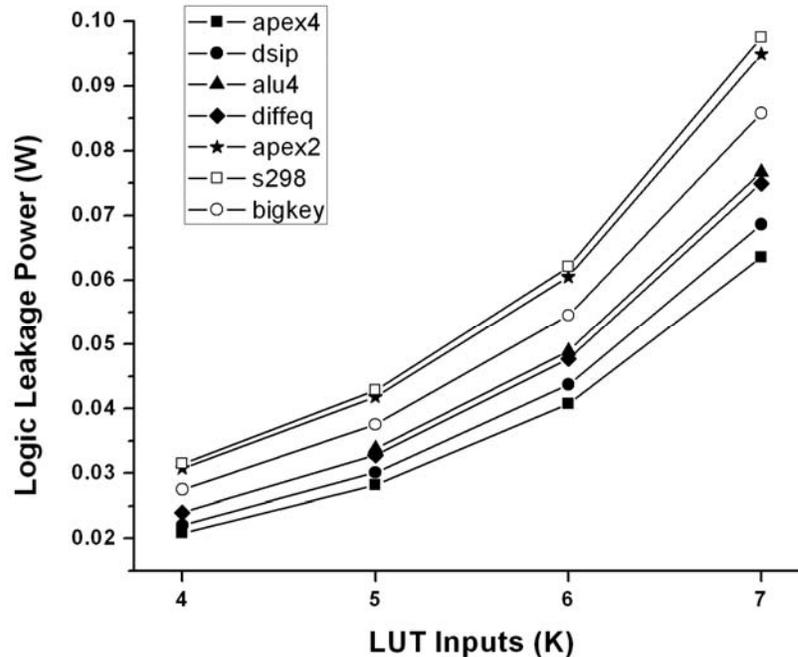


Figure 2.29. Logic block leakage power for designs placed & routed with cluster of $N = 4$, $I = 10$ and of different number of LUT inputs

Increasing the size of LUT increases the logic inside the cluster. But since inputs to the cluster remain constant, basic logic element utilization inside the cluster does not change. This also does not help reduce the number of clusters. But clusters now consume more transistors due to larger LUT. Hence, there is increase in the logic leakage power. Routing leakage power remains unaffected.

2.9.10.7 Effect of Connection Block Flexibility F_c on Leakage Power:

Experiment is conducted for different values of connection block flexibility with cluster of size $N = 4$, $I = 10$ and LUT inputs $K = 4$. Switch block $F_s = 3$ is used.

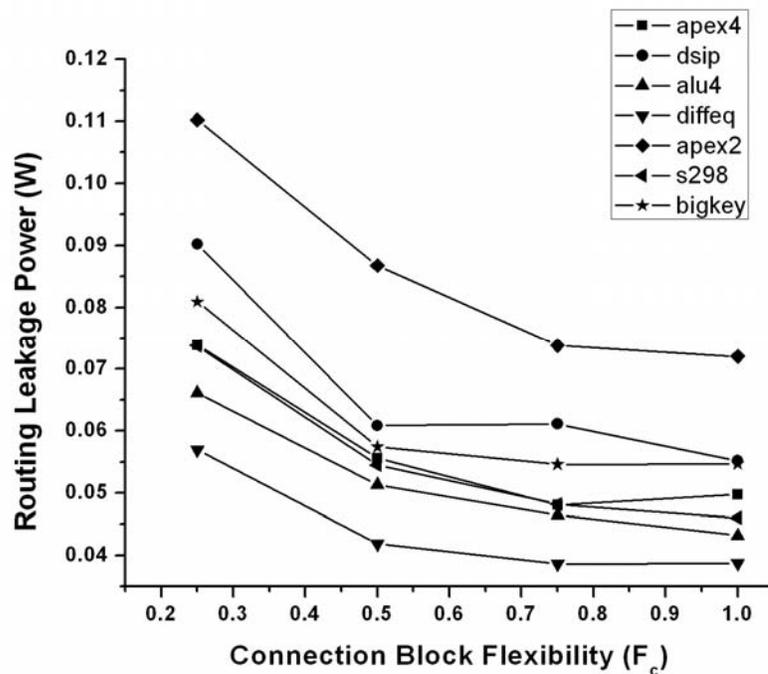


Figure 2.30. Routing leakage power for designs placed & routed on fabric with different connection block flexibilities

Increasing connection block flexibility has no impact on logic block leakage power and helps decrease routing leakage power. Experiment showed that increasing F_c reduces

W_{\min} . For constant switch block flexibility, smaller W_{\min} leads to a smaller switch block. But increasing F_c leads to a larger utilization of transistors in the connection block. From experiment the device array area is shown to be reduced and smaller number of connection blocks is needed. The net result is fewer numbers of transistors being consumed in routing and hence decreasing in routing leakage power.

2.9.11 Conclusions:

In this thesis work, state based leakage estimation technique was added to the VPR CAD tool to enable leakage power estimation in 65nm SRAM based island style FPGAs. It was found that leakage power has increased significantly with scaling and showed that gate and band-to-band leakage power can no longer be ignored. Also it was observed that routing dissipates significantly more leakage power than the actual logic of the design as it uses more number of transistors in the device. Furthermore, increasing cluster size helps in decreasing inter-cluster area but increases intra-cluster area as more and more logic is packed inside the cluster. Hence logic leakage power increases and routing leakage power decreases. Increasing the size of LUT helps increase the logic inside the cluster and but does not affect logic utilization as inputs to cluster is the same. So there is no impact on routing leakage power. But logic leakage power increases. On increasing the number of inputs to the cluster, logic leakage power remains the same and routing leakage power increases. Overall the best settings for achieving minimum leakage power is having size $N = 4$ to $N = 6$, inputs $I = 12$ to $I = 16$, LUT size $K = 4$ or $K = 5$ and larger value of fractional F_c .

CHAPTER 3

COMPOSITE LEAKAGE CURRENT MAXIMIZATION

This section presents a heuristic for maximizing individual leakage components and total leakage currents in a circuit. A heuristic is developed to find the input pattern(s) that can yield maximum leakage in as many gates in the circuit. From previous chapter, it was observed that magnitude of leakage current depends on the logic input pattern to a gate and may vary significantly from one input pattern to the next. So there exists a pattern that can maximize leakage in a single gate. However, application of a particular input to maximize leakage in one gate may cause an implied suboptimal truth assignment at another gate due to Boolean constraints of the circuit. Maximum leakage in a circuit will be less than the sum of the worst case leakage of all the gates. Thus we need to maximize the total leakage without violating the Boolean relationships. Therefore, this is a weighted max-satisfiability problem. The following example illustrates the problem.

Consider an example circuit shown in Figure 3.1. The circuit is entirely composed of 2-input NAND gates. Assume that 2-input NAND gate has the following leakage current values corresponding to various input combinations as shown in Table 3.1.

Input Pattern	Total Leakage (nA)
00	1
01	8
10	7
11	10

Table 3.1. Total leakage corresponding to each input combination in a NAND gate

It can be seen that if a pattern applied sets the inputs of all the NAND gates to '1' then total leakage will be maximized in the circuit. But this is not possible due to the Boolean relationships in the circuit. Suppose a greedy approach is followed in which an input pattern is derived that sets highly controllable gates to maximum leakage state. In the circuit, the gates G1 and G2, G4 and G3 are highly controllable as they have more than one input as a primary input. Thus, all inputs $i_1 \dots i_7$ are set to logic '1' to set gates G1, G2, G4 and G3 in the maximum leakage state. This implies logic '0' at one of the inputs of gates G5, G3 and G6 in the next level, in which leakage of no more than 8 can be achieved due to a logic '0' at one of the inputs. This is a Boolean constraint. After forward implying the input pattern $\langle i_1 i_2 i_3 i_4 i_5 i_6 i_7 \rangle = \langle 111111 \rangle$ for rest of the nodes, the total leakage obtained is 66. Without Boolean constraints, total leakage would have been 80. Total leakage current is the sum of the leakage in all the gates for a given input pattern. The above pattern however does not produce maximum leakage in the circuit. If the pattern $\langle i_1 i_2 i_3 i_4 i_5 i_6 i_7 \rangle = \langle 0100101 \rangle$ is applied, the total leakage will be 70. Note that logic gates G3 and G6 have all transistors sized 2X larger than minimum sized transistors and hence leakage values are also twice as leakage current depends linearly on transistor width. In Figure 3.1, for each gate input two logic values are shown. First logic value corresponds to input pattern applied using the greedy approach and second logic value produces a larger leakage than the greedy approach and for both patterns, leakage current of each gate is shown inside the box.

The above example shows that leakage estimation by adding the maximum leakage in all the gates is a significant over-estimation compared to the actual maximum leakage experienced by a given circuit. Moreover, a greedy approach to obtain pattern for leakage

maximization does not work satisfactorily. Furthermore, in a technology mapped circuit gates of different sizes can have correspondingly varying magnitudes of leakage current making it more inconvenient to use a greedy approach.

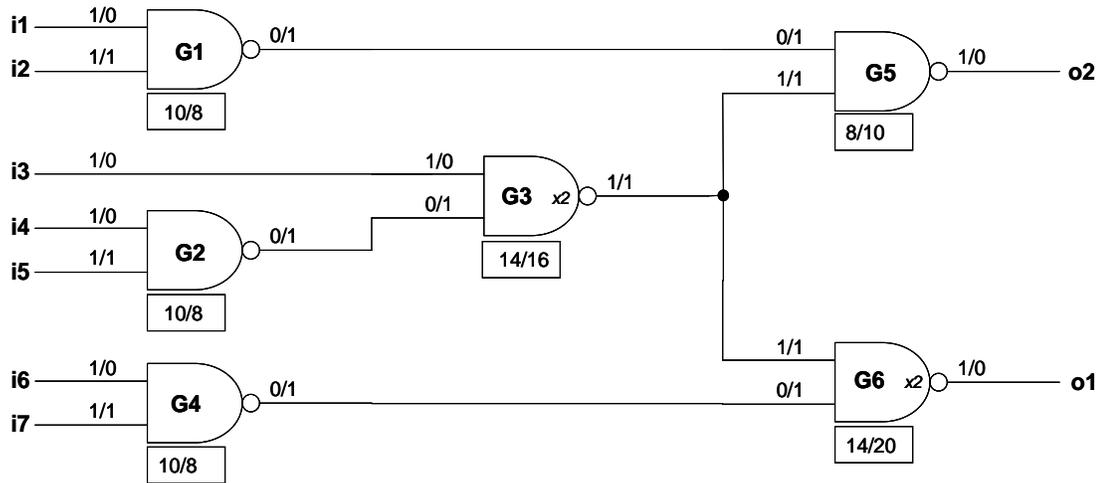


Figure 3.1. Illustration showing weighted max-satisfiability problem

This thesis work formulates a branch-and-bound heuristic for the solution of the NP-hard problem stated above. Instead of finding an exact solution, a tight upper and lower bound for the leakage current is estimated. This approach reduces the search space and makes a practical solution attainable. The solution involves the following steps:

- Step 1. Leakage Characterization of Cell Library:** Leakage currents are pre-computed for all gates in our cell library.
- Step 2. Circuit Modification:** In this step, circuit modification is performed to map our problem for the pattern generation step.

Step 3. Pattern Search for Maximizing Leakage: In this phase, we perform test pattern generation on the modified circuit to determine the pattern that causes maximal leakage current. It consists of the following sub-steps:

- a) **Preprocessing:** Random pattern logic simulation is performed to determine an initial estimate of maximum leakage.
- b) **Branch-and-bound Heuristic:** A branch-and-bound heuristic called Current Maximizing Pattern Generation (CMPG) [31] is applied to improve upon the initial estimate obtained from preprocessing, by performing Boolean search on the modified circuit with the stored library of leakage values to find the pattern that will maximize overall leakage.

3.1 Leakage Characterization of Cell Library

Leakage of each gate in the circuit is characterized as a function of input patterns applied to the gate. Leakage values are tabulated for each gate against its Boolean inputs. This is obtained by simulating each gate against a range of possible Boolean inputs. CMOS gates such as NAND, NOR and XOR are limited by the stack height and typically have few inputs making it possible to perform exhaustive simulation. Below are the steps for computing leakage of a gate.

1. Leakage Current Estimation at transistor level: This is the transistor state based leakage compact model presented in the previous chapter.
2. Leakage Current Estimation at gate level: Exhaustive input pattern simulation is performed on a logic gate to get the transistor bias states. Leakage values are then read from the state look up table created in the previous step by indexing the

appropriate bias state. Total leakage of a gate is obtained by summing the leakage values of all the transistors in the gate. Special Case: Non-conducting transistors in a stack tend to reduce sub-threshold leakage. This is known as transistor stack effect (section 1.2). This effect is accounted by scaling down the sub-threshold leakage current by a factor that depends on the bias state of the drain and source terminals and the number of non-conducting transistors [29][30]. The leakage obtained in nano-amperes is then normalized across all the gates to create an integer value called *weight* that corresponds to the leakage under the specific input to the gate. The branch & bound heuristic will deal with leakage weights in a circuit rather than actual leakage values.

In the actual context specific leakage of a gate in a circuit typically increases slightly due to loading effect. However, when deriving leakage weights, it is only important to know which input states will cause maximum leakage. Figure 3.2 shows the leakage weights for the logic gates. It can be seen that for input pattern <000> in NAND3 gate, in <111> in NOR3 gate and <10> in XOR2 gate yields the highest leakage weights.

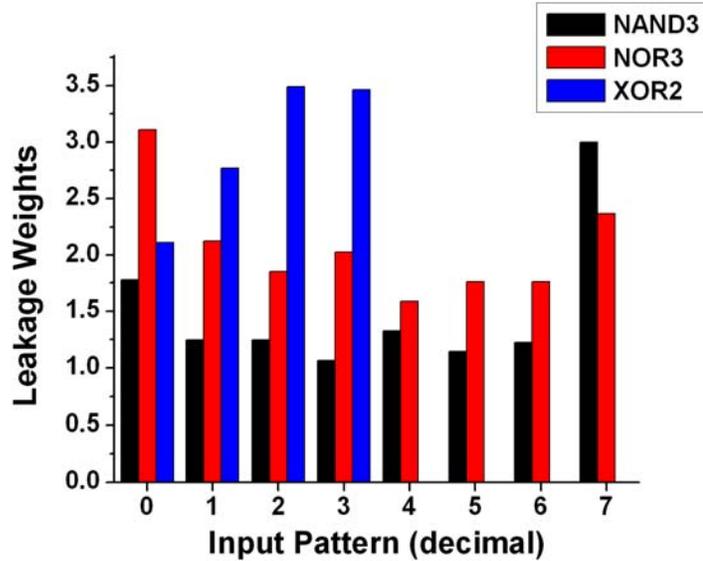


Figure 3.2. Histogram showing leakage weights of three different gates for different input vectors

The leakage characterization step results in a table of weights that correspond to leakage for each gate for every possible Boolean input. If a cell library consists of N gates and the highest number of input for any gate is m , then the space complexity of the library constructed is of the order of $O(N \times 2^m)$.

3.2 Circuit Modification

The circuit netlist supplied to the CMPG algorithm in the pattern generation phase, described in the next section, should have a single weight associated with each logic gate. But, due to pattern dependence of leakage, a gate has leakage weights associated with every input pattern for a logic gate. In order to conform to the single weight assumption, a logic transformation is performed. An example is shown below to illustrate the transformation.

For a 2-input NAND gate having inputs a_0 and a_1 , there are 4 possible input combinations. Each input combination potentially has a unique leakage value described by weights w_0, w_1, w_2, w_3 corresponding to the input combinations $\{00, 01, 10, 11\}$. The NAND gate is replaced in its decoded form with 5 gates as shown in Figure 3.3.

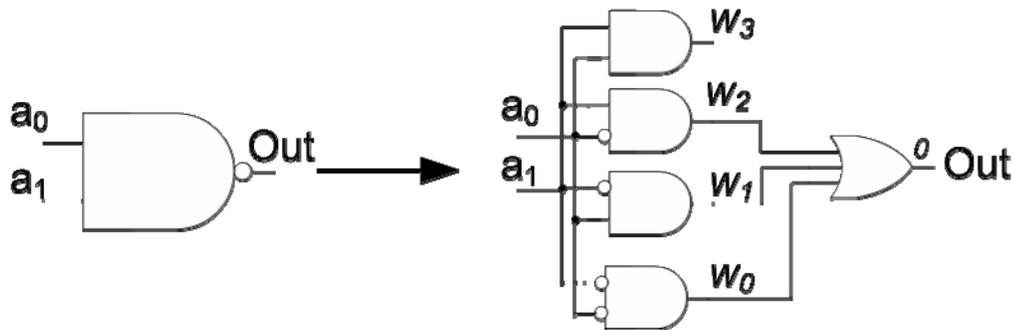


Figure 3.3. Circuit modification

Each product term is realized by a single AND gate with appropriate polarity inversion at the input followed by an OR gate to compute the final output. The weights w_0, w_1, w_2, w_3 are assigned to the AND gates that generate each product term and weight of the OR gate is 0. The output of AND gate with weight w_3 does not input the OR gate because when both inputs to AND gate are logic '1', the output of the AND gate will be logic '1' but the output of the NAND gate must be at logic '0'. In other cases when one or more inputs are at logic '0', the outputs of one of the AND gates having weights w_0, w_1 and w_2 will be at logic '1'. The output of OR gate will also be at logic '1', which would also be the case in a NAND gate. Thus a logically equivalent network is created, where each gate has a single weight.

3.3 Pattern Search for Maximum Leakage

The Current Maximizing Pattern Generation (CMPG) heuristic [31] to solve the Step 2 is used here. It reduces the exponential complexity of the original problem by using the parameter η . The following two measures for η are used.

- 1) **Relative measure:** This is the ratio of leakage weight induced by a pattern and the sum of the maximum leakage weights of all the gates.
- 2) **Absolute measure:** This is the ratio of leakage weight induced by a pattern and the sum of the total leakage weights of all the gates.

For example, in a circuit with n gates, with a gate i having leakage weights W_{1i}, W_{2i}, W_{3i} ... etc. arranged in the descending order (W_{1i} being the maximum), the total of the maximum leakage weights of individual gates is $\sum_{i \in n} W_{1i}$ for the circuit. If W_{pat} is the leakage weight induced by the pattern then $\eta_{relative}$ is given by

$$\eta_{rel} = \frac{W_{pat}}{\sum_{i \in n} W_{1i}} \quad (3.1)$$

The $\eta_{absolute}$ is given by

$$\eta_{abs} = \frac{W_{pat}}{\sum_i \sum_j W_{ji}} \quad (3.2)$$

A pattern which sets all the gates in the circuit in their maximum leakage state will have the maximum $\eta_{absolute}$ value given by

$$\eta_{max} = \frac{\sum_i W_{1i}}{\sum_i \sum_j W_{ji}} \quad (3.3)$$

CMPG algorithm obtains the solution of max-satisfiability problem by using a branch-and-bound heuristic to efficiently explore the search space. It invokes the branch-and-bound algorithm with a target $\eta_{absolute}$ to generate a pattern that satisfies it. The following example explains the significance of $\eta_{absolute}$ and $\eta_{relative}$.

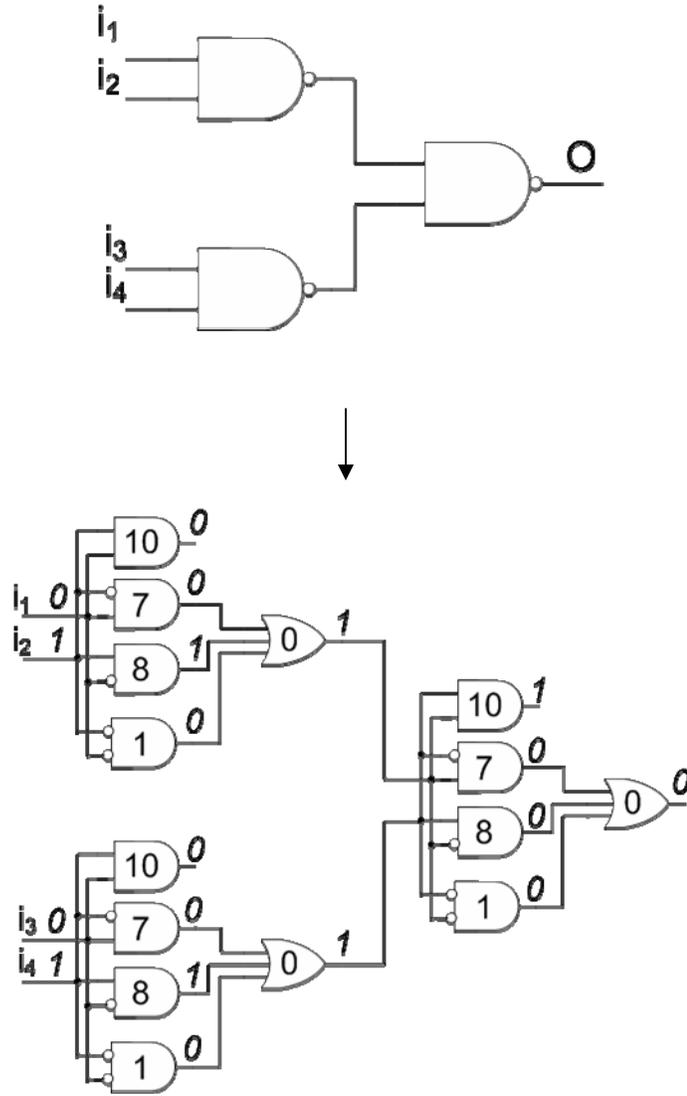


Figure 3.4. Circuit modification for peak leakage estimation

Figure 3.4 shows the circuit modification done for pattern generation. The numbers inside the gates represent the leakage weights. A particular leakage weight is excited if the

output of the gate is set to 1. Thus the maximum possible leakage in the circuit is the sum of the leakage of all the gates which is $W_{max} = 3 * (1+8+7+10) = 78$. For a target $\eta_{absolute}$ value of 0.3, the branch-and-bound algorithm will generate a pattern that sets a total weight of $\eta_{absolute} * W_{max} = 23.4$. A possible solution is the application of the pattern $\langle i_1 i_2 i_3 i_4 \rangle = \langle 0101 \rangle$ which induces a leakage weight of 26. It is not possible to set a leakage weight of W_{max} as it requires a gate to be in all the possible leakage state at the same time. A more realistic goodness metric for the solution is $\eta_{relative}$ which indicates the leakage weight set as a fraction of the sum of worst case leakage weights for all the gates.

3.3 Results

Experiments were performed on the ISCAS 85 benchmark circuits with a goal of maximizing the individual and the total leakage current components. All ISCAS 85 circuits were mapped to a technology library using ABC synthesis tool [32]. The technology library containing of NAND, NOR, INV and XOR gates with various drive strengths and fan-in limited to 3 for NAND and NOR gates was obtained from [33]. Target η to be used by CMPG algorithm is established by random pattern logic simulation in the preprocessing step with an overflow limit of 50 random pattern simulations. For example, by running simulations on c7552 circuit to find the total leakage (Figure 3.5), the maximum leakage current is consumed at the 74th pattern ($\eta = 0.142$), which does not improve for the next 50 patterns. If the target η is attained then the CMPG algorithm increments η by 0.0001 and branch-and-bound search is done by setting the time-outs for both upper and lower bounds of 45 minutes. The time-out values are obtained empirically [31].

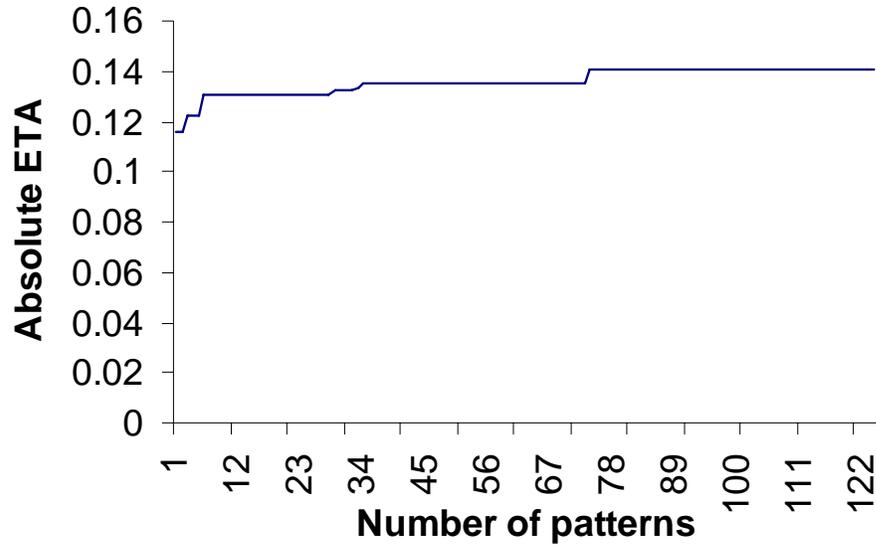


Figure 3.5. Variation of maximum absolute ETA in random pattern logic simulation

Experimental results on ISCAS 85 benchmark circuits are presented for the total leakage (Table 3.5) and its major components, *viz.* Sub-threshold (Table 3.2), Gate (Table 3.3) and BTBT (Table 3.4) leakage.

Circuit Name	Preprocessing			CPMG algorithm				Percentage gain for η_{absolute}	Time in seconds
	Number of Patterns	Max. η_{absolute}	Max. η_{relative}	η_{absolute}		η_{relative}			
				Lower Bound	Upper Bound	Lower Bound	Upper Bound		
c17	68	0.354	0.896	0.354	0.354	0.896	0.896	0	-
c432	87	0.142	0.58	0.176	0.244	0.723	1.000	24.655	9.0
c499	83	0.137	0.507	0.137	0.271	0.507	1.000	0.000	-
c880	89	0.16	0.635	0.217	0.251	0.862	1.000	35.748	60
c1355	114	0.137	0.507	0.137	0.271	0.507	1.000	0.000	-
c1908	52	0.103	0.411	0.111	0.250	0.443	1.000	7.786	8.00
c2670	82	0.136	0.562	0.16	0.241	0.662	1.000	17.794	512
c3540	106	0.146	0.585	0.151	0.250	0.606	1.000	3.590	290
c5315	62	0.147	0.577	0.2	0.255	0.786	1.000	36.222	669
c6288	63	0.291	0.718	0.291	0.405	0.718	1.000	0	-
c7552	125	0.139	0.548	0.166	0.254	0.654	1.000	19.343	74

Table 3.2. Sub-threshold leakage maximization bounds in ISCAS 85 benchmark circuits

Circuit Name	Preprocessing			CPMG algorithm				Percentage gain for η_{absolute}	Time in seconds
	Number of Patterns	Max. η_{absolute}	Max. η_{relative}	η_{absolute}		η_{relative}			
				Lower Bound	Upper Bound	Lower Bound	Upper Bound		
c17	70	0.278	0.757	0.278	0.278	0.757	0.757	0.000	-
c432	94	0.211	0.785	0.214	0.269	0.794	1.000	1.146	0
c499	54	0.216	0.798	0.216	0.271	0.798	1.000	0.000	-
c880	54	0.191	0.733	0.191	0.260	0.733	1.000	0.000	-
c1355	54	0.216	0.798	0.216	0.271	0.798	1.000	0.000	-
c1908	80	0.203	0.777	0.203	0.261	0.777	1.000	0.000	-
c2670	82	0.181	0.720	0.186	0.252	0.740	1.000	2.778	34
c3540	108	0.182	0.716	0.182	0.254	0.716	1.000	0.000	-
c5315	146	0.199	0.743	0.202	0.268	0.754	1.000	1.480	1018
c6288	84	0.254	0.786	0.272	0.324	0.841	1.000	6.997	1
c7552	53	0.214	0.773	0.219	0.277	0.789	1.000	2.070	594

Table 3.3. Gate leakage maximization bounds in ISCAS 85 benchmark circuits

Circuit Name	Preprocessing			CPMG algorithm				Percentage gain for η_{absolute}	Time in seconds
	Number of Patterns	Max. η_{absolute}	Max. η_{relative}	η_{absolute}		η_{relative}			
				Lower Bound	Upper Bound	Lower Bound	Upper Bound		
c17	66	0.321	0.855	0.321	0.321	0.855	0.855	0.000	-
c432	53	0.216	0.657	0.231	0.329	0.703	1	7.002	15
c499	91	0.232	0.727	0.233	0.233	0.73	1	0.413	0
c880	122	0.2	0.634	0.208	0.315	0.662	1	4.416	27
c1355	91	0.233	0.729	0.233	0.319	0.731	1	0.274	0
c1908	67	0.203	0.651	0.205	0.312	0.657	1	0.922	0
c2670	70	0.198	0.654	0.211	0.303	0.698	1	6.728	48
c3540	105	0.184	0.596	0.192	0.308	0.622	1	4.362	6
c5315	56	0.208	0.661	0.211	0.211	0.668	1	1.059	128
c6288	98	0.274	0.796	0.289	0.344	0.837	1	5.151	24
c7552	65	0.224	0.695	0.244	0.322	0.695	1	0	-

Table 3.4. BTBT leakage maximization bounds in ISCAS 85 benchmark circuits

Circuit Name	Preprocessing			CPMG algorithm				Percentage gain for $\eta_{absolute}$	Time in seconds
	Number of Patterns	Max. $\eta_{absolute}$	Max. $\eta_{relative}$	$\eta_{absolute}$		$\eta_{relative}$			
				Lower Bound	Upper Bound	Lower Bound	Upper Bound		
c17	65	0.298	0.928	0.298	0.298	0.928	0.928	0.00	-
c432	93	0.142	0.616	0.167	0.231	0.723	1	17.37	9.00
c499	114	0.138	0.546	0.138	0.254	0.546	1	0	-
c880	89	0.159	0.678	0.206	0.235	0.878	1	29.49	46
c1355	114	0.139	0.546	0.139	0.254	0.546	1	0	-
c1908	52	0.104	0.432	0.107	0.24	0.445	1	3.00	7.00
c2670	82	0.137	0.598	0.157	0.229	0.686	1	14.71	893
c3540	106	0.148	0.622	0.148	0.237	0.622	1	0	-
c5315	62	0.149	0.626	0.189	0.238	0.792	1	26.51	360
c6288	53	0.276	0.776	0.276	0.355	0.776	1	0	-
c7552	125	0.141	0.587	0.162	0.24	0.674	1	14.82	890

Table 3.5. Total leakage maximization bounds in ISCAS 85 benchmark circuits

In each of the tables, the improvement in the lower bounds of $\eta_{relative}$ obtained from the CPMG algorithm and the preprocessing step are compared. The second column represents the total number of patterns generated for preprocessing step for a overflow limit of 50 while the third and fourth columns represent the maximum $\eta_{absolute}$ and $\eta_{relative}$ achieved from preprocessing. The CPMG algorithm starts with the $\eta_{absolute}$ obtained from preprocessing and generates the upper and lower bounds of $\eta_{absolute}$. The bounds for $\eta_{absolute}$ and $\eta_{relative}$ are shown in the columns 6-8. Column 9 calculates the percentage improvement in $\eta_{relative}$ with respect to preprocessing. Column 10 represents the time it took for CPMG to reach the improved lower bound. This time does not include time-outs and the time spent in searching for the upper bound. It can be seen from the tables that CPMG is able to provide good amount of improvement in the lower bound of $\eta_{relative}$ as compared to preprocessing. Moreover, it reaches the solution very fast.

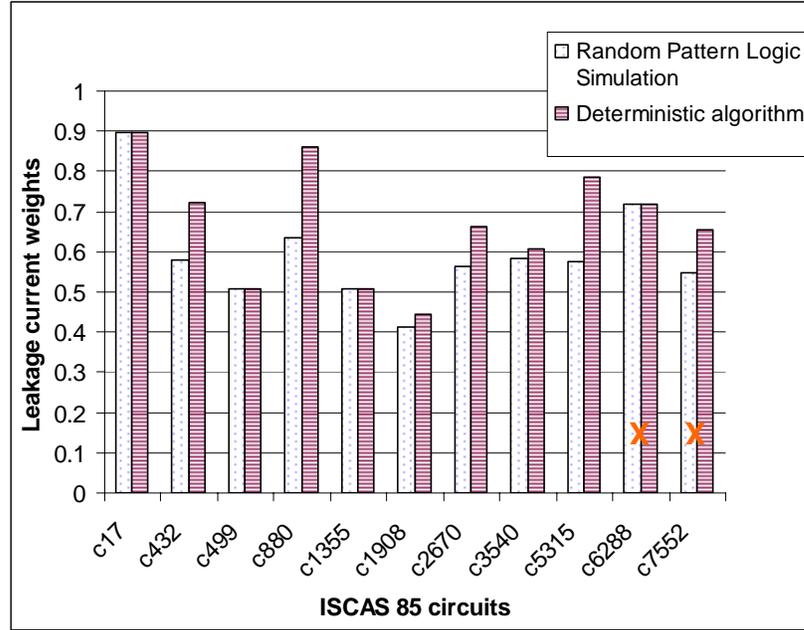


Figure 3.6. Improvement in $\eta_{relative}$ from CMPG algorithm over preprocessing for sub-threshold leakage

In Figure 3.6, in circuits marked with an X we obtain meaningful result where the ILP based solution from Chai and Kuhelman failed to find an answer within a reasonable period of computation [34]. It should be noted that if the standard cell library contains both high and low- V_T gates, then our approach targets gates with low- V_T instead of high- V_T logic gates as low- V_T gates produce more leakage. The proposed approach is applicable in combinational circuits only. For sequential circuits operating in functional mode, input pattern found to produce lower bound of maximum leakage might lead to an unreachable state. In that case, the pattern is not applicable. Further, for testing very large combinational circuits, scan chain is needed for applying the patterns and it is unknown which patterns might lead to unreachable states. Like any other ATPG technique, the approach suffers from the same problem.

3.4 Conclusions

In this thesis work, the problem of pattern generation for estimating maximum (minimum) composite leakage was studied. Our study shows that the CMPG algorithm almost always gives a higher estimation of composite leakage than random simulation alone. The technique gives a firm lower bound on composite leakage that is significantly higher than values obtained from random simulation alone. This shows the value of progressively tightening the bounds for an asymptotically exact solution over other techniques that attempts to find an exact solution directly. Since, leakage is a significant component of the total power consumption in technologies 65nm and below, accurate estimation of leakage allows better estimation of headroom for dynamic power during wafer test where tests suffer from power limitation due to limitation of probing methods.

CHAPTER 4

IMPACT OF DYNAMIC POWER ON LEAKAGE CURRENT

This section of the thesis analyzes the effect of considering both leakage and dynamic power together at given time t in a given circuit. This study investigates how much of the overall dynamic current is contributed by leakage. Several papers have been reported in the area of power estimation [25]-[27]. In non circuit-level simulation cases, the total power of a gate/logic network is computed by the simple equation:

$$P_{total} = P_{dynamic} + P_{static} = \alpha \frac{1}{2} C_L V_{DD}^2 f + I_{leak} V_{DD} \quad (4.1)$$

In this equation, both dynamic and static power are summed but are not computed concurrently. This equation models dynamic power as a probabilistic entity by the parameter α , which is the switching activity factor and is pattern independent. On the other hand leakage current I_{LEAK} constitutes only of sub-threshold leakage, which is pattern dependent and only occurs in ‘OFF’ transistors. Furthermore, in a 45nm technology node, gate and BTBT leakages have become dominant and it was demonstrated in previous chapters that all these three leakage components vary greatly with the input patterns applied. To account for this, leakage and switching currents need to be computed together through gate-level circuit simulation, which will not only help consider switching of nodes with respect to time for accurately estimating switching current but also accounts for pattern dependent leakage current. The additional benefit of the method is the capability to perform circuit simulation on fairly large circuits, for which HSPICE is not feasible. The solution involves a two-step process. In the first step, leakage

current compact models are created based on Berkeley Predictive Technology Model for 45 nm. Second step is to estimate switching current. Since leakage current compact models were developed and presented in section 3.1, switching current models are only presented here.

4.1 Switching Current Estimation

Switching current depends on rise and fall times of the input transition and the load capacitance present at the output. Compact model to estimate switching current was developed by performing a set of HSPICE simulation for the entire cell library under different fan-outs and input signal transition slopes. A regression analysis is done on the data to obtain a set of equations for peak switching current and output transition slope as a function of input transition slope and number of fan-out. To reduce the complexity of the model, peak switching current values are considered. These pre-characterized equations were applied for computing switching currents for gates in-situ. An example of such an equation is shown below for a 3-input NAND gate that drives five times its input capacitance. In these equations, the input slope (m) is specified in nanoseconds.

$$I = \left\{ \begin{array}{l} 10^{-5} \times \left(0.711 + 1.74 \times \exp\left(-\frac{m}{3.42}\right) + 4.65 \times \exp\left(-\frac{m}{0.55}\right) \right) \quad m \leq 10 \\ 10^{-6} \times \left(4.85 + 2.35 \times \exp\left(-\frac{m}{32.58}\right) + 7.33 \times \exp\left(-\frac{m}{6.25}\right) \right) \quad m > 10 \end{array} \right. \quad (4.2)$$

There are different equations for different loads and different cells in our compact model. Another set of equations provide output signal propagation delay and output signal transition slope for a given input slope. Examples of output rise t_r and fall t_f times and

input to output propagation delay t_p as a function of input transition slope m for the said 3-input NAND gate is shown below.

$$\begin{aligned}
 t_p &= \left\{ 10^{-10} \times \left(2.51 + 1.91m - 0.135m^2 + 5.33 \times 10^{-3} m^3 - 1.19 \times 10^{-4} m^4 \right. \right. \\
 &\quad \left. \left. + 1.48 \times 10^{-6} m^5 - 9.58 \times 10^{-9} m^6 + 2.5 \times 10^{-11} m^7 \right) \right\} \\
 t_r &= \left\{ 10^{-10} \times \left(5.4 + 1.92m - 0.011m^2 + 5.09 \times 10^{-5} m^3 \right) \right\} \\
 t_f &= \left\{ 10^{-10} \times \left(11.96 + 1.66m - 8.25 \times 10^{-3} m^2 + 3.59 \times 10^{-5} m^3 \right) \right\}
 \end{aligned} \tag{4.3}$$

4.2 Total Current Estimation

Total current comprise of switching current and leakage current. Since these currents are dependent on logic states of the circuit under consideration, random pattern simulation is used to estimate total current. The simulation consists of two steps. First logic simulation is performed and then leakage and switching currents are overlaid on a time axis to compute the overall current waveform.

Logic simulation is performed using an event driven simulator. During logic simulation actual switching delays are not used. At the end of logic simulation, we determine which gates have switched and which gates have not. This method of computation however ignores power due to signal glitches. To compute glitch power, exact glitch waveforms must be known and the compact switching current model needs to be enhanced for partial switching. This leads to unprecedented complexity. That is why most dynamic power analysis tools ignore switching current but compensate for that by using some heuristic multiplicative factor.

For placement of switching current on time axis, we use gate delay propagation equations. Once the delay values are obtained, they are placed on a unit delay scale after normalization and it is assumed that the peak current is consumed by the gate during that unit time duration. It has been shown previously, that such unit delay often approximates exact delay effects reasonably well [35]. The switching current depends on input signal slope. In order to calculate input signal slopes for all the gates, we start out at the primary inputs with a sharp input transition and by using pre-computed equations we estimate the output slope. The output slope transition is then propagated forward. During this step, the leakage and switching currents are computed for each gate using compact models described in the previous section. The gates that are switching are assumed to have no leakage current while the non-switching gates are presumed to have leakage currents that are computed by knowing their logic input values after logic simulation as described earlier. Also, while estimating leakage current, loading effect [3] is not used, as it has negligible effect on total power in the presence of switching current.

We divide the logic simulation into a number of time steps to show the total current drawn at different times (Figure 4.1). The number of time steps is equal to the critical path length of the circuit assuming that each gate has one time-step delay (Unit Gate Delay). For logic simulation, a pattern pair is generated such that it excites the critical path in the circuit. This pattern pair is obtained by performing random pattern simulation and finding the number of gates switched in all the paths. If the number of gates switched does not change after several random patterns, then the last pattern found is used.

Comparison of total current with and without leakage for c7552

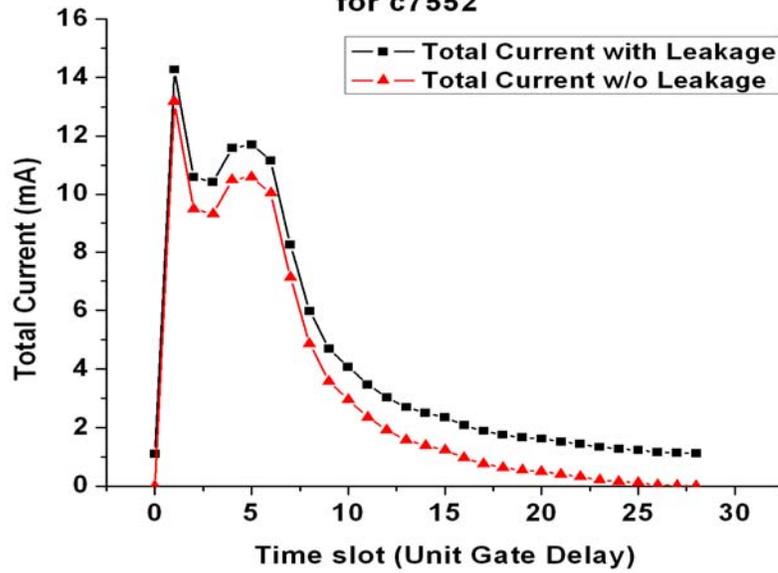


Figure 4.1. Plot of total current with and without leakage for c7552 with time

Variation of Leakage with time for c7552

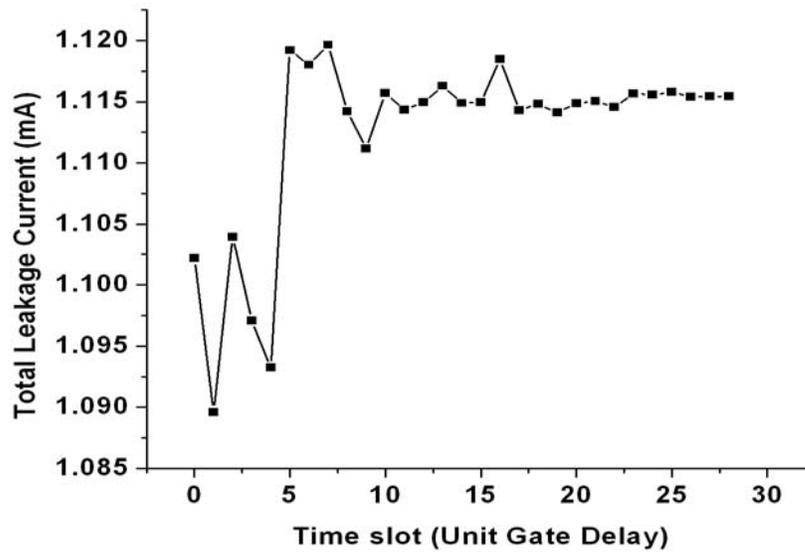


Figure 4.2. Plot of total leakage current for c7552 with time

4.3 Results

Experiments were conducted on 10 ISCAS benchmark circuits were converted into cell library consisting of up to 3-input primitive gates of the following types: INV, NAND, NOR, XOR by using SIS.

Below it is demonstrated that by having switching and leakage power computation concurrently, leakage power can deviate by more than 8% with time. As signal transition propagates through level of gates at time t_1 , level k gates in the circuit are switching and consuming dynamic power whereas level 0 to level $k-1$ and level $k+1$ to level N gates are idle and consuming leakage power, which greatly depends on the input patterns currently at those gate inputs. At time t_2 , switching propagates to next level of gates. Level $k+1$ gates are switching and level 0 to level k and level $k+2$ to level N gates are consuming leakage power only. In the duration of time from t_1 to t_2 , the input patterns at level $k+1$ gate inputs have changed as signal transition propagated. Some inputs have changed from $0 \rightarrow 1$ and others $1 \rightarrow 0$ and hence leakage power also changed. This accounts for variation in leakage current and hence in total power. For different circuits, leakage power can increase or decrease over time. This effect is not accounted when switching current is computed separately from leakage power. In that case, all the gates are assumed to be idle and leakage power consumed remains constant.

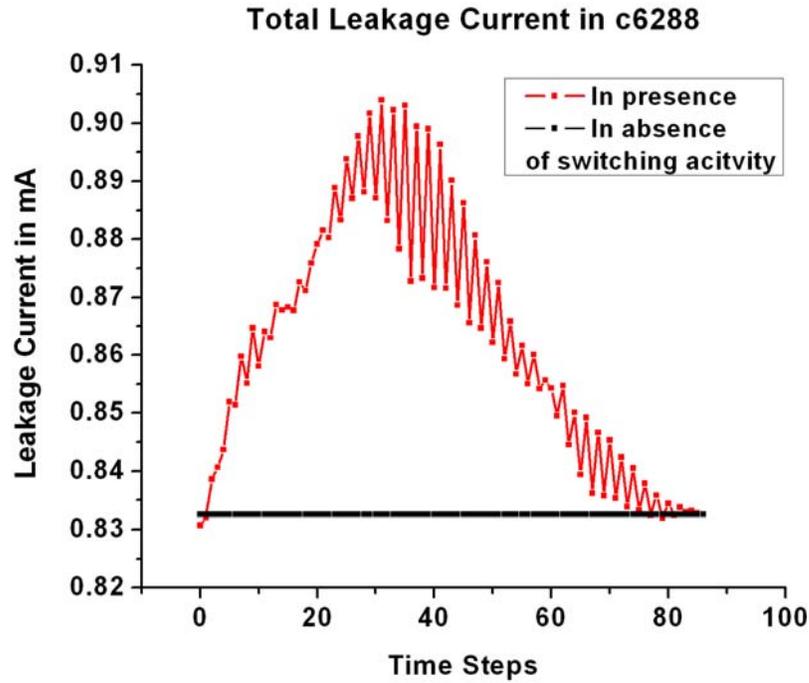


Figure 4.3. Leakage current in presence and absence of switching activity

Below shown are the switching current and leakage values are averaged over all the time steps. For zero delay model simulation, same procedure is followed, except that there is only one time slot at which the switching current of all the gates in the circuit, are added.

Circuit	Average Total Current (mA)	Average Leakage Current (mA)	Leakage in % of the Total Current
c17	0.104	0.00246	2.36
c423	0.422	0.090	21.43
c499	0.517	0.144	27.84
c880	0.690	0.145	21.08
c1355	0.895	0.208	23.23
c1908	0.694	0.220	31.73
c2670	1.336	0.368	27.56
c3540	1.438	0.483	33.62
c5315	2.262	0.722	31.93
c6288	9.542	0.853	8.94
c7552	4.379	1.112	25.40

Table 4.1. Comparison of average values of total and leakage current in mA for ISCAS benchmarks circuits

Circuit	Peak Switching Current (mA)		
	Delay Model	Zero Delay Model	% Difference
c17	0.284	0.407	30.21
c423	2.423	8.294	70.79
c499	3.917	7.839	50.04
c880	5.340	11.433	53.29
c1355	8.476	19.235	55.93
c1908	3.845	11.847	67.55
c2670	9.324	25.170	62.96
c3540	7.050	36.283	80.57
c5315	16.089	52.352	69.27
c6288	25.049	1060.015	97.64
c7552	13.172	94.730	86.10

Table 4.2. Comparison of peak switching current in mA for different delay models for ISCAS benchmark circuits

Our total current estimation technique has been validated using HSPICE. Each cell in the library is simulated in HSPICE to get leakage. Leakage current model derived in previous chapters is replaced by leakage current values from HSPICE. The same switching current model is used as it was derived using HSPICE. Table 4.3 compares the average total current obtained from our technique for each of the above technology mapped benchmark circuits to the average total current obtained by simulating each of those circuits in HSPICE.

Circuit	Average Total Current (mA)			% Difference
	Our method		HSPICE	
	No calibration	Calibration		
c17	0.102	0.00186	0.002	7.00
c423	0.334	0.074	0.077	3.89
c499	0.378	0.133	0.141	5.19
c880	0.549	0.152	0.160	5.00
c1355	0.693	0.157	0.164	4.27
c1908	0.480	0.1424	0.142	0.28
c2670	0.979	0.291	0.302	3.64
c3540	0.969	0.523	0.537	2.61
c5315	1.560	0.596	0.612	2.61
c6288	8.714	1.522	1.537	0.98
c7552	3.298	1.01	1.044	3.26

Table 4.3. Comparison of average total current obtained from HSPICE and our technique with and without calibration

From Table 4.3, it can be seen that average total current from our method are higher than HSPICE results. This is because our method employs unit delay model in which all the gates in a particular level are consuming peak switching current at the same time slot. This is not true in the case of HSPICE which uses variable delay model and switching current equals to $C_L di/dt$ during switching period t . It should be noted that total charge must be conserved in a circuit no matter which models are used for estimation. This allows average peak current values obtained by our method for each time slot to be calibrated by scaling it down by the ratio of area under curve of switching current values versus time of our method and HSPICE. Thus, for charge conservation, area under curve of peak current values versus time from our method and area under curve of switching current versus time in HSPICE must be equal.

$$\int_{t_0}^{t_1} I^{hspice} dt = \int_{t_0}^{t_1} I_{peak}^{our_method} dt = Q$$

$$Calibration\ Factor = \frac{Total\ Charge\ from\ our\ technique}{Total\ Charge\ from\ HSPICE} \quad (4.4)$$

It should be noted that calibration was done in order to show that average total current obtained by our method is validated. Table 4.3 shows on average of less than 5% error difference from HSPICE results. Hence our method can provide reasonable results on fairly large circuits such as c7552 in a small time, on which HSPICE is not feasible. Without calibration the actual peak currents are found to be similar to HSPICE results (Figure 4.4).

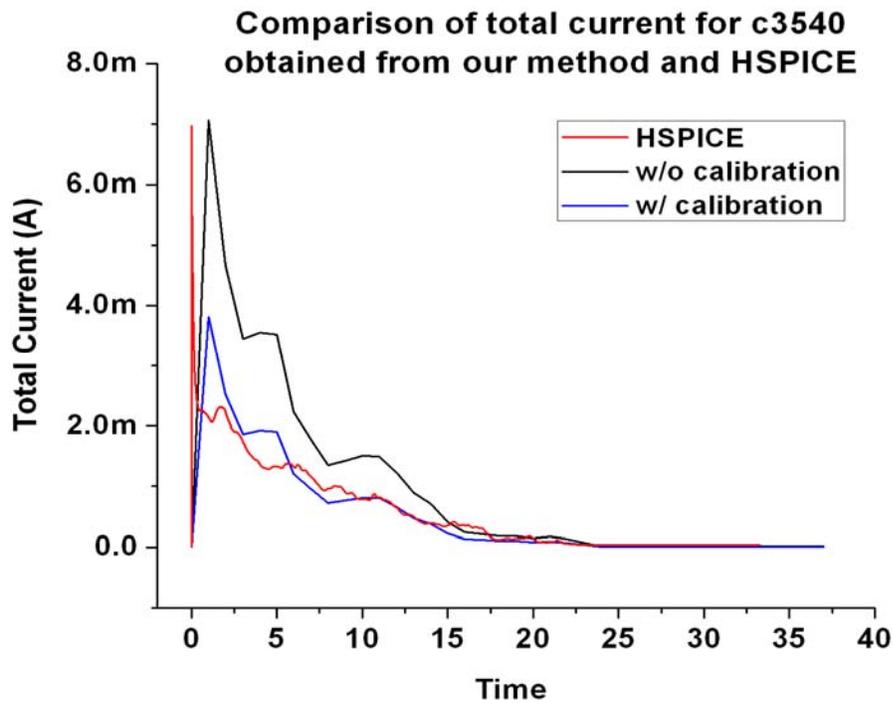


Figure 4.4. Total current obtained from our technique and HSPICE for c3540

4.4 Conclusions

In this thesis work, the contribution of leakage current during switching was investigated. In order to conduct the study a compact model to estimate leakage and switching currents in circuits was developed. On an average it was found that leakage current is about ~24% of the total switching current. For one circuit, the difference was found to be as high as ~34%. Typically, switching power is computed based on $\frac{1}{2}\alpha CV^2f$ formula that ignores switching delays. The switching delays reduce the peak current by spreading it out in time. Zero-delay model over-estimates peak current by ~66% in benchmark circuits. By concurrently estimating switching power, leakage power was shown to vary significantly. In summary, from this study we conclude that all future dynamic current/power computation needs to account for leakage in non-switching nodes.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

The thesis provides in depth focus on leakage power analysis and estimation for the next generation of VLSI design targeted in 65 and 45nm technology. The thesis work reflects upon major sources of leakage such as sub-threshold, gate oxide tunneling and band-to-band tunneling leakage that will create high power dissipation in VLSI designs. This work analyzes leakage effects such as loading effect and stack effect and to deliver a simple, faster and accurate technique for leakage estimation and maximization to help the designer in creating better designs before taping to silicon.

CHAPTER 1 summarizes the leakage current concepts and effects and CHAPTER 2 shows the methodology for an efficient and accurate leakage estimation technique that considers all major sources of leakage and incorporate loading effect for estimating leakage power. An different version of the technique is also shown that computes gate leakage iteratively using Newton Raphson Method to yield a better estimate at the cost of performance. The results were shown for ISCAS benchmark circuits and the technique can be used on fairly large circuits where HSPICE is infeasible. A leakage estimation technique for FPGAs is also developed, which was incorporated into the Versatile Place and Route tool to help designers to get leakage power of their prototype design after placed and route.

CHAPTER 3 discusses the heuristics for maximizing leakage current in circuits. The heuristic generates the input pattern(s) that can yield maximum leakage attainable

considering Boolean constraints. The heuristics was shown to give the upper and lower bounds on maximum leakage in ISCAS benchmark circuits in reasonable time.

Finally, CHAPTER 4 shows the impact of leakage power in presence of dynamic. It was shown that leakage power can increase or decrease over time when circuit is consuming dynamic power. This chapter also discusses the methodology for efficient estimation of total power with time.

5.2 Future Work

The leakage estimation work presented in this thesis can easily be extended into future technology nodes of 32nm and 22nm. The only difference remains is to create a new compact model for leakage and switching current for whichever process technology is used. Future processes will incorporate strained silicon to improve the device ON current on the face of scaling supply voltages. This will cause a huge increase in sub-threshold and band-to-band tunneling leakages. It will be interesting to know the effects of strained silicon on circuit leakage power and its effect on loading voltages. For the BPTM process model, loading voltages were not significantly high to cause a design violation but with strained silicon this could be possible in future process. If this is possible then finding the pattern(s) or conditions that can maximize loading effect remains a challenging task. Further to increase the accuracy of the technique several other factors such as process variation, temperature, multi- V_T can also be considered by expanding the cell library and the compact models. Also in the future, the industry might complete port to using Finfets as a device from CMOS. Understanding the leakage mechanisms and effects such as loading opens a new dimension for analysis.

BIBLIOGRAPHY

- [1] ITRS Roadmap: http://www.itrs.net/Common/2004Update/2004_03_PIDS.pdf
- [2] S. Mukhopadhyay, A. Raychowdhury, K. Roy, "Accurate Estimation of Total Leakage Current in Scaled CMOS Logic Circuits Based on Compact Current Modeling," *IEEE/ACM Design Automation Conference (DAC) 2003*, pp. 169-174
- [3] S. Mukhopadhyay, S. Bhunia, K. Roy, "Modeling and Analysis of Loading Effect in Leakage of Nano-Scaled Bulk-CMOS Logic Circuits," *Proceedings on Design And Test Europe Conference (DATE) 2005*, pp. 224-229 Vol. 1
- [4] R. Brown, J. Burns, A. Devgan, R. Brown, "Efficient Techniques for Gate Leakage Estimation," *Proceedings on International Symposium on Low Power Electronics and Design (ISLPED) '03*, pp. 100-103
- [5] H. Rahman, C. Chakrabarti, "A Leakage Estimation and Reduction Technique for Scaled CMOS Logic Circuits Considering Gate Leakage", *Proceedings on International Symposium on Circuits and Systems 2004*, pp. 297-300 Vol. 2
- [6] C. Hu et al., "BSIM4 Gate Leakage Model Including Source-Drain Partition," *International Electron Device Meeting 2000*, pp. 815-818
- [7] C. Hu et al., "BSIM4.5.0 Mosfet Model", User's Manual, 2004
- [8] K.N. Yang, H.T. Huang, M.J. Chen, Y.M. Lin, M.C. Yu, S.M. Jang, D.C.H. Yu, M.S. Liang, "Characterization and modeling of edge direct tunneling (EDT) leakage in ultrathin gate oxide MOSFETs", *IEEE Transactions on Electron Devices*, Vol. 48, Issue 6, June 2001, pp. 1159-1164
- [9] M. Drazdziulis, P. Larsson-Edefors, "A Gate Leakage Reduction Strategy for Future CMOS Circuits", *European Solid-State Circuits Conf. '03*, pp. 317-320
- [10] D. Lee, W. Kwong, D. Blaauw, D. Sylvester, "Simultaneous Sub-threshold and Gate-Oxide Tunneling Leakage Current Analysis in Nanometer CMOS Design", *International Symposium on Quality Electronic Design'03*, pp. 287-292
- [11] D. Lee, D. Blaauw, D. Sylvester, "Gate Oxide Leakage Current Analysis and Reduction for VLSI Circuits", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pp. 155-166 Vol. 12
- [12] Y. Cao, M. Orhansky, T. Sato, D. Sylvester, C. Hu et al., "Spice up your MOSFET Modeling", *Circuits and Device Magazine, IEEE 2003*, pp. 17-23 Vol. 19, Issue 4
- [13] <http://www.altera.com/products/devices/stratix2/features/density/st2-density.html>
- [14] <http://www.altera.com/products/devices/stratix2/features/st2-90nmpower.html>

- [15] L. Fei, Y. Lin, H. Lei, C. Deming, J. Cong, "Power modeling and characteristics of field programmable gate arrays", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp 1712- 1724, Vol. 24, Issue 11, Nov. 2005
- [16] K.K.W. Poon, S.J.E. Wilton, A. Yan, "A Detailed Power Model for Field-Programmable Gate Arrays", *In ACM Transactions on Design Automation of Electronic Systems (TODAES)*, Vol. 10, Issue 2, April 2005, pp. 279-302
- [17] K.K.W. Poon, A. Yan, S.J.E. Wilton, "A Flexible Power Model for FPGAs", *International Conference on Field-Programmable Logic and Applications, Sept '02*
- [18] K.K.W. Poon, Power Model User's Manual: http://www.ece.ubc.ca/~stevew/powermodel/power_manual_v110.pdf
- [19] K.K.W. Poon, "Power Estimation for Field-Programmable Gate Arrays", M.A.Sc Thesis, University of British Columbia, 2002
- [20] A. Ferré and J. Figueras, "On estimating bounds of the quiescent current for IDDQ testing," *In Proceedings 14th VLSI Test Symposium*, pp. 106-111, 1996
- [21] M. C. Johnson, D. Somasekhar, and K. Roy, "Models and algorithms for bounds on leakage in CMOS circuits," *IEEE Transactions on Computer-Aided Design*, Vol. 18, pp. 714-725, June 1999
- [22] M. R. Garey, and D. S. Johnson. "Computers and Intractability: A Guide to the Theory of NP-Completeness". New York: W. H. Freeman, 1979
- [23] S. Bobba, and I. N. Hajj. "Maximum leakage power estimation for CMOS circuits". *IEEE Alessandro Volta Memorial Workshop on Low-Power Design*, pp. 116-124, 1999
- [24] A.Ferre, and J. Figueras. "Leakage power bounds in CMOS digital technologies". *IEEE Transactions on Computer-Aided Design*, Vol. 21, pp. 731-738, June 2002
- [25] W. Hung, Y. Xie, N. Vijakrishnan, M. Kandemir, M.J. Irwin, Y. Tsai, "Total Power Optimization through Simultaneously Multiple-VDD Multiple-VTH Assignment and Device Sizing with Stack Forcing", *International Symposium on Low Power Electronics and Design (ISLPED) '04*, pp. 144-149
- [26] T. Karnik, Ye Yibin, J. Tschanz, Wei Liqiong, S.Burns, V. Govindarajulu, V. De, S. Borkar, "Total Power Optimization By Simultaneous Dual-Vt Allocation and device sizing in high performance microprocessors", *Proceedings on IEEE/ACM Design Automation Conference (DAC) 2002*, pp. 486-491
- [27] A.J. Bhavnagarwala, B.L. Austin, K.A. Bowman, J.D. Meindl, "A minimum total power methodology for projecting limits on CMOS GSI", *IEEE Transactions on Very Large Scale Integration Systems*, Vol. 8, Issue 3, Jun 2000, pp. 235-251

- [28] NGSPICE: Mix-level/mixed-signal circuit simulator based on Berkeley spice3f5, Cider1b1 and Xspice, part of GPL'd suite of Electronic Design Automation Tools (gEDA project)
- [29] R. Gu, M. Elmasry, "Power Dissipation Analysis and Optimization of Deep Submicron CMOS Digital Circuits", *IEEE Journal of Solid-State Circuits*, pp. 707-714, Vol. 31 No. 5, May 1996
- [30] S. Yang, W. Wolf, N. Vijaykrishnan, Y. Xie, W. Wang, "Accurate Stacking Effect Macro-modeling of Leakage Power in Sub-100nm Circuits", *VLSI Design 2005, International Conference on*, pp. 165 – 170
- [31] K. Ganeshpure, "Automatic Pattern Generation for Computing Maximum Switching & Leakage Currents and Detecting Crosstalk Faults", MS Thesis, University of Massachusetts Amherst, 2007
- [32] ABC: A System for Sequential Synthesis and Verification, developed by Berkeley Logic Synthesis and Verification Group
- [33] ASIC Standard Cell Library Design by Graham Petley, <http://www.vlsitechnology.org>
- [34] D. Chai, and A. Kuehlmann, "Circuit-based Preprocessing of ILP and Its applications in Leakage Minimization and Power Estimation," in Proc. *IEEE Intl. Conf. Computer Design (ICCD)*, 2004, pp. 387-392.
- [35] Michael S. Hsiao, Elizabeth M. Rudnick, and Janak H. Patel, "Peak Power Estimation of VLSI Circuits: New Peak Power Measures," *IEEE Transactions On Very Large Scale Integration Systems*, vol. 8, no. 4, August 2000
- [36] V. Betz, J. Rose, "VPR: A New Packing, Placement and Routing Tool for FPGA Research", *Intl. Workshop on Field Programmable Logic and Applications*, 1997
- [37] V. Betz, "VPR and T-VPack User's Manual Version 4.30
- [38] V. Betz, J. Rose, "Cluster-based Logic Block for FPGAs: Area-Efficiency vs. Input Sharing and Size", In *Proceedings, Custom Integrated Circuits Conference*, 1997
- [39] E. Ahmed, J. Rose, "The Effect of LUT and Cluster Size on Deep-Submicron FPGA Performance and Density", *IEEE Transactions on VLSI*, 2004
- [40] T. Krishnamohan, Z. Krivokapic, K. Uchida, Y. Nishi, K. C. Saraswat, "High-Mobility Ultrathin Strained Ge MOSFETs on Bulk and SOI With Low Band-to-Band Tunneling Leakage: Experiments", *IEEE Transactions on Electron Devices*, Vol. 53, No. 5, May 2006

- [41] R. Rao, A. Srivastava, D. Blaauw, D. Sylvester, “Statistical Analysis of Sub-threshold Leakage Current for VLSI Circuits”, *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, pp. 131-139
- [42] R. Bryant, “COSMOS: A Compiled Simulator for MOS Circuits”, *DAC 1987, Proceedings*, pp. 9-16
- [43] SIS: Logic Synthesis of Synchronous and Asynchronous Sequential Circuit program, UC Berkeley

PUBLICATIONS FROM THIS THESIS RESEARCH

1. A. Rastogi, Wei Chen, A. Sanyal, S. Kundu, “An Efficient Technique for Leakage Current Estimation in Sub 65nm Scaled CMOS Circuits Based on Loading Effect”, *20th International Conference on VLSI Design 2007*, pp. 583 – 588
2. A. Rastogi, Wei Chen, S. Kundu, “On Estimating Impact of Loading Effect on Leakage Current in Sub-65nm Scaled CMOS Circuits Based on Newton-Raphson Method”, *44th ACM/IEEE Design Automation Conference (DAC) 2007*, pp. 712 – 715
3. A. Rastogi, K. Ganeshpure, S. Kundu, “A Study on Impact of Leakage Current on Dynamic Power”, *IEEE International Symposium on Circuits and Systems (ISCAS) 2007*, pp. 1069 – 1072
4. A. Rastogi, K. Ganeshpure, A. Sanyal, S. Kundu, “Pattern Generation for Composite Leakage Current Maximization”, To appear in *12th IEEE European Test Symposium ETS 2007*