CMOS LEAKAGE POWER REDUCTION AND DATA RETENTION

G.W.G.K.N. Udayanga

148469U

Degree of Master of Science in Electronics and Automation

Department of Electronics and Telecommunication Engineering

University of Moratuwa

Sri Lanka

07 January 2019

CMOS LEAKAGE POWER REDUCTION AND DATA RETENTION

Gangoda Withana Gamamge Kashita Niranjan Udayanga

148469U

Thesis submitted in partial fulfillment of the requirements for the degree Master of Science in Electronics and Automation

Department of Electronic and Telecommunication Engineering

University of Moratuwa
Sri Lanka

07 January 2019

Table of Contents

Declaration of Candidate and the Supervisor

I declare that this is my own work and this thesis doesn’t incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other university or institute of higher learning and to the best of my knowledge and belief it doesn’t contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: Date:

The Above Candidate has carried out research for the masters thesis under my supervision.

Name of the supervisor: Dr. S. Thayaparan

Signature of the supervisor: Date:

Acknowledgement

I would like to acknowledge the fine support and guidance given by all the lecturers of Electronics and Automation Master's Degree program conducted by Department of Electronics and Telecommunication Engineering University of Moratuwa. And Special thank goes to my dearest supervisor Dr. S. Thayaparan for giving me an interesting and valuable research topic to pursue my master thesis. At the same time my supervisor's continuous guidance helped me a lot to resolve difficulties and problems faced while doing this research. And the support of my MSc 2014/2015 batch mates was a huge encouragement during the Masters degree's time.

Abstract

As silicon technology scaling, leakage power dissipation has become the most significant component from all CMOS power dissipation mechanisms. Minimum Leakage Vector(MLV) is used as a combinational logic leakage power reduction technique when a system is in standby mode. Compared to MLV, though an excellent leakage power reduction can be achieved with power gating technique it has some drawbacks like higher retention time and system state loss.

In this thesis we combine MLV and power gating techniques to achieve more leakage power reduction compared to MLV while mitigating prior mentioned drawbacks of power gating.

Instead of full chip power gating, we developed a simple algorithm which runs in linear time to identify the prospective locations for power gating once combination logic is fed with its MLV. The algorithm was implemented in tcl language and run on top of design compiler shell for a synthesized netlist.

Flip flops and input ports were modified to feed MLV in standby mode while facilitating for partial power gating within the flops without losing flop state to retain the system state back in active mode.

Flop modifications were extended to feed MLV in scan mode also so that scan mode leakage reduction can also be achieved while successful scan shifting carrying out.

Our implementations were tested with four selected ISCAS89 benchmarks using fast spice simulations with synopsys XA. We were able to achieve 30%-40% additional leakage power reduction compared to standalone MLV. The measured wake up time was always less than 0.25ns for all benchmarks while with standalone power gating this is more than a nano second or couple of nano seconds . Successful operation in scan mode and state retention of flops after standby mode were also verified. Rough estimate in area increment due to newly added infrastructure was also carried out.

List of Figures

List of Tables

List of Abbreviations

CF	Constant Field
CGPG	Coarse Grain Power Gating
CMOS	Complementary Metal Oxide Semiconductor
CV	Constant Voltage
DIBL	Drain Induced Barrier Lowering
DSM	Deep Sub Micron
EDA	Electronic Design Automation
FGPG	Fine Grain Power Gating
LECTOR	Leakage Control Transistors
MLV	Minimum Leakage Vector
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
MTCMOS	Multi Threshold CMOS
NMOS	N-type MOSFET
PDA	Personal Digital Assistants
PMOS	P-type MOSFET
SOC	System On Chip
VLSI	Very Large System Integration
VTCMOS	Variable Threshold CMOS

1. INTRODUCTION

1.1 CMOS for low power VLSI design

From 1960s, the time where the integration circuits were invented, down scaling of silicon technology helps to satisfy the increasing user demands for more functionality and high performance with a low cost. Initially power consumption was not a big issue and didn't take much attention. The rapid development in VLSI technology facilitates to fabricate a complete system in a single chip(SOC). As a result many portable devices like laptops, cellular phones, PDAs were introduced. As these portable devices are powered by batteries, power was recognized as critical parameter of SOC design.

The increasing demand for portable systems lead to consider the increment of battery life as a major requirement in SOC design because with the limited battery life systems have to fulfil high demands which requires high power consumption.

Though battery industry has already made successful tryouts to develop batteries with a higher energy capacity than conventional Nickel-Cadmium (NiCad) batteries, it doesn't cause to have very high energy capacity. This is considered as a primary reasons due to which portable applications should be incorporated with low power design techniques in the VLSI domain.

High power dissipation is a very bad thing in all kinds of DSM technologies. (DSM refers to technology nodes whose transistor length is lesser than 180um). As an example, Pentium IV processor developed in 2000s which was based on DSM technology, when operates at 3.4GHz, it consumes 130W of power with 1.3V supply [1]

Reliability is another reason which encourages low power design. As technology scaling down, current density also increases with power density. Due to Large current densities, problems like hot carrier induced device degradation and electro migration are occurred [1]. In addition, the heat gradient across the chip causing thermal and mechanical stress leads to early breakdown [1]. So to maintain an enhanced reliability of the system, power reduction is a very important thing.

VLSI/SOC design allows different digital circuit modules into a single chip. For example mobile phone which is comprised with CPU, RAM, camera module, GPS module can introduced. Basic unit of building Digital circuit is the transistor (MOSFET type more specifically). So VLSI chip contains millions of transistors fabricated on a IC. Obviously designers can’t design the ICs at the transistor level. They require more abstracted view on combining transistors to obtain the desired functionality. That abstraction is the well-known gate level abstraction which provides Boolean logic gates (Ex: AND, OR, NOT) build upon combining MOSFETs. Those logic gates are designed based on logic families (Ex: RTL, TTL, CMOS). Complementary Metal Oxide Semiconductor(CMOS) technique allows a high density of logic functions on a chip. CMOS has become the commonly used technology for VLSI designs due to two main reasons. One is it allows to have a higher logic density. Second reason is there is very small static current between power and ground when there is no switching activity taking place in logic gates. So simply VLSI power dissipation means CMOS power dissipation.

Although power dissipation is important for modern CMOS based VLSI designs, performance/speed and area are also very important considerations for a system. So the low power designs usually involve making tradeoff such as timing vs power, area vs power and so on. Reducing power while maintain the expected system performance but compromising the minimum required area for the system is a commonly addressed design problem in CMOS low power design techniques.

1.2 Motivation

Methodologies for low power design range from device/process level to the algorithmic level. Reducing the supply voltage (VDD) simultaneously to technology scaling is one of method which significantly reduces the power because VDD has a quadratic relation with dynamic power consumption. But to maintain performance as supply voltage reduces, the transistor threshold voltage (V_{th}) also should be reduced. But threshold voltage reduction leads to increase subthreshold leakage currents exponentially [2]. But when it comes to designs which are even more scaled from 45nm technology, in addition to subthreshold leakage current the gate leakages are also significant.

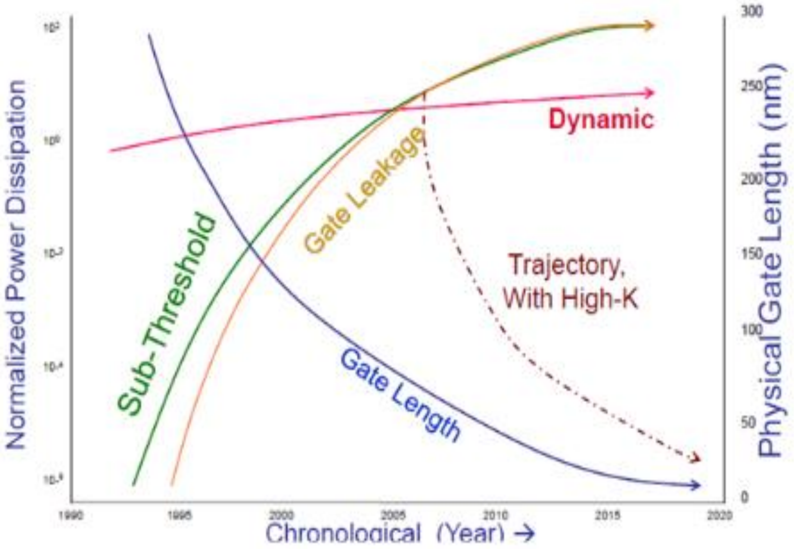


Figure 1.1 – Leakage and dynamic power dissipation with technology scaling

Source: Adapted from [4]

In today's world with the highly scaled nanometer technologies (gate length below 90nm) the leakage power component is more significant compared to dynamic power consumption. Figure 1.1 shows the contribution of leakage and dynamic power as technology scales down [4].

So the investigation leakage power reduction techniques has become a major research area in CMOS power reduction. So in this thesis we are focusing on leakage power reduction. When analyzing leakage power reduction techniques, we have seen that power gating is the best approach. But power gating of a whole system causes to lose the system state. So if we require to retain the previous system state once the system is powered again after power gating it requires to add additional circuities for data retention. In addition to data retention issue, power gating has another drawback. That is, when power is provided back to the system after power gating (in a wakeup) it requires some considerable time (from couple of Nano seconds to milliseconds based on the size of the gated design/module) to settle the system avoiding ground bounce which causes to have undefined logic states otherwise. Same time in a wakeup cycle, the circuit should not be powered at once. Because say if we have done the ground (VSS) power gating and before power gating If a logic gate's output was '0' due to VSS gating now it's output is partially charged up to VDD. So when the power is fed back to the system the charged output of this logic gate is connected back to the ground and the output charge continues to discharge. If the design has thousands of such gates then all those gates start to discharge their charged outputs at once causing a high current flow across the circuits leading to high heat dissipation finally creating damages to the circuit elements. To avoid this the system should be powered up again portion by portion in sequential fashion which adds another delay for the normal operation after power gating.

VLSI system is consisting of many modules. When one module is active one or many other modules might be inactive. For example, consider a mobile phone. Camera module or video module are activated when those are started to get use. Until then those are in idle state. Though there is no any dynamic power consumption of these idling modules there is a leakage power dissipation. Hence if these modules can be power gated we can reduce the leakage power. If the previous state of these modules should be retained after power is given back after power gating, then addition of retention logic in addition to power gating is required. At the same time longer settle time after powering up after power gating reduces the availability of those modules for the system once system wants them. For example, say a system requires a floating point calculating module rarely. But the calculations must happen as soon as input comes because these calculated values might be the required inputs for some control system. In that case higher power up time after power gating is not acceptable. Throughout this thesis we will use the word **‘standby mode’** to refer to the idling mode of the modules which requires state retention and short settle time after all kind of treatments done for leakage power reduction in idle mode of the desired module.

So the motivation for this research is to investigate the applicability of power gating technique in combination with some other leakage power reduction technique to achieve more leakage power reduction compared to other leakage power reduction technique while minimizing drawbacks (higher settle time after power reduction, state retention) of power gating technique.

1.3 Contributions

The main contribution of this thesis can be summarized as follows

- Instead of implementing full chip power gating to achieve leakage power reduction when a module is in standby mode we combine a leakage reduction technique called Minimum Leakage Vector (MLV) with power gating so that power gating based power reduction is done only around higher leakage power dissipating points of the module which are recognized after applying MLV for the combinational portion of the circuit. With this the intention is to achieve high leakage power reduction compared to MLV while mitigating the drawbacks of power gating technique.
- Propose a new method to feed MLV with Modified Flip flops and Input ports so that in standby mode in addition to leakage reduction in combinational logic, leakage reduction of Flops can also be achieved while preserving the state of the flops which exist prior to power gating.
- Extend above 2 so that MLV can be fed to combinational logic in scan mode which helps to achieve power reduction while normal scan shift can happen through the QN (inverted output of the flop) of the Flops while Q (output of the Flop) is feeding to combinational logic which is blocked by setting MLV at flop outputs in the scan mode.

1.4 Thesis Organization

Chapter 2 describes the background for the thesis. It covers many information found with literature survey including power dissipation mechanisms and leakage power reduction techniques. Chapter 3 does a thorough study on Minimum Leakage Vector, Power Gating and Data Retention which are three main bases for this research work discuss about data retention techniques. Chapter 4 describe the research approach, methodology, and results of the research. Chapter 5 describes conclusions and suggestions for Future work. In Appendix it contains some of the computer programs and scripts which were used in this research.

2. BACKGROUND

2.1 CMOS Power Dissipation

Mechanisms of CMOS power dissipation can be divided into two classes: dynamic power dissipation and static power dissipation.

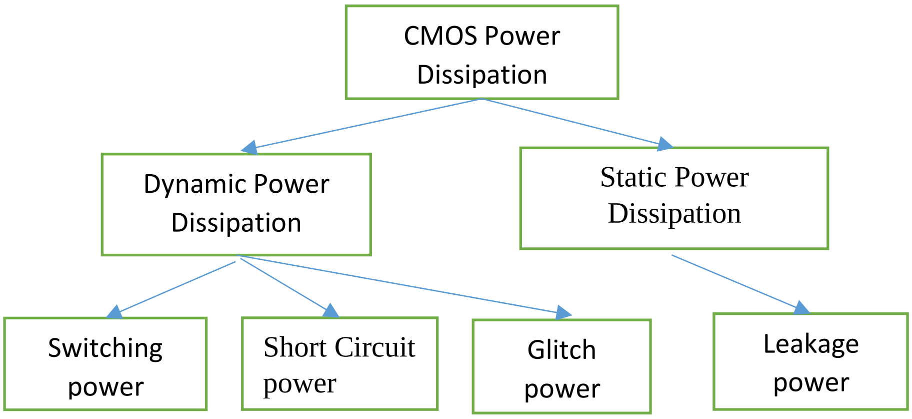


Figure 2.1 CMOS Power Dissipation Mechanisms

Dynamic power dissipation comes to play when circuit is operational. That is when the circuit it performing some task on some data. As circuit is non operational or inactive or the circuit is in power gated mode, static power dissipation becomes an issue. Figure 2.1 Summarizes these power dissipation mechanisms [2], [3].

2.1.1 Dynamic Power Dissipation

As illustrated in Figure 2.1, three sub categories can be identified under Dynamic power dissipation. Those are switched power dissipation, short-circuit power dissipation and glitch power dissipation.

2.1.1.1 Switch power Dissipation

To transmit information in CMOS circuits it is required to have repeated charging and discharging. Switching power dissipation occurs as result of this repetitive charging and discharging. Figure 2.2 shows how the switching power dissipation happens in a CMOS inverter.

As output logic level is transitioning from low to high, the capacitance C_L is charged through PMOS. While charging part of the energy supplies by power supplies losses due to resistance of PMOS and other part is stored in C_L. As output transits from high to low, C_L is discharged through NMOS. Due to resistance of NMOS, stored energy in C_L losses.

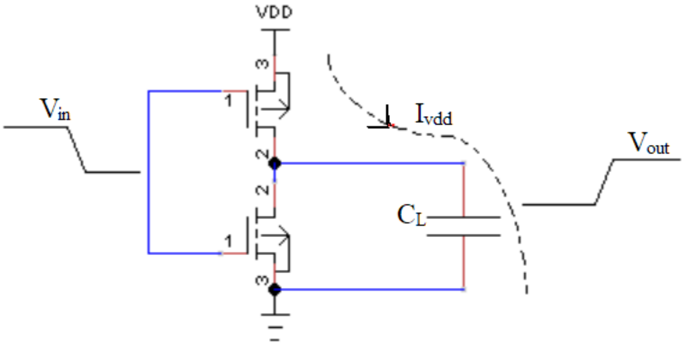


Figure 2.2 CMOS inverter for switching power analysis

so the switching power of the inverter can be calculated with equation 2.1 [2].

$$P_{dynamic\text{-}switching} = \alpha C_L V_{dd}^2 f \qquad \text{(eq. 2.1)}$$

Where, C_L = total Load capacitance

V_{dd} = supply Voltage

f = operating frequency

α = switching activity factor for the gate

The switching activity factor α is the probability that logic gate's output would be at logic one for the considered period of the operating clock. of the operating The parameter α depends on many factors like logic style, statistics of the input signal, boolean function of the logic gate.

2.1.1.2 Short Circuit Power Dissipation

Non-zero rise times and fall times in real circuits it causes to conduct both the PMOS and NMOS of the CMOS gate simultaneously. The short circuit current of CMOS inverter is shown in Figure 2.3.

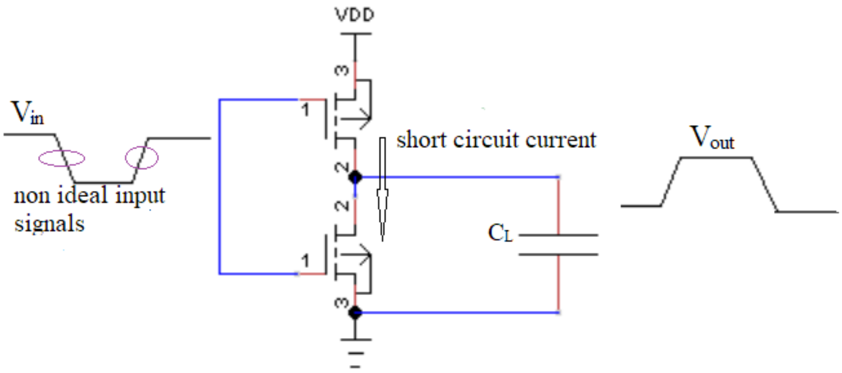


Figure 2.3 Short Circuit current of CMOS inverter

Short circuit current can be calculated using the equation 2.2, where constant *K* depends on the size of the transistors size and technology parameters, V_{th} is the threshold voltage, *Vdd* is the supply voltage, τ is the rise or fall time of the input signal, and *f* is the clock/operating frequency. According to the equation 2.2 it is obvious that there is linear dependency between τ and short circuit power dissipation. Hence reduction of transition times of input signals help to decrease the short circuit current. However a lesser short circuit current dissipation can be achieved with a increased load capacitance too [1]. But still, there is a tradeoff between this kind of short circuit current reduction method and a system performance because the propagation delay causes to increases with high load capacitances.

$$P_{\text{short-circuit}} = K\,(V_{dd} - 2V_{th})^3 \tau f \qquad \text{(eq. 2.2)}$$

2.1.1.3 Glitch power dissipation

Unnecessary signal transitions which do not contribute to flow any useful information through the logic gates are recognized as glitches. Generated glitches and propagated glitches are the two major categories. If there is a time skew among the input signals which are reaching toward a gate, at the output, there is an obvious risk of having a generated glitch. If any output change occurred due to glitch reached to an input port, it causes to create a propagate glitch. The glitch percentage of a circuit which is comprised with many combinational gates depends on the logic function, the logic depth, and the gate fanouts of the circuit.

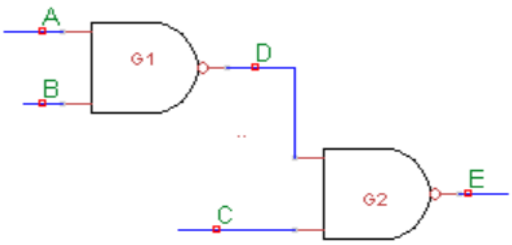


Figure 2.4 Multi-level NAND network for a glitch at node E

In Figure 2.4, When these two NAND gates have the same propagation delay and all A,B,C input signals arrive at the same time, the output net is suffered from glitching. This is shown in Figure 2.5 [1].

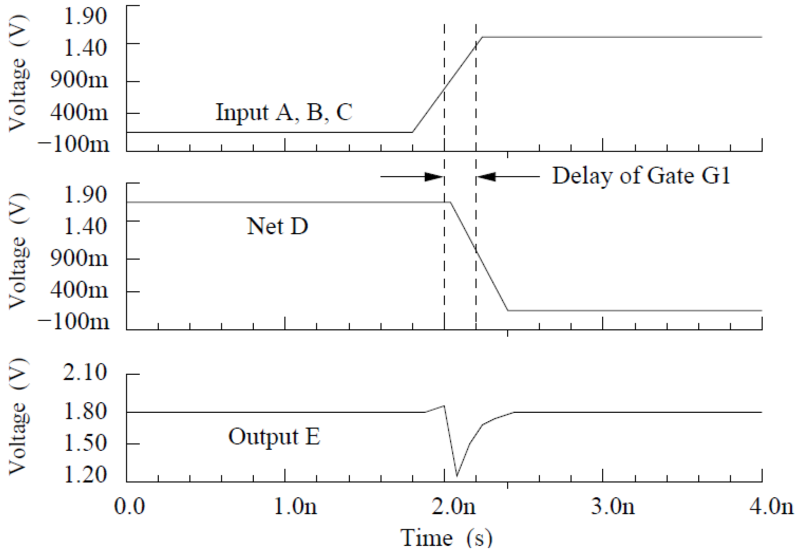


Figure 2.5 Signal glitch illustraion in a Multi-level CMOS circuit

Source: Adapted from [1]

2.1.2 Static power dissipation

In CMOS leakage currents which are appeared when the system is idle are the main cause for static power dissipation. Theoretically, there shouldn’t be any power consumption in CMOS when there are no output transitions in logic gates (or circuit is idling). This is because, either one of pull-up or pull-down networks are always turned off which prevents static current flow. But in reality, always there is a leakage current which leads to some power consumption by CMOS gates. For the static power consumption, the contribution of a single logic gate is extremely small, but with millions of such logic gates, the total leakage current becomes very significant. Further, as transistors gets more and more smaller in size the doping levels are also required to be increased causing a threshold voltage reduction which leads to increment in leakage current.

Different mechanisms that contributes for leakage current in a short channel CMOS transistor are illustrated in Figure 2.6 [2].

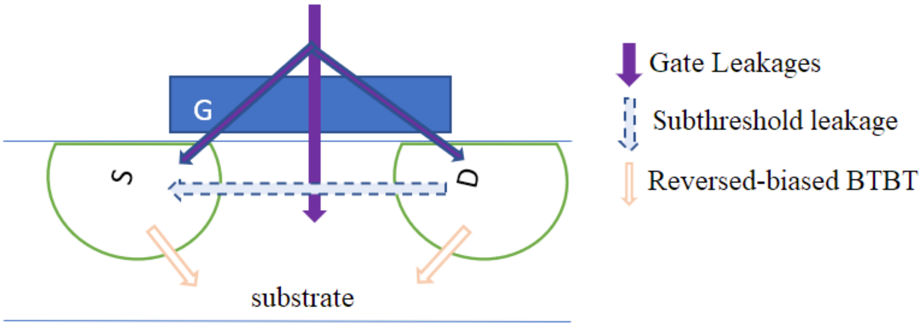


Figure 2.6 Major Leakage Mechanisms in MOS transistor

2.1.2.1 Subthreshold Leakage

To reduce the dynamic power consumption as technology scales, Supply voltage is required to be lowered. Simultaneously the Threshold voltage(V_{th}) of the transistor also required to be lowered to maintain higher current density otherwise which affect the switching speed of the transistor. But, decrease in V_{th} scaling leads to increase subthreshold leakage currents. In weak inversion region of a transistor, the current flows between drain and source is referred to as subthreshold current. In weak inversion, concentration of minority carriers of the MOSFET channel is almost zero, also the channel has no horizontal electric field. But because of drain to source voltage a small longitudinal electric field exists. Under this electric field, between the drain and the source of MOSFET carriers are moved by diffusion . This weak inversion diffusion current is the major contributor for the subthreshold current and which shows exponential dependency to both gate-to-source voltage(V_{gs}) and V_{th}. Equation 2.3 is the basic equation for subthreshold current calculation which is developed based on BSIM MOS transistor model [6].

$$I_{subthresho\ ld} = I_0 e^{\frac{V_{gs}-V_{th}}{nV_T}}\left[1-e^{-\frac{V_{ds}}{V_T}}\right] \qquad \text{(eq. 2.3)}$$

Where I0 = $W\mu_0C_{ox}V_T{}^2e^{1.8}/L$, V_T = KT/q is the thermal voltage, V_{th} is the threshold voltage V_{ds} and V_{gs} are the drain, source and gate, source voltages. L and W are effective transistor Length and Width , C_{ox} is for gate oxide capacitance, μ_0 is the carrier mobility and n is the subthreshold swing coefficient.

2.1.2.2 Gate Oxide Leakage

With the aggressive device scaling many short channel effects like V_{th} roll-off and DIBL are also required to be taken care of and reduced [7]. So to decrease the effect of those short channel effects gate oxide thickness of transistors are also reduced. With decreased oxide thickness the lateral electron field between gate and bulk is risen causing to direct tunneling of electrons from bulk/source/drain to gate across the gate insulator generating gate oxide tunneling/leakage current. The gate oxide leakage current is modeled as in Equation 2.4 [2].

$$I_{gate} = W.L.A\left(\frac{V_{ox}}{t_{ox}}\right)^2 \exp\left(\frac{-B\left(1-\left(1-\frac{V_{ox}}{\phi_{ox}}\right)^{3/2}\right)}{\frac{V_{ox}}{t_{ox}}}\right) \qquad \text{(eq. 2.4)}$$

where W and L are the effective transistor width and length, A= $q^3/16\pi^2h\ \phi_{ox}$, $B=4\pi\sqrt{2mox}\ \phi_{ox}{}^{3/2}/3hq$, m_{ox} is the effective mass of the tunneling particle ϕ_{ox} is the tunneling barrier height t_{ox} is the oxide thickness, h is 1/2π times planck's constant and q is the electron charge. Downscaling of gatelengths from 45nm shows significant gate current sometimes even larger than subthreshold currents in bulk CMOS.

2.1.2.3 Reverse biased Diode Leakage

When the p-n junction between drain and bulk of the transistor is reversed biased, it stills has a current conduction which is drawn from the power supply. This is know as the reverse saturation current or reversed biased diode leakage current. The reversed biased p-n junction current can be explained with the equation 2.5 [2].

$$I_{reverse} = A.\, J_s\, (e^{qVbias/KT} - 1) \qquad \text{(eq. 2.5)}$$

where,

V_{bias} - Reverse bias voltage across the junction

J_S - Reverse saturation current density

A - junction area.

T – temperature

With a minimized junction area it is possible to maintain lower reversed biased diode leakage. It has exponential dependency with temperature while J_s also significantly increases at high temperatures.

This thesis is focused on the Leakage power reduction through reduction of leakage current. In that case as subthreshold current and gate leakage current are the more significant contributors for leakage, we focus on reduction of both subthreshold and gate leakage currents.

2.2 Technology Scaling

As Moore’s law explained ‘number of transistors per square inch doubles with every 18 months’ from 1960s CMOS technology is down scaling at a rapid rate. To have a high level of integration, it requires to have a reduction of major dimensions of a typical MOSFET while increasing the doping level. Figure 2.7 illustrates the dimension scaling with scaling factor S [1]. There are two types of scaling strategies for MOSFETs. Those are constant field (CF) and constant voltage scaling(CV). In CF, power supply together with vertical and horizontal dimensions of the transistor along are scaled down by a factor of S. Meanwhile it requires to increase doping densities by the same factor S to maintain magnitude of required electric field. In CV both terminal and power supply voltages are remained unchanged while increasing the doping densities by a factor of S^2 so that it can maintain the charge-field relation.

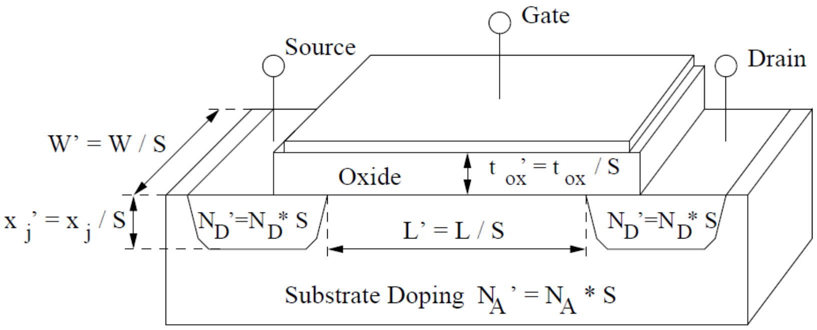


Figure 2.7 MOSFET Scaling with scaling factor S

Source: Adapted from [1]

Table 2.1 summarizes the scaling factors of MOSFET for all key dimensions and doping densities and power supply for above mentioned two scaling strategies [1].

Parameter	Constant Field (CF)	Constant Voltage (CV)
Channel Length (L)	1/S	1/S
Channel Width (W)	1/S	1/S
Gate Oxide thickness (tox)	1/S	1/S
Junction Depth (xj)	1/S	1/S
Power supply Voltage (VDD)	1/S	1
Threshold Voltage (Vth)	1/S	1
Doping Densities (NA , ND)	S	S^2
Oxide Capacitance (Cox)	S	S
Drain Current (I_D)	1/S	S
Delay (τ)	1/S	$1/S^2$
Power dissipation (P)	$1/S^2$	S
Leakage Power ($P_{leakage}$)	Exp	1
Power density (P/Area)	1	S^3
Power delay product	$1/S^3$	1/S

Table 2.1 MOSFET device scaling characteristics

2.3 CMOS Leakage Power Reduction Techniques

In this subsection we study commonly available techniques for power reduction.

2.3.1 transistor Stacking

When two or more series-connected transistors become “off” state, this is known as an off transistor stack which results to conduct a low leakage current in idle mode due to the self-reverse biasing effects. Figure 2.8 shows a 2 transistor stack. When both transistor M2 and M1 are turned off, because of the small drain current, the intermediate node voltage V_x is positive. As a result,

- *Vgs1* which is the Gate-to-source voltage of M1 transistor becomes negative causing subthreshold leakage reduction
- Negative body-source voltage (*Vbs1*) of transistor M1, increases the threshold voltage (*Vth*) of M1 which leads to subthreshold leakage current reduction
- As drain--source potential (*Vds1*) of M1 decreases, due to the DIBL effect *Vth* of M1 get increases, and hence subthreshold leakage is reduced. (less Drain Induced Barrier Lowering) of M1, reduces the subthreshold leakage. V_x

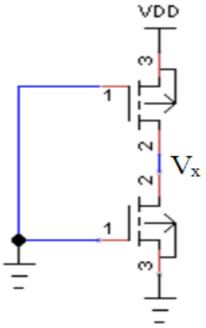
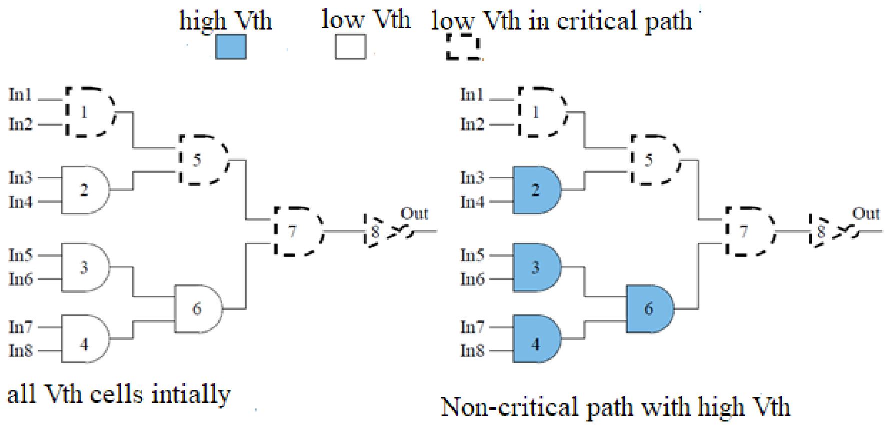


Figure 2.8 Two NMOS off-transistor stack

2.3.2 Dual V_{th} Partitioning

Since multiple threshold voltages are provided for transistors, a dual Vth process can be introduced to design circuits so that low Vth cells can be used to build critical timing paths to achieve higher performance while high Vth cells can be used to build non-critical timing paths for low leakage power consumption [2].



all Vth cells intially

Non-critical path with high Vth

Figure 2.9 Dual Vth Partitioning scheme

2.3.3 Reverse Body Biasing

Consider a NMOS transistor where substrate is biased with a higher voltage than its source voltage. In this case transistor is reversed body biased due to which transistor threshold voltage is increased which leads to a leakage current reduction of transistor. The Relationship between transistor threshold voltage and the body effect is illustrated with equation 2.6 [8].

$$V_{TH} = V_{T0_n} + \gamma\left(\sqrt{2\phi_f + V_{SB}} - \sqrt{2\phi_f}\right) \qquad \text{(eq. 2.6)}$$

Where, V_{BS} is the source body/substrate voltage, V_{T0n} is the threshold voltage when V_{SB}=0, ϕ_f is a physical parameter ($2\phi f \approx 0.6V$ for NMOS and $0.75V$ for PMOS) and γ is a process parameter called body-effect parameter ($\gamma \approx 0.4V1/2$ for NMOS and $-0.5V1/2$ for PMOS).

2.3.4 Variable Threshold CMOS (VTCMOS)

Variable Threshold CMOS (VTCMOS) is a circuit design technique developed to reduce standby leakage currents in low VDD and low V_{th} applications. This uses the concept of reverse body biasing to achieve leakage reduction. In this case the CMOS circuit operates as usual with the expected performance in the active mode but in the standby mode, a substrate bias voltage is generated using a separate control circuit. Due to this generated bias voltage, reversed body biased effect is experienced by the circuit causing a leakage reduction. Figure 2.10 shows VTCMOS concept for Inverter Circuit [2].

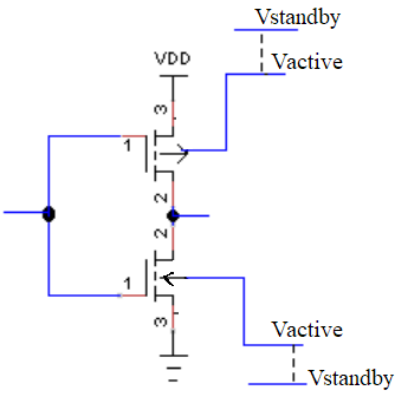


Figure 2.10 VTCMOS for standby leakage reduction

2.3.5 Minimum Leakage Vector(MLV)

Based on the different input combinations same combinational logic gate shows different leakage currents. This is basically due to the transistor stacking effect which causes a subthreshold leakage reduction. In addition, the location of off transistors causes to generate different gate leakages [13]. Table 2.2 shows the leakage subthreshold leakage of NAND gate in 130nm process [2].

Input vector	Subthreshold leakage current (nA)
00	3.94
01	15.25
10	13.65
11	4.57

Table 2.2 subthreshold leakage current of NAND gate in 130nm process

So in the standby mode if it is possible to feed the MLV for the circuit the leakage current can be reduced. But for a large combinational circuit finding the MLV is challengeable. So MLV concept is used together with some other approaches like gate modifications for leakage reduction. This concept is one of the basis for our research from which we extend our method for more power reduction. This method is more studied in chapter 3.

2.3.6 Multi Threshold CMOS

In MTCMOS, low V_{th} cells are used for logic circuit design and high Vth transistors are used to isolated permanent voltage supply from the logic circuit. Turning these high V_{th} transistors in standby mode causes reduce leakage power [2]. Though it has the advantage of having a leakage reduction, state loss in and after standby mode is major drawback. Sametime added gating transistors causes to add an extra area. Figure 2.11 shows the MTCMOS structure for simple combinational circuit. In original MTCMOS concept both PMOS and NMOS are used sleep transistors as in 2.11.a. But there two different versions through which also a significant leakage reduction can be achieved as shown in 2.11.b and 2.11.c.

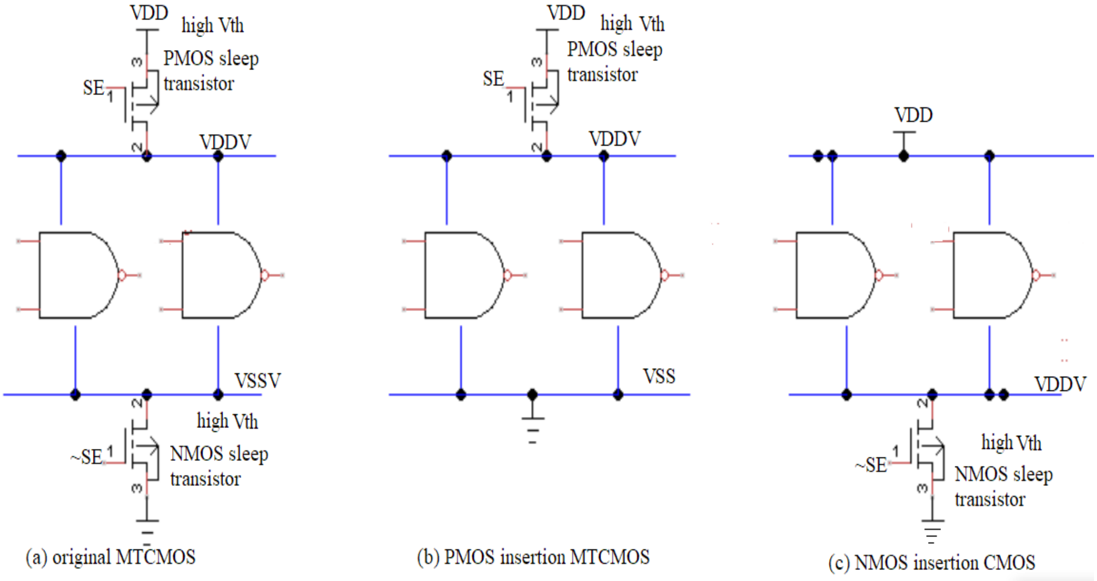


Figure 2.11 MTCMOS technique for leakage power reduction

2.3.7 Power gating

The MTCMOS technique is the basis for the power gating. In Power gating, PMOS or NMOS transistors are used as sleep transistors which are inserted in between permanent power supply and logic gates. In active mode these sleep transistors switched on while in the standby mode those are switched off causing a reduced current flow. We use this power gating concept with MLV in our research to have standby power reduction. This concept is more examined with Chapter 3.

2.3.8 LECTOR concept

LECTOR technique helps to reduce both standby leakage power as well as run time leakage power. With LECTOR two high threshold voltage transistors called leakage controller transistors(LCT) are inserted in between pullup and pulldown network of a logic gate so that PMOS LCT is directly connected to the pullup network and NMOS LCT is directly connected to the pulldown network. Once inserted for any input vector of the logic gate once of these LCTs are always closer to its cutoff voltage. As a result the resistor between VDD and GND increases. Hence a reduction in leakage current. Figure 2.12 shows the LECTOR concept [9].

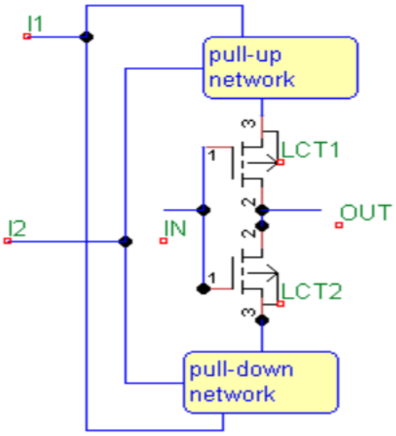


Figure 2.12 LECTOR technique for leakage reduction

3. ANALYSIS OF MLV, POWER GATING AND DATA RETENTION

This chapter thoroughly studies the techniques/concepts which are the basis for this research.

3.1 Minimum Leakage Vector

The MLV concept is introduced in chapter 2 subsection 3.5. The basic idea is to maintain the input combination for a combinational circuit which causes to dissipate the lowest leakage power when the circuit is idling. Though finding MLV for basic combinational logic circuit like NAND, NOR, AND, OR is easy, it is big challenge for large combination logic circuit which consist of millions of basic logic gates because it is difficult to find a common input vector for whole combinational circuit which leads to cause the MLV for each internal basic logic gates. Hence even after applying the MLV for large combinational logic some other methods are used in combination. Some of the approaches that researchers are investigated can be listed as follows.

3.1.1 Chose best input vector from set of Random vectors

Input sufficient number of randomly chosen input vectors for the inputs (here the inputs refer to primary inputs as well as sequential outputs which feed to combinational logic) of combinational circuit and chose the vector which gives the minimum leakage power. As per the previously research, for a large circuit the best MLV can be chosen with 10,000 distinct random vectors with 99% confidence that the vectors with less leakage is less than 0.5% of the entire vector population. Even with this chosen best vector the circuit still possess 15%-17% gates in worst leakage state [11].

3.1.2 Combination of Random MLV and Gate modification

Chose the best input vector for leakage reduction through a random search as above 3.1.1 and modify the worst leakage gates so that in the standby mode they give a low leakage while preserving the normal mode functionality of those gates as it is. This modification is done only if overall power reduction due to this modification of the gate itself and its fan-out gates are lesser than existing leakage. Once modification is confirmed then it requires to check the outputs of gates in the fan-out of the modified gate, and if any output change, add that output changed gate as worse case gate and add that to worse gate set so that algorithm will run for that gate also [12].

3.1.3 Dynamic programming based approach

Divide the Combinational circuitry into tree circuits and for each tree circuit identify the MLV using dynamic programming. When combining those tree circuits, it requires to resolve the conflicts in the combining branches; That is, say we have a net ‘A’ appears as the input of tree circuit-2 and by dynamic programing the value ‘0’ is assigned to this net to contribute as a bit of minimum leakage vector. The same net ‘A’ can be the output the of tree circuit-1. So once apply the its own MLV for tree circuit-1, this net ‘A’ achieves the value ‘1’. When combining these two tree circuits it introduces a conflict for net A which occurs once MLV applies. The conflict can be resolved by modifying the driver of net A so that with the input of standby signal it outputs the value ‘0’ which is required input value for MLV of tree circuit-2 [13]. Figure 3.1 shows a usage of new logic gate to combine two tree circuits so that both tree circuits prevail their MLVs in standby mode.

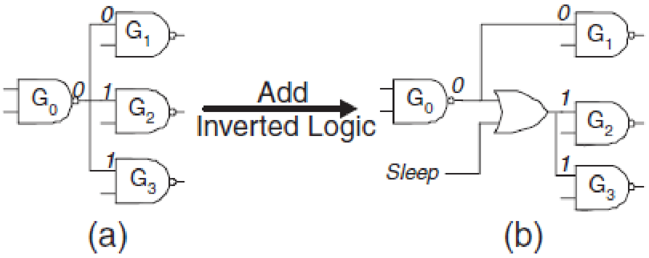


Figure 3.1 Logic Insertion to combine tree circuits to feed MLV in standby Mode

3.1.4 Use of optimization algorithms

Usage of optimization algorithms to find a better MLV for combinational circuit based on a randomly chosen initial input vector. After modelling the MLV assignment problem as an optimization problem, Researchers have used many heuristics to obtain a better optimized MLV. Genetic Algorithms, Differential evolution, and particle swarm optimization are some of the attempts tried out in the literature [14], [15].

3.1.5 MLV feeding Mechanisms

3.1.5.1 Using blocking logic

Here at the output of registers/primary inputs a combinational gate introduced so that in the standby mode the output this combination logic gate feeds the required MLV vector digit from its input flop. For example, in the standby mode if this flop requires to output ‘0’ as a part of MLV at output of flop the AND logic shown in Figure 3.2 can be introduced at the output of the flop [16].

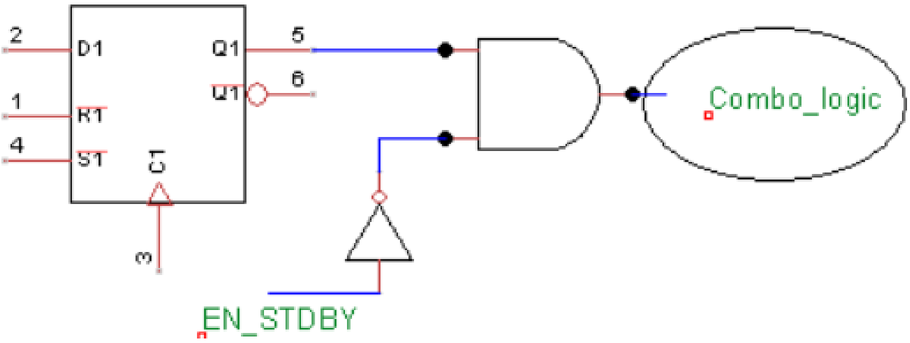


Figure 3.2 Using Blocking Logic to feed value ‘0’ at flop output

3.1.5.2 Using a Mux

In the Standby mode, the mux will feed the MLV to the combinational logic circuitry while in the normal operation MUX outputs the value which is fed by relevant

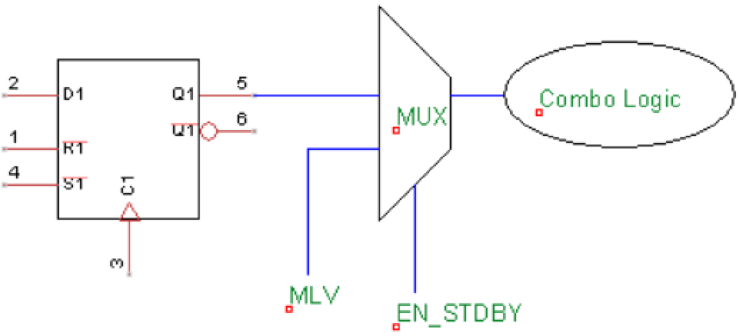


Figure 3.3 Using MUXes to feed MLV

driving flop of it. Figure 3.2 shows this MUX based logic [16].

3.1.5.3 Using scan chain

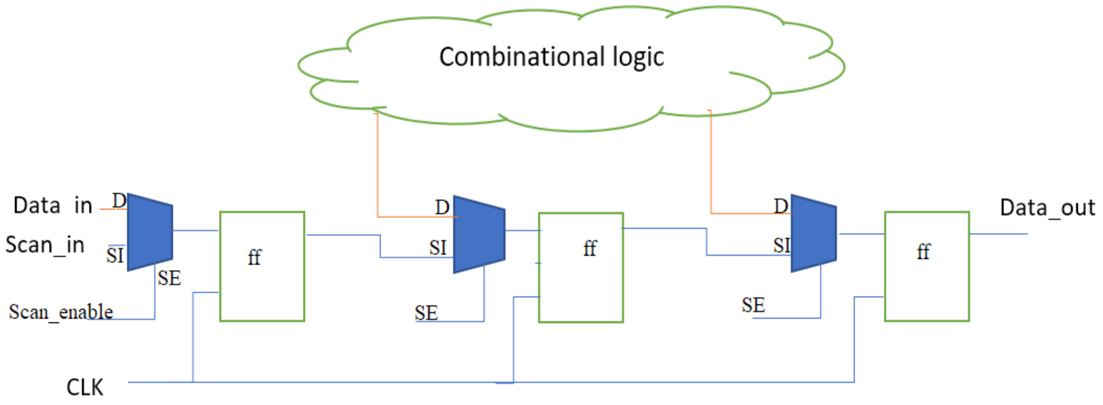


Figure 3.4 Scan Chain for initializing Flops

Just before switching into standby mode the scan chain is enabled and the MLV is shifted accordingly through the scan chain [16]. In this approach all the inputs for the combinational logic of the circuit are assumed to be obtained through flops. But the problem with this approach is the flop values which were latched before going into standby mode are got replaced by the MLV values for the standby mode [10]. Refers to Figure 3.4, the MLV can be fed from Scan In port after asserting scan enable and using the clocking of CLK.

3.1.5.4 Using Scan Enable and First level gating

To enable standby mode, instead of using separate signal, with these approach researchers have used scan enable signal to switch in to standby mode. As circuit is switching to scan mode, the immediate fan-out (level-1 fan-out) combinational logic gates are power gated and for their output the relevant minimum leakage values are fed. With this approach problem with losing the stored data of flops has been resolved. With this approach problem with losing the stored data of flops has been resolved. The pullups and pull downs to feed MLV can be reduced by reducing the fan-out of flops. Figure 3.5 shows the First level gating concept [17]. As part of MLV value ‘0’ is fed with a NMOS pull down at out1 whose control signal is ‘GATING CONTROL’ itself.

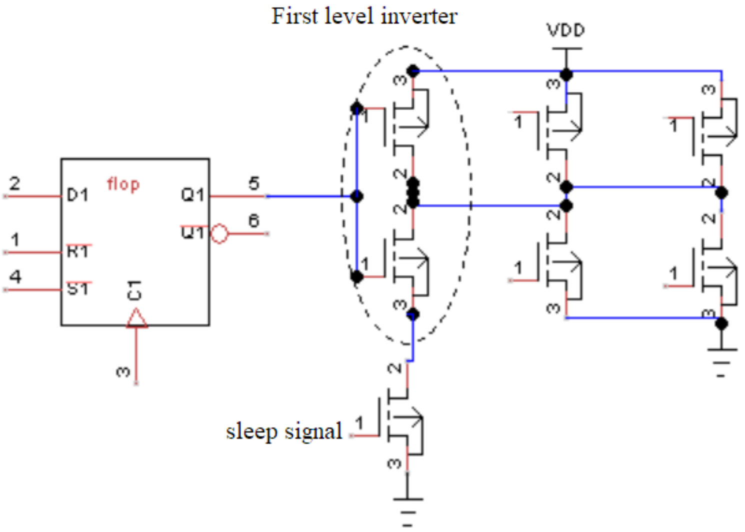


Figure 3.5 First Level gating concept

3.1.5.5 Use a NAND / NOR logic instead of last stage inverter of the flop

In the Figure 3.6 it shows how flop can be modified to feed the expected value for the MLV from this flop. With this, it can be seen the output inverter is replaced with NOR gate so that this NOR gate has ‘Standby’ signal as another signal.

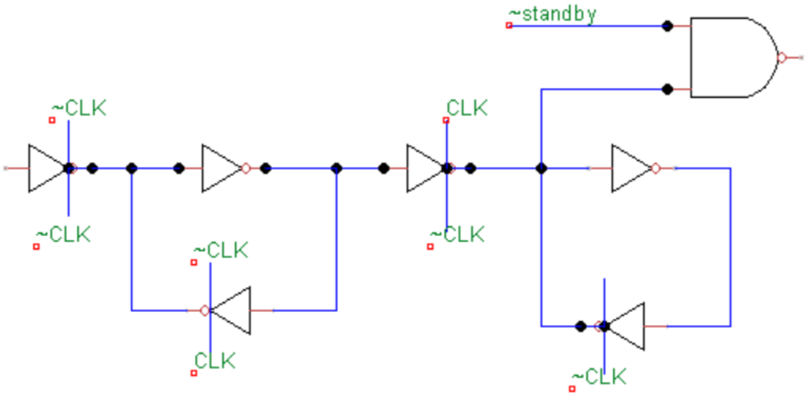


Figure 3.6 Flop’s Output Buffer Modification to Feed value ‘1’ for MLV

Hence in the Standby mode the flop always outputs value ‘1’ irrespective of the stored value within it [18].

3.2 Power Gating

As already introduce in 2.3.7 the power gating is the processes of disconnecting the power rails from the logic gates/circuit in the sleep/standby mode. Figure 3.7 shows the basic concept of using header and footer switches in Power gating. Turning off the sleep transistors provide reduced leakage due to two major reasons. First, the reduced width compared to transistors being gated causes to provide a reduction in leakage current drawn from the supply in standby mode of the circuit. of the sleep transistor is usually less than total width of transistors being gated. Second reason is the stacking effect of transistors (which explained in 2.3.1) due to which a reduced leakage current.

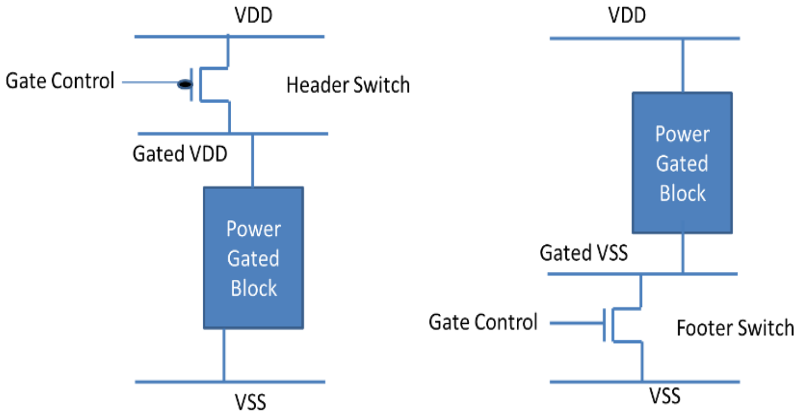


Figure 3.7 Use of Header and Footer Switches in Power Gating

In implementation Perspective, power gating can be divided into two categories; Fine Grain Power Gating and Coarse Grain Power Gating [19].

3.2.1 Fine Grain Power Gating(FGPG)

With Fine grain power gating, a sleep transistor is also included with every standard cell. As a result the chip area is increased. With FGPG, power gating is done individually for each cell due to which hard to resolve problems like inter-cluster voltage variations are occurred. Since sleep transistor is part of a standard cell, standard cell designers/IP vendors should taken care of the challenges in sleep transistor design. With the existing tools these kinds of cell designs are also handled for implementation and validate for typical rules of standard cell designs.

3.2.2 Coarse Grain Power Gating(CGPG)

Though in FGPG, it inserts sleep transistor for each cell with CGPG set of sleep transistors are inserted in a ring style or grid style for a complete chip or module which encapsulates millions of standard cells. There standard cells are driven with virtual power networks as sleep transistors are appeared in between permanent power network and virtual power network. So power gating/sleep transistors are parts of power network rather than being a part of standard cells. Two implementation structures of CGPG can be explained as follows.

- Based on Ring of sleep transistors [19]

  The sleep transistors are inserted around the module which should be power gated and the power gates are switched as a ring. Compared to grid style this method can be easily implemented.

- Based on Columns of sleep transistors [19]

  Instead of inserting the power gates around a module, with this implementations sleep transistors are inserted within the module constrained to identified grids/columns. High metal layers are provided with permanent power while virtual power networks are in lower metal layers. Reduced IR drop is a main advantage of this method compared to ring style.

3.2.3 Power Gating Parameters

To implement the power gating successfully it requires to consider following parameters with their values carefully [19].

- Size of the Power Gate:

  The size of the power gate must be selected to handle the amount of switching current at any point of given time. The gate must be bigger such that there is no measurable voltage drop due to the gate. The size of the gate is selected to be around 3 times of the switching capacitance as a Rule of Thumb. The Designers can also choose header (PMOS) or footer (NMOS) gate for the designing circuits. For the same switching current NMOS footer gates are to be smaller in area. Switching current can be accurately measured by using of Dynamic power analysis tools and can predict the size for the power gating.

- Slew rate

  Slew rate is an important parameter to determine power gating efficiency in power gating. It takes more time to switch off and switch-on the circuit when the slew rate of the controlling signal increases.

- Switching Capacitance

  If a large amount of the circuit is switched simultaneously, the resulting "rush current" can compromise the power network integrity due to which the switching capacitance is identified as a critical design parameter for power gating. In order to prevent undesired rush current the sleep transistors are turn on sequentially or in a distributed manner.

- Leakage of Power Gate:

  Power gating is implemented by using the active transistors to maximize power savings the attention should be given to leakage power of power gates also.

3.2.4 Drawbacks in power gating

These are the major drawbacks with power gating technique.

3.2.4.1 Large wakeup times after power gating

After gating the internal nodes of the gated module is in floating/non-deterministic logic states. If this is Footer based gating once GND is connected, the partially charge output capacitances of gates start to discharge at once which causes a ground bounce which takes couple of nano seconds or sometimes milliseconds to settle down in deterministic logic states. In addition, an additional time is required as power is connected back in sequential fashion to avoid circuit damages due to high currents which might be flown otherwise this sequential power up is not implemented. These causes to increase the wakeup time after power gating which is undesired.

3.2.4.2 Problem with state retention

With power gating internal register states get loses. Hence addition circuitry is required to retain the prior power gating stage after the system wakeup after power gating.

3.2.4.3 performance drop due to on resistance

Due to the on resistance of the gating transistors, in the active mode of the circuit there is a IR drop from the actual supply voltage of the circuit. This supply voltage drop increases the propagation delay of logic gates hence the performance drops.

3.2.4.4 Challenges in sleep transistor design

The optimum transistor size, required number of sleep transistors are important to maintain a better performance and reduced standby leakage power of the entire circuit.

3.2.5 Zig-zag power gating

In Zigzag power gating a predefined input vector is fed to the combinational logic circuit just before power gating. Since this is a predefined vector we know the output voltage values of each logic gate. So if the output of logic gate is ‘1’ we can implement VSS gating without any output data change. And if the output of the logic gate is ‘0’ we can do the VDD gating without losing the output logic states of the logic gate. And this kind of power gating causes to very less wakeup time compared to

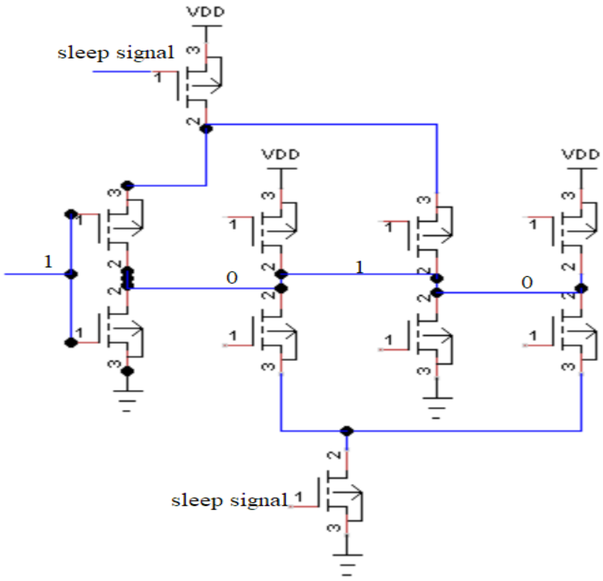


Figure 3.8 Zigzag power gating

typical power gating as there are no any floating logic states within the circuit once the power is fed back. Power gating done in this manner is called the zigzag power gating technique which is illustrated in Figure 3.8 [21].

3.3 Data Retention

As mentioned in 3.2.4.2 with power gating states of the gating module/values which were there in flops before power gating lose once power gating happen. Hence if the state should be retained after the power gating either the desired flops should not be power gated or some extra low power consuming circuitries should be added to the existing circuit to preserve the state even when the module is power gated as it is in sleep/standby mode. Here we are interested in data retention in registers/flip-flops. When considering retention registers there are three commonly used retention register types [20].

3.3.1 Single Control Live Save

In single control live save, standard master slave flip-flop is used with minimal modification to retain the state. Master latched which is built upon fast Low Vth transistors are power gated. Slave latched is always powered on which is designed with high Vth transistors. AND gate is used to isolate clock input during power down. Figure 3.9 illustrates the Single Control live slave concept. Three state buffers (T1) between latches are there to isolate slave input. In Figure 3.9 NRETAIN is the control signal to retain the saved value after power gating.

Minimal area usage of single control signal are the advantages of this method. Slow save latch, longer input data hold time, need to make sure clock is low when restoring data are some disadvantages of this mechanism.

3.3.2 Dual Control Balloon

Here standard master-slave latch is used with added retention Latch. Retention latch which is built with high Vth transistors are always powered on. There are two Control signals called SAVE and NRESTORE. The Dual Control Balloon technique is illustrated in Figure 3.10. Minimal leakage power, almost no performance loss compared to non-retaining flip-flop and Clock phase independent restore are the main advantages of this method. Area increment, compared to live-slave and using two control signals are the main two disadvantages.

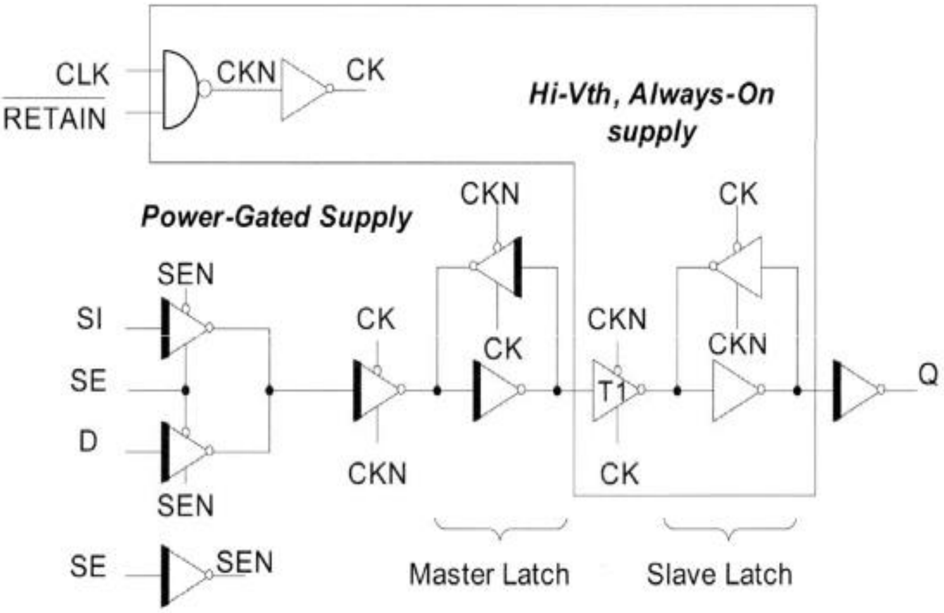


Figure 3.9 Single Control Live Save

Source: Adapted from [20]

3.3.3 Single Control Balloon

Here also Retention latch built with high Vth transistors is used. Same Control Signal NRET (as shown in Figure 3.11) is used to save during run time and restore on edge. Retention latch inputs are isolate by Three state buffers. Almost minimal leakage power, almost no performance loss, compared to non-retaining flip-flop, Clock phase independent restore and reduced system level dynamic power due to only one control signal network are the major advantages of this mechanism. Area increment compared to live-slave is a disadvantage of this method.

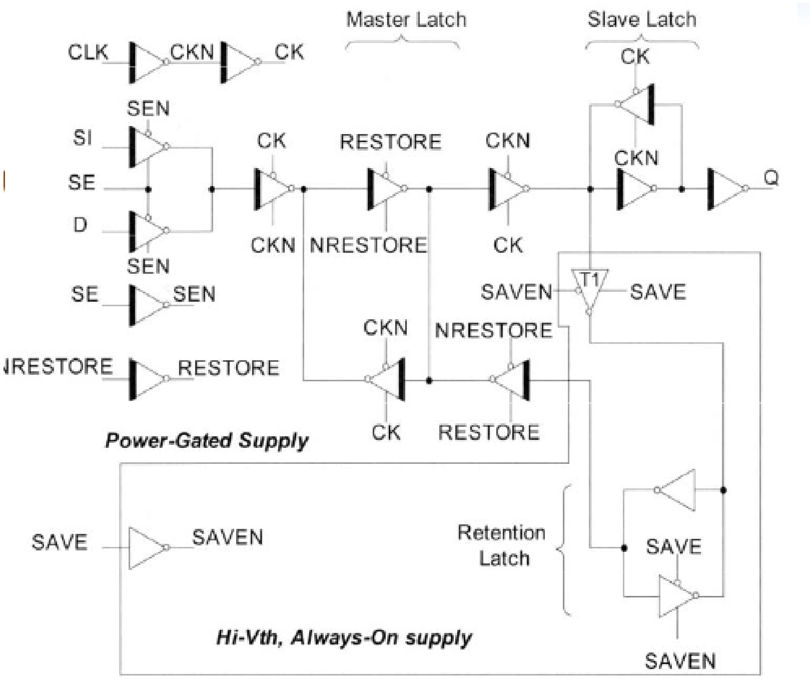


Figure 3.10 Dual Control Balloon for State Retention

Source: Adapted from [20]

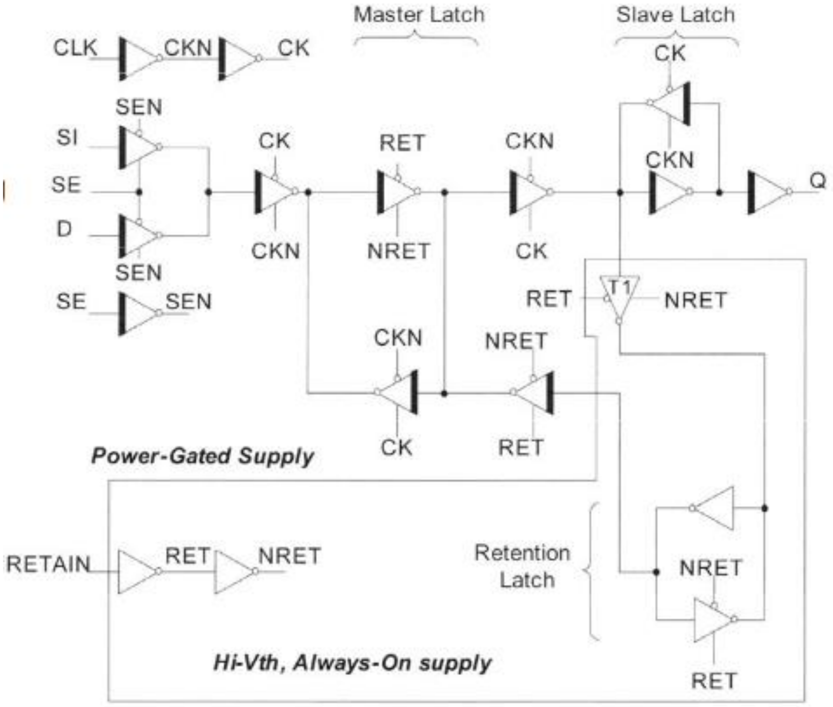


Figure 3.11 Single Control Balloon for data retention

Source: Adapted from [20]

4. METHODOLOGY AND TEST RESULTS

4.1 Research Approach

As already explained in the Section 3.1 Minimum Leakage Vector(MLV) is a leakage power reduction technique which reduces the leakage currents based on the input vector fed to the inputs of the combinational logic gates. Table 4.1 shows the subthreshold and gate leakages for 2 input NAND gate. The simulations were done based on nangate 45nm library [3] [22]. In Table 4.1, A1 is the input of NMOS transistor directly connected to gate output. According those results it seems that most of the time gate leakages are more significant than subthreshold leakages of gates. At the same time, we can clearly see based on the inputs of gates and their ordering the leakage current varies.

Input Vector(A1A2)	00	01	10	11
Subthreshold leakage(nA)	0.45	6.54	3.153	7.04
Gate Leakage (nA)	2.5	15.8	0.4	26.52

Table 4.1 Subthreshold and Gate Leakages of 2 input NAND gate

Not like for simple logic gates, Finding the MLV for a large combinational circuit is difficult because the best possible vector for many logic gates might be not the best for some other gates and hence the latter mentioned gates causes a high power consumption even after applying MLV. Hence in the literature researchers have carried out many approaches to feed an input vector which causes a less leakage power for these highly leaky gates also. For that they either have added new gates or have done modifications to the existing gates so that in the standby mode these gates to less leaky. Adding a new gate so that it causes to feed the MLV for a high leaky gate only is not better because this newly added gate and desired high leaky gate collectively might give a higher leakage though now highly leaky gate's individual leakage value reduces. Hence they have always considered for an overall leakage reduction around that high leaky gate after adding this new gate for leakage reduction by feeding the changed output value to some other nearby gates also. Even in the gate modification they have concerned about the same matter.

Since power gating always cause to have leakage power reduction our initial thought was to treat these gates with power gating so that output of the gating gate is still preserved as it is due to which there is no effect for other less leaky gates a no output change. And we can use the same gating transistor to provide the power gating for some other gates which outputs the same value as high leaky gate which causes to have more power reduction. In the sense of MLV we still require to do some change for the input of this leaky gate to have better power reduction. Normally the high leaky input vectors are the vectors which contains higher number of logic ‘1’ compared to logic ‘0’. So if we can do the VDD power gating for driver of this high leaky gate who feed logic value ‘1’ then we can feed a logic value close to ‘0’. With this idea we are encourage to do VDD power gating around high leaky gates. But the highly reduced output voltage of drivers causes to change the output values of the VDD gated gates. Figure 4.1.a shows CMOS NAND gate who was identified as a high leaky gate after feeding MLV, which is driven by an inverter. 4.1.b shows the VDD gating of the same 2 input NAND gate with the driving inverter. In Figure 4.2.a it shows the same NAND gate which driving more load than 4.1.a. Figure 4.2.b illustrates the VDD gating of desired NAND gate in 4.2.a with its driving inverter. Figure 4.2.b circuit is augmented by adding a partially pullup to ‘inv_out’ in 4.2.c.

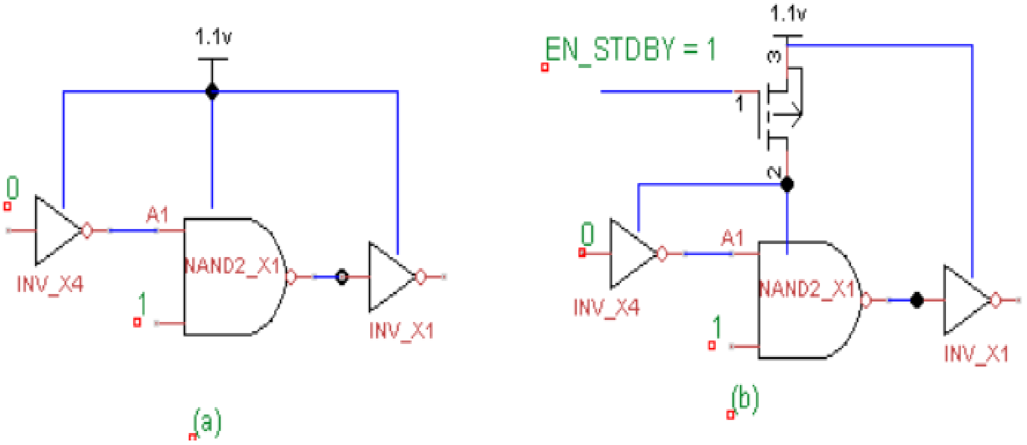


Figure 4.1 VDD gated NAND with its driver

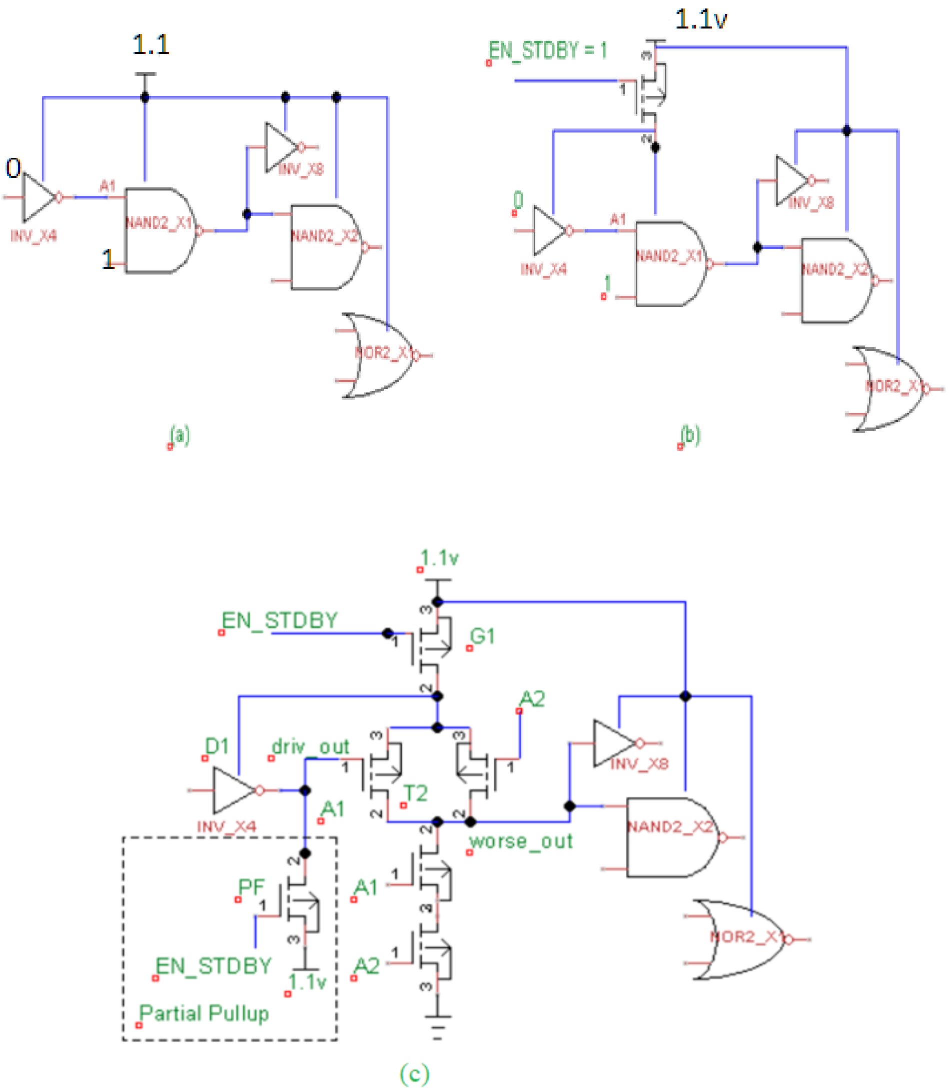


Figure 4.2 partial pullups in standby mode for VDD gated NAND gate

Inspired from the concept of LECTOR which is described in chapter 2 subsection 2.3.8, we can still maintain the same output level of NAND gate in Figure 4.1.b even with the power gating same as 4.1.b. But according 4.2.b when the driving capacitance of NAND gate increases the output voltage level increases from expected logic value ‘0’ which is undesired. Circuit in Figure 4.2.c is obtained by adding a partially pullup to ‘inv_out’ in 4.2.b. so Even with the driver gating and loading a higher capacitance load, by providing a near threshold voltage for the gated driver’s output we can still maintain the output voltage logic levels as expected. Partially reduced supply voltage inputs for a gate of a logic gate considerably reduces the gate leakage. Actually Figure 4.2.c shows how we can do the power gating of prospective gates including driver while providing a partially cut down supply voltage to the gated driver output. To provide only a partial voltage we have used a NMOS transmission gate as shown.

Circuit	Vinv_out (V)	Vnand_out (V)	Gate Leakage at A1 (nA)	Gate Leakage of partially pullup NMOS (nA)
4.1. a	1.1	0.00010	13.337	NA
4.2. a	1.1	0.00014	13.38	NA
4.1. b	0.02	0.095	0.0716	NA
4.2. b	0.097	0.1828	0.03417	NA
4.2. c	0.4997	0.00012	0.4955	0.116

Table 4.2 output voltage and gate leakages with power gating in standby mode

According to the results of spice simulations of circuits in Figure 4.1 and 4.2 we can see the gate leakages can also be reduced while maintain the load output as it is with partial pullups. Table 4.2 Summarizes these details.

So We are introducing three important terms which is frequently used in upcoming sections of the thesis.

Gated Group: The group which is formed by a set of logic gates whose output is 0 if their logic type is NAND NOR or NOT or whose output is ‘1’ if their logic type is AND type, OR type or BUFFER type and the selected drivers for the desired high leaky logic gate after feeding the MLV for the whole combinational circuitry of the circuit. All the gates of this gated group are constrained by 4.1.1 so that it makes easier in power planning and placement.

Decayed Net: The output net of a driver which belongs to above gated Group.

Partial Pullup: A NMOS transmission gate who gets VDD as input and whose output drives above Decayed Net.

In Figure 4.2.c the net ‘inv_out’ is a decayed Net. The Gated NAND gate and inverter are the elements of Gated group. Partial pullup is also shown in the same Figure 4.2.c.

4.1.1 Same level VDD gating to form Gated Groups

Here the same level is defined with respect to a given high leaky gate identified after feeding MLV to the combinational logic. The logic gates which are appeared as directly connected loads of a decayed net are the same level gates. In addition to chosen high leaky gate and drivers of decayed nets these same level gates are the only possible logic gates which comes for a gating group. But partial pullups can also be in the Gated group based on implementer preference. Due to this same level VDD gating constraint as already mentioned the all the gates within gated group can be placed within the same row with driver itself with a less placement constraint. Figure 4.3 shows how this Gating group is can be physically implemented.

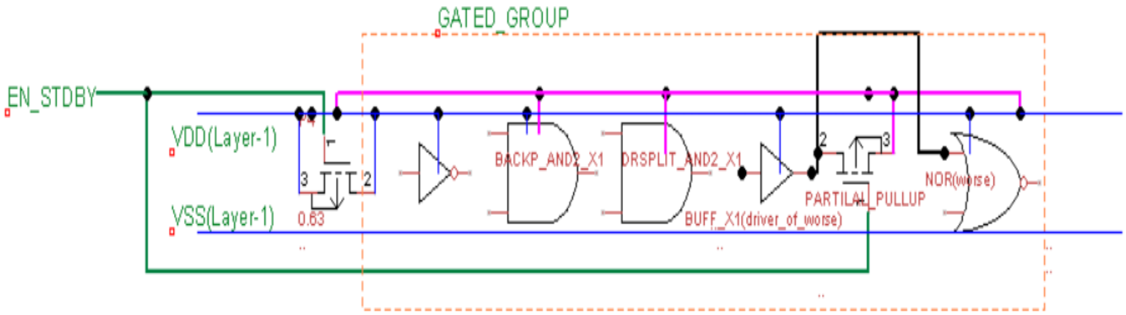


Figure 4.3 Physical placement of a Gated Group and partial pullups

4.1.2 Power reduction of immediate Fan-out gates of Decayed Nets exterior to Gated Groups

The same level gates which doesn’t belong to the gating group (gates in Figure 5.2) are having an additional advantage with their gate leakage reduction due to logic ‘1’ driver gating even with partially pulled up by partial pullups.

With this basic idea we modify the flops to feed MLV. With the same modification we achieve some flop power reduction also while preserving flop previous states so that system state can be retained. We augment the same feeding method so that it can Feed MLV while circuit is in scan mode redirecting scan chain through inverted flop outputs (QN) assuming the combinational logic is always fed with usual flop output Q.

4.2 Timing/Performance Considerations of Gated Groups

System performance is a major concern in VLSI design. After initializing the system based on user inputs the system outputs should be available as soon as some other component of the system wants to have it. So there is maximum delay defined for a timing path in between any two sequential elements/flops or in between selected input and output. This is defined as ‘Required time’ defined with respect to the system specifications. The ‘required time’ is constrained by the clock speed of the system and hence for any timing path in between two sequential elements the logic gate delays plays a major role which is why static timing analysis is done. So it is important to have some more time than all the required times for logic gates driving flop and receiving flop. This additional available time is known ‘positive slack’ which is desired. The Slack is defined as in equation 4.1.

Slack = [Clock cycle time – (hold time of sending flop + setup time of receiver + combinational logic delays in timing path)] (eq. 4.1)

Propagation delay of logic gates are the major contributor for combinational logic delay in a timing path.

4.2.1 propagation delay

A change of input logic levels for a logic gate causes to change the output logic level of a signal. Propagation delay of a logic gate is the time it takes to change the output level of the gate as input logic level changes. The transition of output signal can be low to high or high to low. As input changes if the gate output changes from low logic level to high logic level that is called low to high propagation delay ($t_{PLH)}$ .With a input transition if the gate output logic level changes from high to low logic level that is called high to low propagation delay (t_{PHL}).

The Figure 4.4 illustrates the t_{PLH} and t_{PHL} for a CMOS inverter. For all signal transitions, as logic level crosses 50% of the voltage which is for ‘logic high value’ is chosen as the reference point to measure propagation delays.

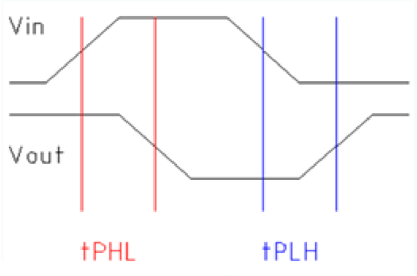


Figure 4.4 Hight to low (t_{PHL}) vs low to high propagation delays (tP_{LH})

So the average propagation delay(t_p) of the inverter is calculated as in Equation 4.2 [23].

$$\mathbf{t_p = \frac{1}{2}\left(t_{PHL} + t_{PLH}\right)} \quad \text{(eq. 4.2)}$$

Based on the electrical characteristics of the CMOS inverter, the low to high and high to low propagation delays of CMOS inverter are calculated with equation 4.3 and equation 4.4 respectively [23].

$$\mathbf{t_{PLH} \approx \frac{C_L V_{DD}}{\frac{W_p}{L_p}\mu_p C_{ox}\left(V_{DD} + V_{Tp}\right)^2}} \quad \text{(eq. 4.3)}$$

$$\mathbf{t_{PHL} \approx \frac{C_L V_{DD}}{\frac{W_n}{L_n}\mu_n C_{ox}\left(V_{DD} - V_{Tn}\right)^2}} \quad \text{(eq. 4.4)}$$

Where, VDD = Supply Voltage

Wp, Wn = widths of PMOS and NMOS transistors respectively

C_L = Load Capacitance

V_{Tp}, V_{Tn} = Threshold voltages of PMOS and NMOS respectively

Lp, Ln = Gate Lengths of PMOS and NMOS respectively

Cox = Gate capacitance

Power gating causes a voltage to drop across the gating Transistor even in the dynamic operation mode of the system due to the on resistance of gating transistor.

In our Research though we don’t do the power gating of whole module logic gates in the Gated group are subjected to supply voltage drop as many gates in the Gated group are changing output logic states from 0 to 1 causing charging the output capacitances of those gates wit drawing the current form supply. According to the equations 4.3 and 4.4 as the supply voltage decreases the propagation delays of the gates get increase causing a performance drop of the circuit. So if the circuit is expected to operate with a 10% performance drop when the supply voltage is 1.1v, when all the other factors are constant the maximum allowable voltage drop is 0.1v as for the logic gates through the gating transistor it provides 1.0v.

In our work as the gating transistor we use typical/normal threshold PMOS transistor with width 0.63um and length 0.005um which is equivalent to pull up transistor of a inverter in the 45nm nangate library. With that transistor we ensure to maintain a maximum 0.1v voltage drop so that the maximum performance drop is 10% which happen only when all the gates in a timing paths are belong to gating groups and each group possess it’s maximum allowable number of gates. But this situation happens very rarely because all the worst case gates are not appeared sequentially. Because of this performance drop is almost always lesser than 10%.

4.2.2 How maximum Gated group size is decided

The Gated group size is all the transistors which belong to a gating group except gating transistor and partial pullup transistors. We have added different types of logic gates to a group and as the loads of their outputs, added a separate set of gates which are not under gating group and under the non-gated supply. (Say this is setup -1) For a given moment the maximum current is drawn from the gated supply when all the outputs of the gated group transistors are charged to logic ‘1’. So with different kinds of such setups with the help of spice simulations we identified that we can have up to logic gates which are equivalent 75 inverters with drive strength ‘1’ in the sense of gate capacitance. Figure 4.5 shows the logic outputs of different kinds of gates in a Gated group as circuit is in dynamic operating mode and all the loads of the gates of gated groups are drowning currents from the power supply. Figure 4.6 shows the logic outputs of same gates when those are not in a Gated group We have included the related hspice program in the appendix.

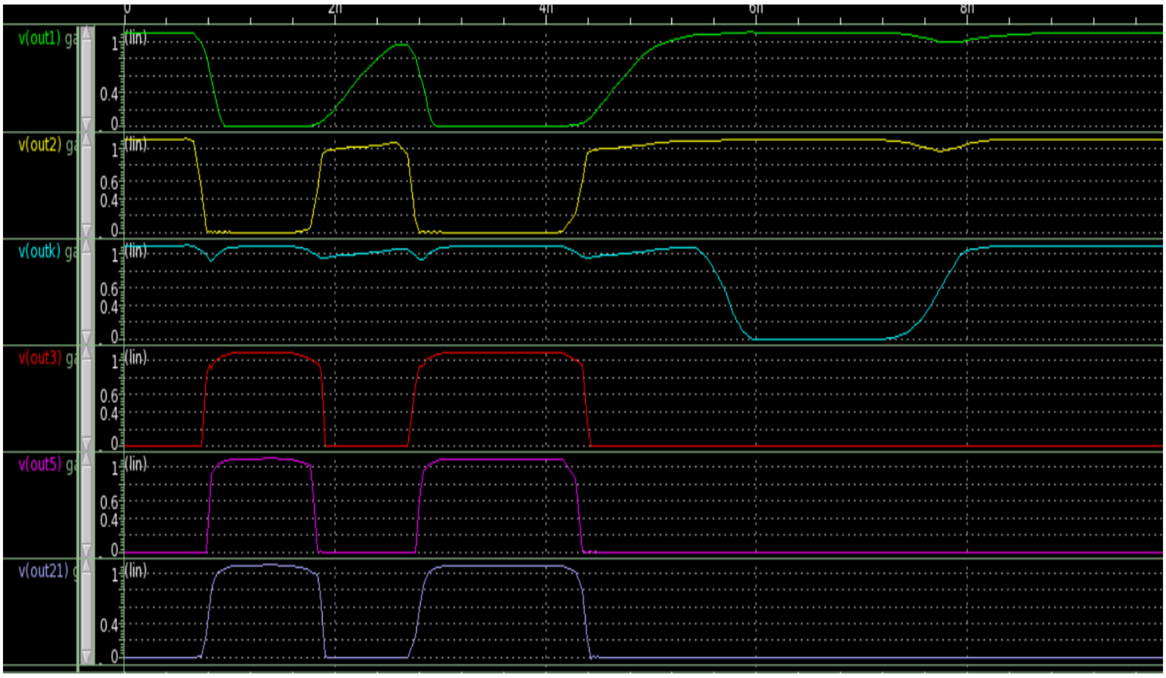


Figure 4.5 Outputs of Gated NOR3_X2, INV_X2, NAND2_X2, OR2_X2, AND2_X2, BUF_X2 in dynamic operation in when Gated Group has maximum possible capacitance

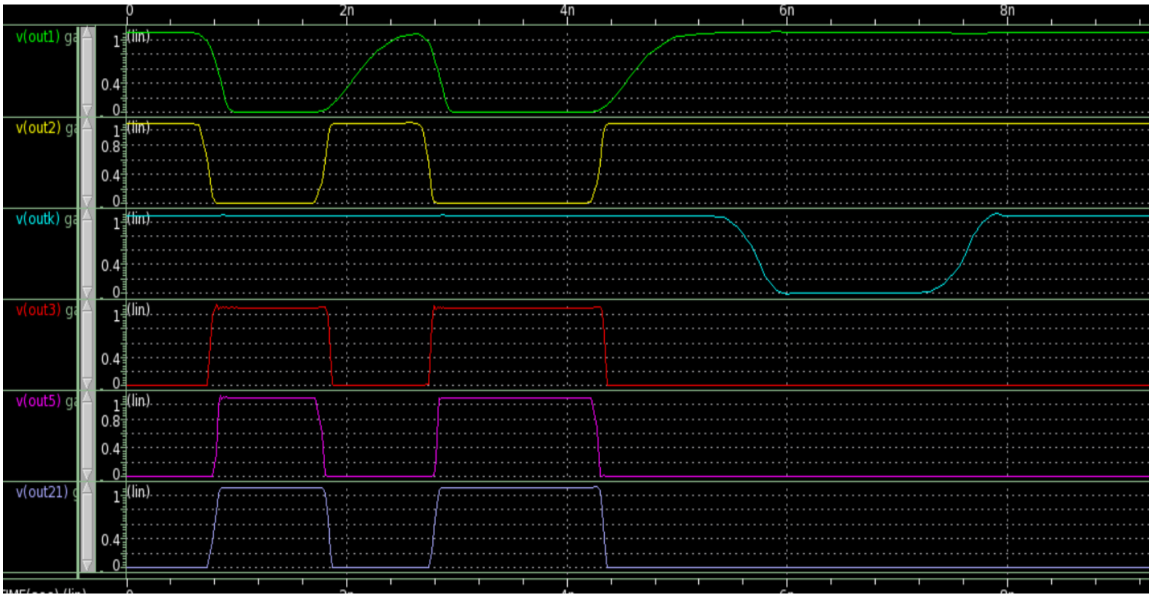


Figure 4.6 Outputs of NOR3_X2, INV_X2, NAND2_X2, OR2_X2, AND2_X2, BUF_X2 in while those not within a Gated Group

So According to equation 4.4, as vdd drops from 1.1v to 1.0v 28% propagation delay increment is experienced by PMOS when it has a threshold voltage of 0.471v. For a given timing path the worse case happens when all the gates belong to different Gated groups. So when design is synthesized, 0.28ns input delay is added so that it always maintain 0.28ns positive slack and hence timing constraints are preserved. Here in all simulations our system clock period was 2ns. Assuming half othe elements of the timing paths are in gated Groups we decided above 0.28ns value. Normally for a given timing path all the gates can't belong to Gated Groups. Hence this additional slack is always act as an upper bound.

4.3 Methodology

This section presents all the major steps carried out in through the Research. We have used the nangate 45nm open cell library which is developed based on 45nm BSIM4 model card for bulk CMOS in transistor behavior modelling. We have used five selected benchmarks from ISCAS89 benchmark suite to test our algorithms, circuit and gate level modifications with fast spice simulations using Synopsys XA. Table 4.3 shows some important statistics for chosen benchmarks. All the flops in every benchmark is connected to one scan chain.

Benchmark	Number of Inputs	Number of outputs	Number of D Flipflops	Number of Inverters/buffers	Number of (NOR + NAND + OR + AND) gates
S298	3	6	14	44	75
S953	16	23	29	84	311
S9234	36	39	211	3570	2027
S15850	77	150	534	6324	3448

Table 4.3 Important statistics for the four ISCAS89 benchmarks used throughout this Research

4.3.1 Synthesize the Benchmarks with Synopsys design compiler (DC)

NangateOpenCellLibrary_typical_ccs.lib standard open cell library was convertered to DC compatible ‘.db’ format (NangateOpenCellLibrary.db) by using the synopsys library compiler. To synthesize a selected benchmark, as the input for the design compiler we have given, the Verilog file of the selected benchmark, synopsys design constraint file (sdc) and NangateOpenCellLibrary.db. Some of the used input constraints are Clock period 2.5ns (400MHz clock speed), clock Latency 0.1ns and minimum IO delay 0.2ns. We have set the operating temperature as 25^0C and operating voltage is set to 0.95V as decided according to the explanation in section 4.2.

Then the benchmarks are synthesized so that in the synthesized netlist it contains only the Flip flops and the 6 basic combinational logic gate types; That is INVERTER, NAND, NOR, BUFFER, AND,OR gates. Those are the only elements from which original benchmarks are comprised with. Figure 4.7 shows the Synthesized netlist of s298 benchmark. The synthesized netlist also can be dumped into a text file of Verilog format. (say this file is s298Synthesized.v)

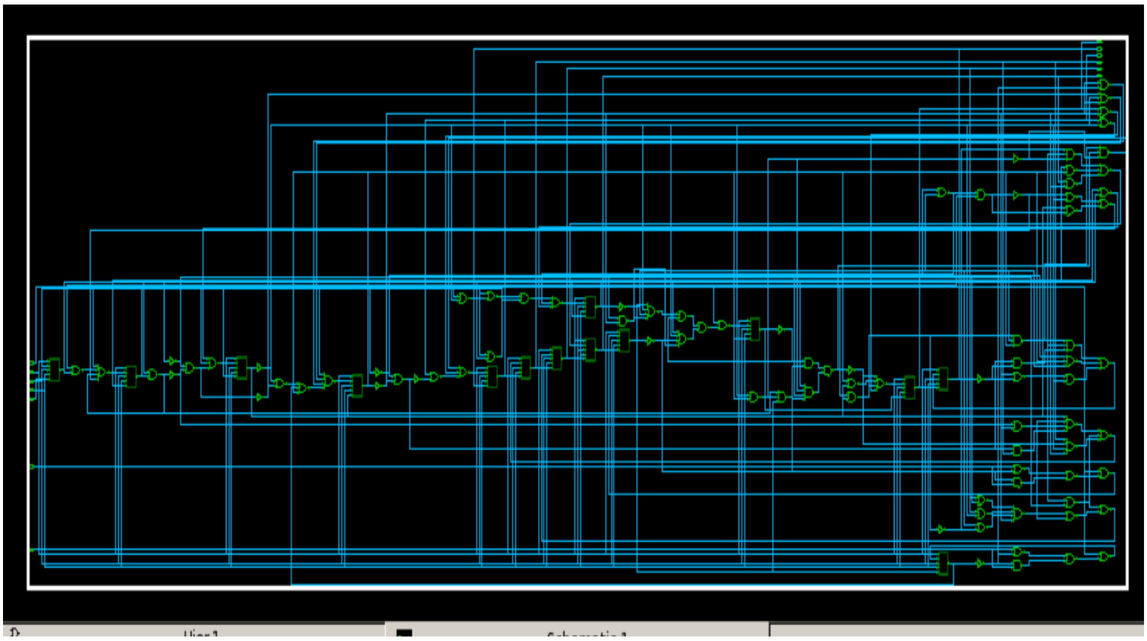


Figure 4.7 DC compiler synthesized Netlist of s298 benchmark

4.3.2 Find the minimum leakage vector from randomly generated 100 vectors

- Added power related information to the synthesized netlist using its dumped Verilog file. For this We have used a shell script which is included the Appendix
- Convert the DC synthesized netlist with augmented power information to hspice Using ‘nettran’ command available with synopsys Hercules tool.

  To convert the s298synthesized.v in 5.3.1 the following command is used.

  “Nettran -verilog synthesizeds298.v -cdl ../libs/spice/nangateCustom.cdl -outType spice -outName s298subthres.sp -verilog-b0 VSS -verilog-b1 VDD”

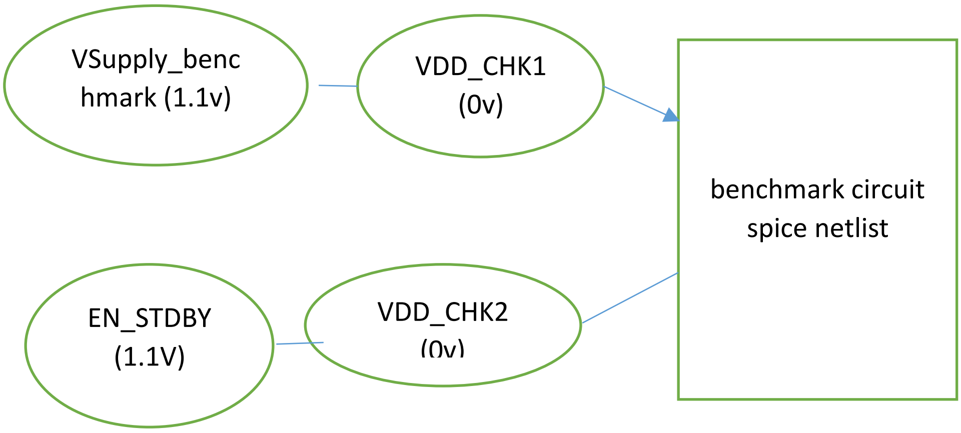
  In above nangateCustom.cdl contains the hspice netlists for basic logic gates and Flipflops which are used to synthesize the benchmark.

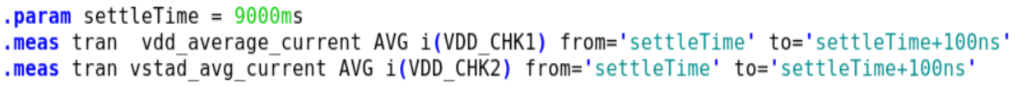
- Generate 100random vectors so that each vector contains number of digits equal to [number of registers + number of inputs] of the desired benchmark

  For example, if it s298 each vector should contain 17 digits as it has 14 flops and 3 inputs.

- Spice simulation to measure leakage current for each randomly generated vector

  To initialize the flip-flops with relevant digits from the input vector we use the scan chain. The scan shift is carried out with slower clock (200MHz) compared to normal operating clock (400MHz). To assign relevant MLV logic values for input ports we use a vector file (.vec file format). A sample file for s298 design is included in the Appendix. To do fast spice simulations we use synopsys XA tool. And for each test vector we measure the current after about 8000ms from the last dynamic operation of the circuit; That is when the circuit is idling. And this is the leakage current of the circuit for that input vector. Figure 4.7 shows the basic concept using zero voltages sources and leaving sufficient settle time to measure leakage current with a spice simulation.



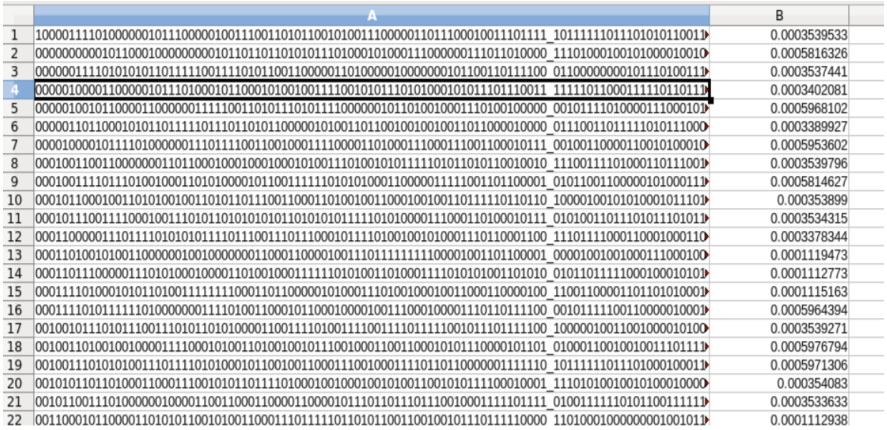


```
.param settleTime = 9000ms
.meas tran  vdd_average_current AVG i(VDD_CHK1) from='settleTime' to='settleTime+100ns'
.meas tran vstad_avg_current AVG i(VDD_CHK2) from='settleTime' to='settleTime+100ns'
```

Figure 4.8 Setup to find leakage current with spice simulation

So, Leakage power = 1.1 * (vdd_avg_current + vstd_avg_current)

- Likewise, we do the fast spice simulations for all 100 vectors and measure the leakage current values. Among them Chose the random vector which gives the least leakage current across the zero voltage source in Figure 4.8. And this is considered as the MLV for the benchmark. Figure 4.9 shows collected leakage current details for some randomly generated vectors for s15850 benchmark. The MLV is in the first row of that spreadsheet.



	A	B
1	10000111101000000101110000010011100110101100101001110000011011100010011101111_10111111011101010110011	0.0003539533
2	0000000000101100010000000000101101101010111010001010001110000001110110100 0_11101000100101000010010	0.0005816326
3	0000001111010101011011111001111010110011000001101000001000000001011001101111 00_01100000000101110100111	0.0003537441
4	00000100001100000101110100010110001010010011110010101110101000101011101110011_11111011000111110110111	0.0003402081
5	00000100101100001100000011111001101011101011110000000101101001000111010010000_00101111010000111000101	0.0005968102
6	00000110110001010110111110111011010110000010100110110010010011011000010000_01110011011111010111000	0.0003389927
7	00001000010111101000000111011110011001000111100001101000111000111001100010111_00100110000110010100010	0.0005953602
8	000100110011000000001101100010001000100010100111010010101111101011010110010010_11100111101000110111001	0.0003539796
9	00010011110111010010001101010000101100111111010101000110000011111001101100001_01011001100000010100011 1	0.0005814627
10	00010110001001101010010011010110111001100011010010011000100100110111110110110_10000100101010001011101	0.000353899
11	00010111001111000100111010110101010101101010101111101010000111000110100010111_01010011011101011101011	0.0003534315
12	00011000001110111101010101111011100111011100010111101001001010001110110001100_11101111000110001000110	0.0003378344
13	000110100101001100000001001000000011000110000100111011111111100001001101100001_00001001001000111000100	0.0001119473
14	0001101110000011101010001000011010010001111110101001101010011010001111010101001101010_01011011111000100010101	0.0001112773
15	00011110100010101101001111111100011011000001010001110100100010011000110000100_11001100001101101010001	0.0001115163
16	00011110101111110100000001111010011000101100010000100111000100001110110111100_00101111100110000010001	0.0005964394
17	00100101110101110011101011010100001100111101001111001111011111001011101111100_10000010011001000010100	0.0003539271
18	00100110100100100001111000101001101001001011100100011001100010101110000101101_01000110010010011101111	0.0005976794
19	00100111010101001110111101010001011001001100011100100011110110110000001111110_10111111011101000100011	0.0005971306
20	00101011011010001100011100101011011110100010010001001010011001010111100010001_11101010010100010000	0.000354083
21	00101100111010000001000011001100011000011000001011101101110111001000111110111 1_01001111101011001111 1	0.0003533633
22	00110001011000011010101100101001100011101111101101011001100100101110111110000_11010001000000001001011	0.0001112938

Figure 4.9 Leakage Currents for different inputs vectors for s15850 benchmark

4.3.3 Form Gated Groups with found MLV by using Synopsys Design Compiler

Feed the MLV which was found in above 4.3.2 to the synthesized netlist using ‘set_case_anaysis’ command of the design compiler. Identify gates with high leakage power dissipation based on the input vector of those gates once the MLV is propagated through the combinational logic. For a given gate if all the inputs are fed as logic ‘1’ that gate is identified as a high leaky gate. At the same time for the gates which have four inputs, if at least three of those inputs are feeding logic ‘1’ Then that gate is also considered as a high leaky gate. But for any kind of high leaky gate the maximum decayed input net amount is 2. Figure 4.10 shows high leaky NAND gates with their input vectors achieved after MLV propagation.

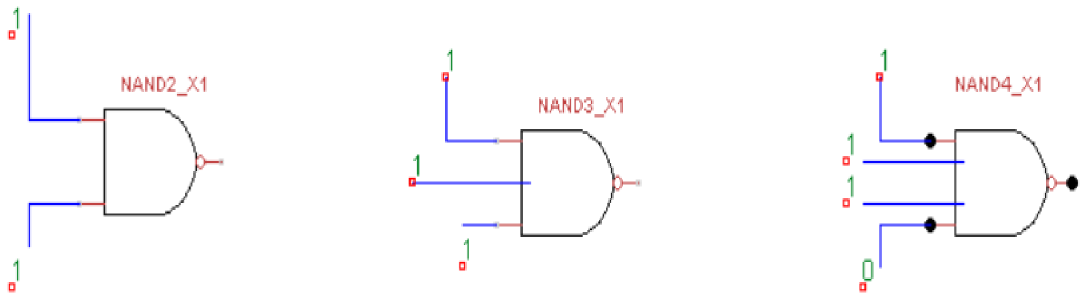
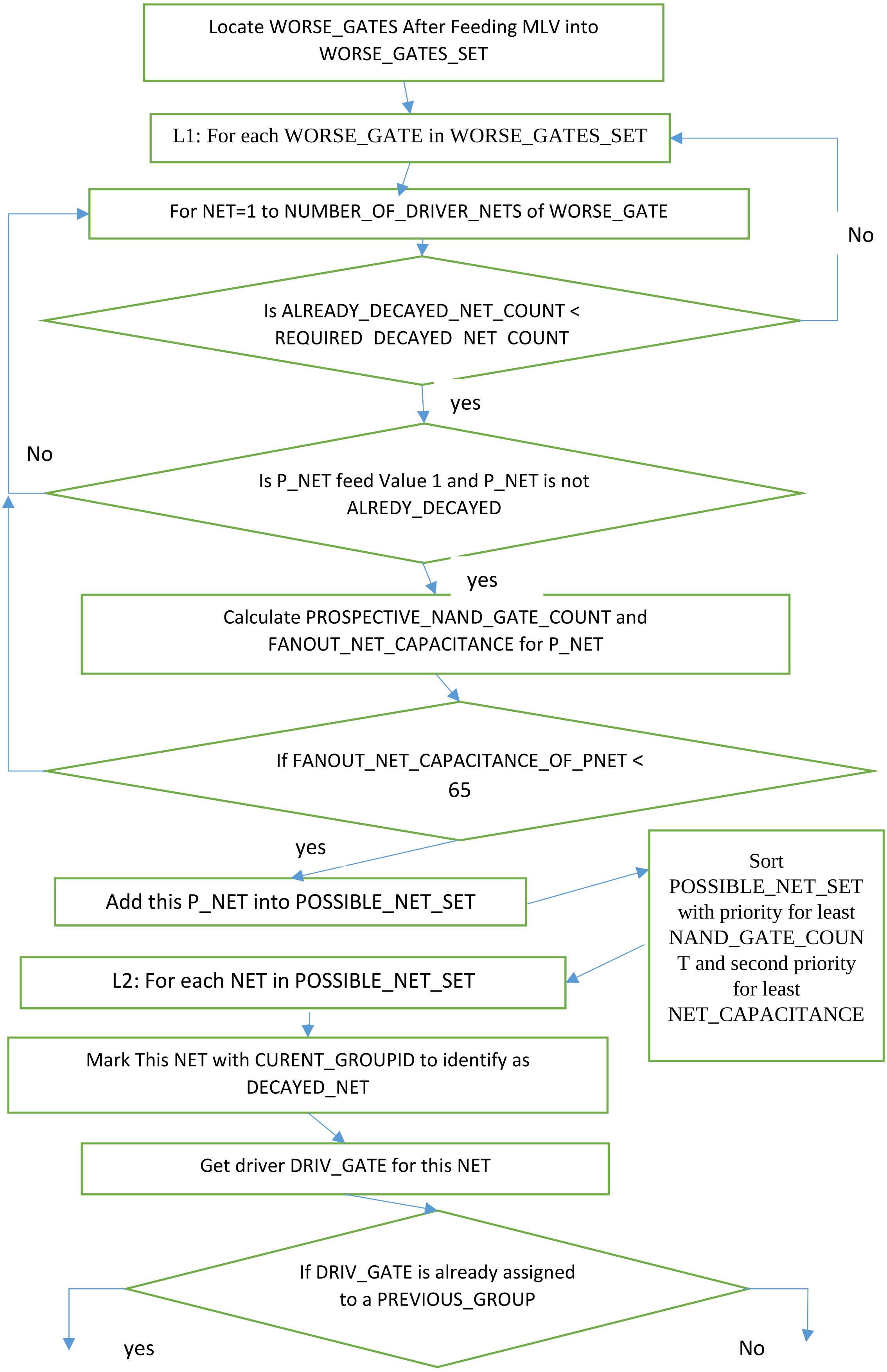
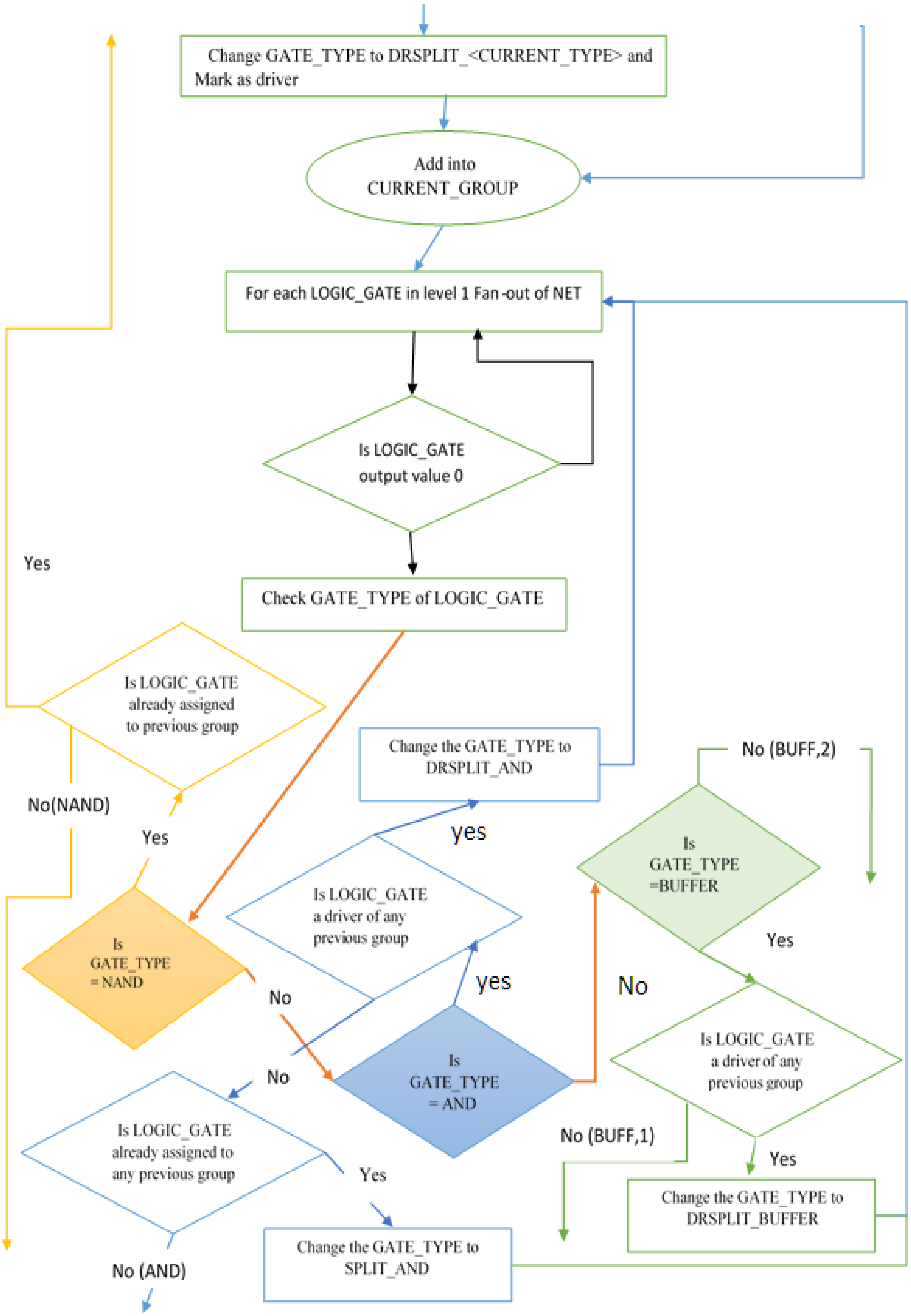
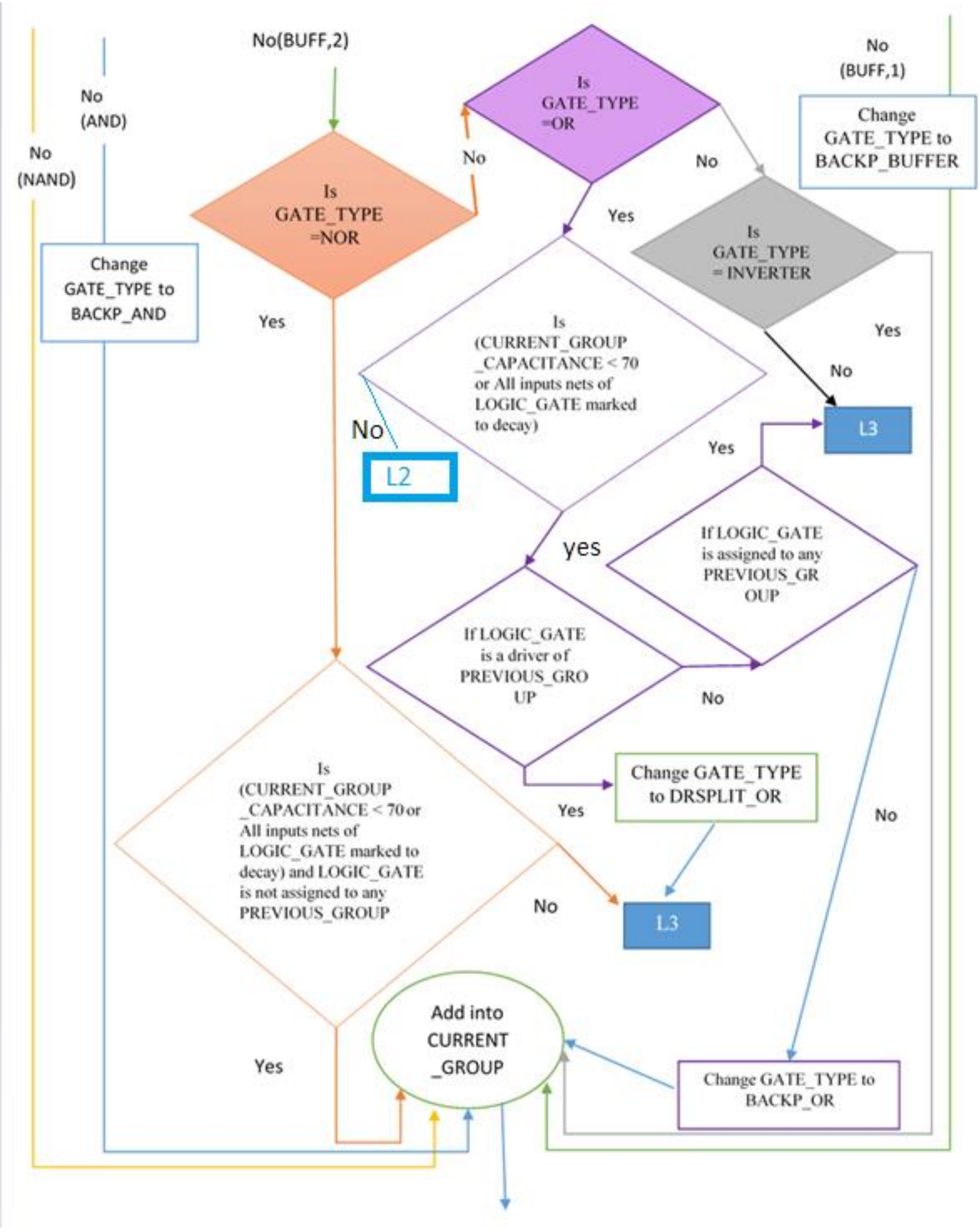


Figure 4.10 – Possible high leaky NAND gates after MLV propagation through combo logic

Any high leaky gate is referred as WORSE_GATE in algorithm explained in Figure 4.11 which describes the algorithm used to form Gated Groups which is run on top of DC synthesized netlist. The algorithm is implemented in tcl language so that it can be executed in Synopsys Design Compiler. The algorithm is executed around each high leaky gate identified with MLV propagation. The time complexity of the algorithm is O(n) where n is the number of gates in the design. (Considering the step of collecting high leaky gates from all the gates in the design). Since the time complexity is linear to number of gates this is easily solvable algorithm. This is a Greedy algorithm because the high leaky gates which identified earlier are subjected to have better leakage reduction treatments (For example can decay 2 inputs nets) rather than the high leaky gates identified later. However as already mentioned in


Locate WORSE_GATES After Feeding MLV into WORSE_GATES_SET
L1: For each WORSE_GATE in WORSE_GATES_SET
For NET=1 to NUMBER_OF_DRIVER_NETS of WORSE_GATE
Is ALREADY_DECAYED_NET_COUNT < REQUIRED DECAYED NET COUNT
No
yes
Is P_NET feed Value 1 and P_NET is not ALREDY_DECAYED
No
yes
Calculate PROSPECTIVE_NAND_GATE_COUNT and FANOUT_NET_CAPACITANCE for P_NET
If FANOUT_NET_CAPACITANCE_OF_PNET < 65
yes
Add this P_NET into POSSIBLE_NET_SET
Sort POSSIBLE_NET_SET with priority for least NAND_GATE_COUNT and second priority for least NET_CAPACITANCE
L2: For each NET in POSSIBLE_NET_SET
Mark This NET with CURENT_GROUPID to identify as DECAYED_NET
Get driver DRIV_GATE for this NET
If DRIV_GATE is already assigned to a PREVIOUS_GROUP
yes
No


Change GATE_TYPE to DRSPLIT_<CURRENT_TYPE> and Mark as driver
Add into CURRENT_GROUP
For each LOGIC_GATE in level 1 Fan-out of NET
Is LOGIC_GATE output value 0
Check GATE_TYPE of LOGIC_GATE
Yes
Is LOGIC_GATE already assigned to previous group
No(NAND)
Yes
Is GATE_TYPE = NAND
Change the GATE_TYPE to DRSPLIT_AND
yes
Is LOGIC_GATE a driver of any previous group
yes
No
No
Is GATE_TYPE = AND
No
No (BUFF,2)
Is GATE_TYPE =BUFFER
Yes
Is LOGIC_GATE a driver of any previous group
No (BUFF,1)
Yes
Change the GATE_TYPE to DRSPLIT_BUFFER
Is LOGIC_GATE already assigned to any previous group
Yes
Change the GATE_TYPE to SPLIT_AND
No (AND)


No(BUFF,2)
No
(AND)
No
(NAND)
Is
GATE_TYPE
=OR
No
(BUFF,1)
Change
GATE_TYPE to
BACKP_BUFFER
No
No
Is
GATE_TYPE
=NOR
Yes
Is
GATE_TYPE
= INVERTER
Change
GATE_TYPE to
BACKP_AND
Yes
Yes
Is
(CURRENT_GROUP
_CAPACITANCE < 70
or All inputs nets of
LOGIC_GATE marked
to decay)
No
No
L2
L3
Yes
If LOGIC_GATE
is assigned to any
PREVIOUS_GR
OUP
yes
If LOGIC_GATE
is a driver of
PREVIOUS_GRO
UP
No
Change GATE_TYPE
to DRSPLIT_OR
Yes
No
Is
(CURRENT_GROUP
_CAPACITANCE < 70 or
All inputs nets of
LOGIC_GATE marked to
decay) and LOGIC_GATE
is not assigned to any
PREVIOUS_GROUP
No
L3
Add into
CURRENT
_GROUP
Yes
Change GATE_TYPE to
BACKP_OR

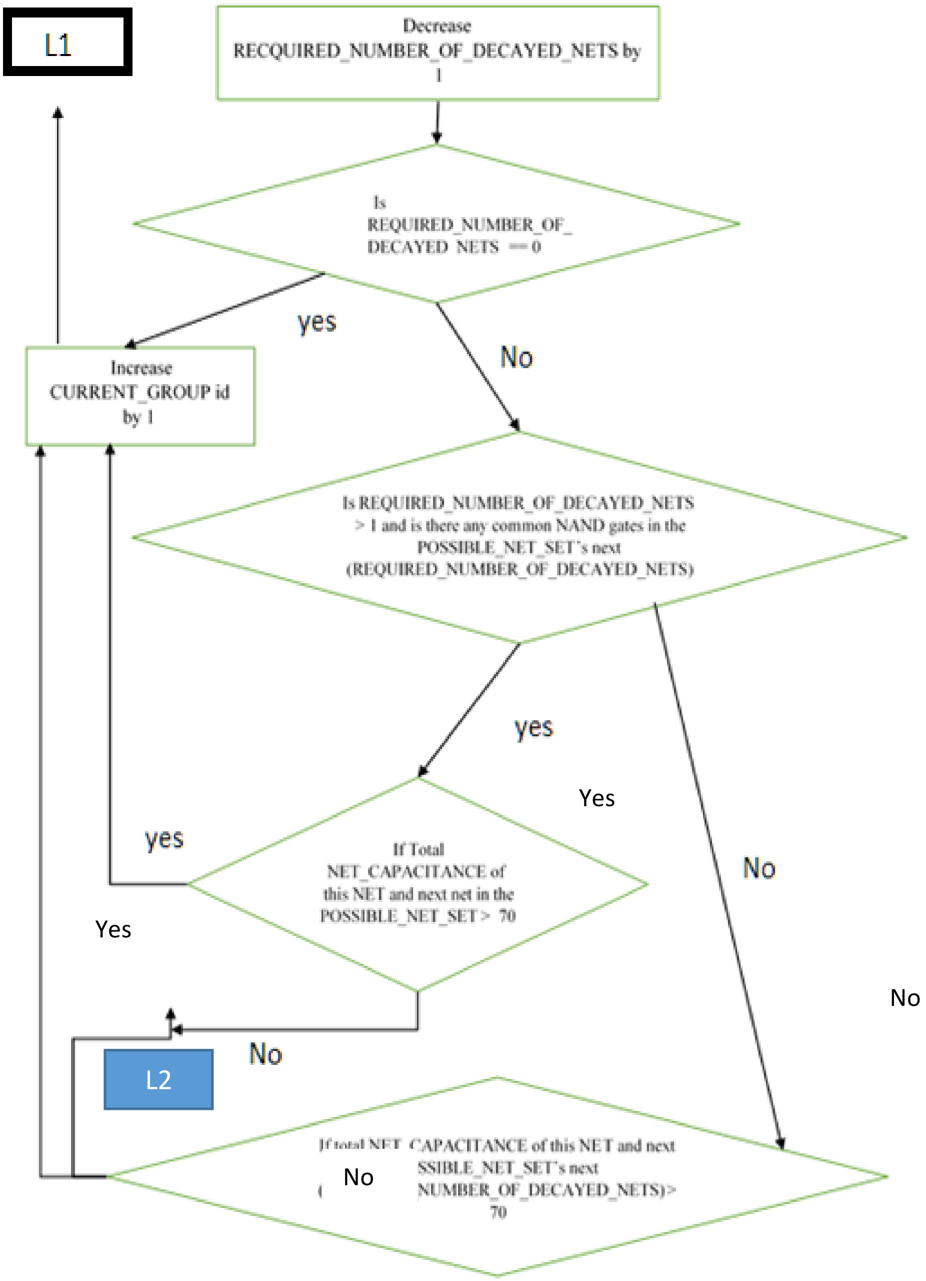

L1
Decrease RECQUIRED_NUMBER_OF_DECAYED_NETS by 1
Is REQUIRED_NUMBER_OF_ DECAYED_NETS == 0
yes
No
Increase CURRENT_GROUP id by 1
Is REQUIRED_NUMBER_OF_DECAYED_NETS > 1 and is there any common NAND gates in the POSSIBLE_NET_SET's next (REQUIRED_NUMBER_OF_DECAYED_NETS)
yes
Yes
No
yes
If Total NET_CAPACITANCE of this NET and next net in the POSSIBLE_NET_SET> 70
Yes
No
No
L2
If total NET_CAPACITANCE of this NET and next
SSIBLE_NET_SET's next
(No NUMBER_OF_DECAYED_NETS)>
70

Figure 4. 11 Algorithm to Find Gated groups in a synthesized design

section 4.1, the VDD gating is done with PMOS which is having 0.63um width and 0.005 um gate length with typical/normal threshold voltage (typical Vth). Partial pullups are NMOS transistors with 0.415um width and 0.05um length with high Vth.

Figure 4.12 explains couple of steps by running the algorithm expressed in Figure 4.11 which is used to form Gated Groups.

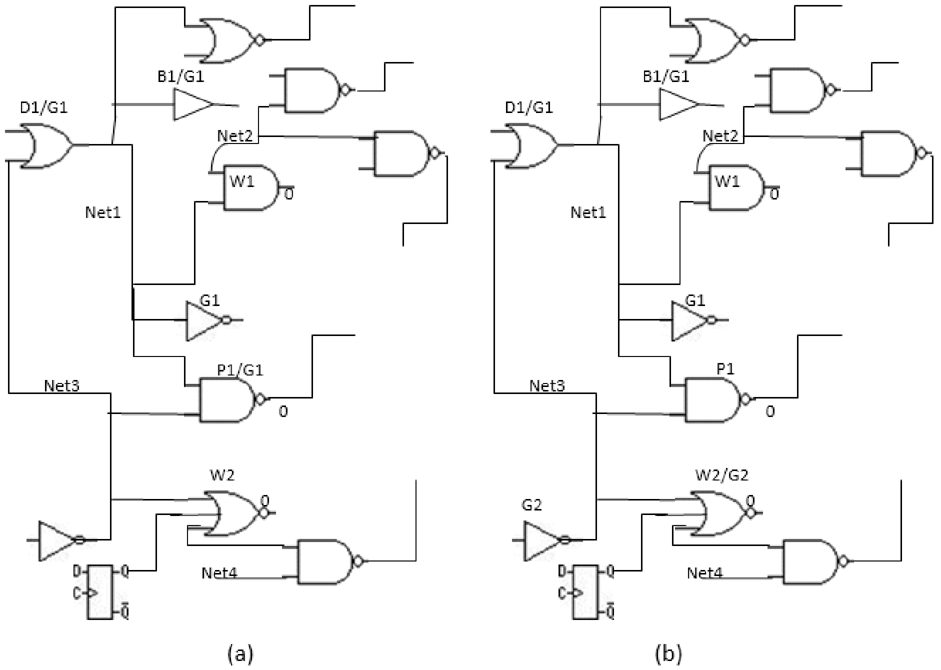


Figure 4.12 Processing two worse leaky gates with algorithm in 4.10

In 4.12.a, Algorithm locates worse leak gate w1. From it's two driving nets Net1 and Net2, Net1 is selected as decayed Net whose driver is D1. Gate B1 is marked for backup power. So all the gates which have label G1 is grouped into a same group called G1 with respect worse leaky gate w1.

In 4.12.b, Algorithm executes after Following step in 4.12.a and algorithm locates worse leaky gate 'w2'. The driving nets directly connected to flop outputs are ignored as flops are not included in Gated groups. So candidates nets to decay are Net3 and Net4. Among them Net 3 is selected as it has least NAND count from both candidates. As gate D1 is a driver of previous group G1, D1 is marked as a split driver. P1 is marked as splitter NAND and is removed from G1 so that it stands without any group. The new group form is G2 whose members are having label G2.

4.3.4 Required Gate modifications for grouping

Consider an AND gate which gets both inputs as ‘1’. In this case, this AND is identified as a worse leak gate and hence it should be included into the Gated Group. AS we focus on VDD gating for all Gated groups, to maintain the output logic value of this AND gate (which is ‘1’) we can feed the non-gated/original power supply to the inverter which is within the AND gate. So this AND gate is modified as BACKP_AND. Figure 4.13 shows such a modification for library cell AND2_X1. As already mentioned in algorithm the name for this modified gate is BACKP_AND2_X1. As shown in Figure 4.3, parallel to the power Gated Group’s Gated supply rail Now we have the permanent supply rail also. Hence this task can be easily attained.

Consider a scenario where 3 input NAND gate is getting two input nets from two drivers coming from 2 separate groups. Since we don’t know which driver is power gated first (which is based on the placement information of the buffers in the EN_STDBY signal which signals the power gating of Gated groups) we can’t put this into any of those group or any other group. In this scenario a safe self-gating based on its output value is suggested as in Figure 4.14. this new modified AND cell is named as SPLIT_AND; In the Figure since we have modified AND2_X1 library cell, this is named as SPLIT_AND2_X1. Once the original AND gate replace as SPLIT_AND2_X1 cell, that cell doesn’t want to belong to any of the Gated as its own gating transistor do the safe power gating without losing output value. But if this is going to be act as a driver for some other Gated Group this gate should be a member of that group as output net is required to be decayed. Though internal structure is same for both cases to differentiate two scenarios we use a different name called DRSPLIT_AND2_X1 for the latter case.

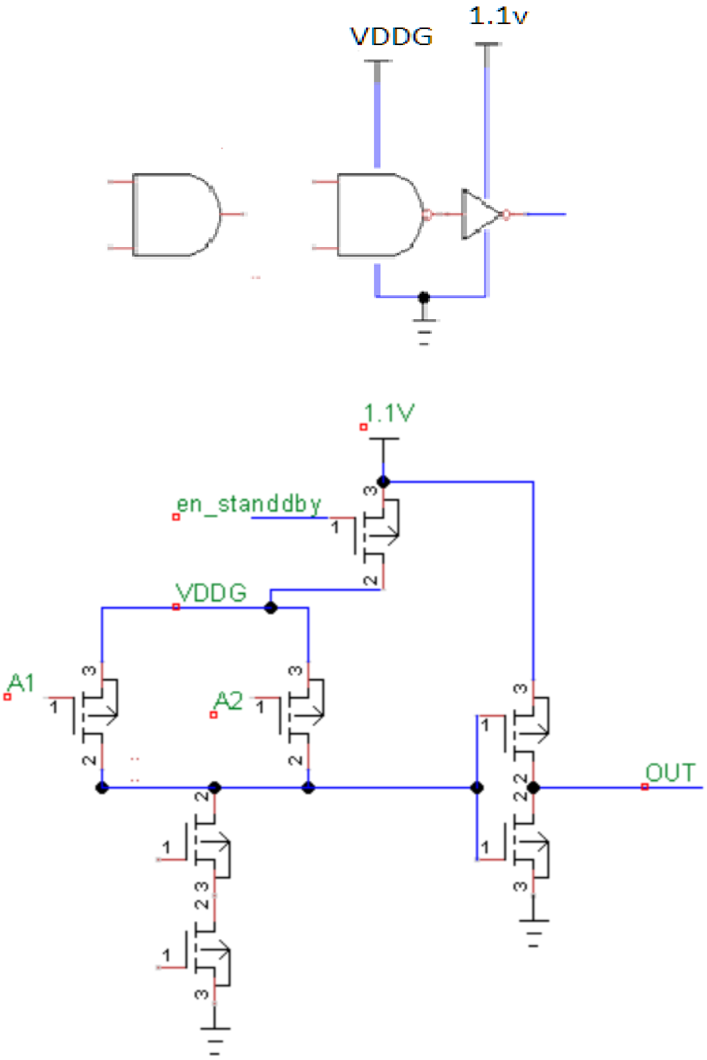
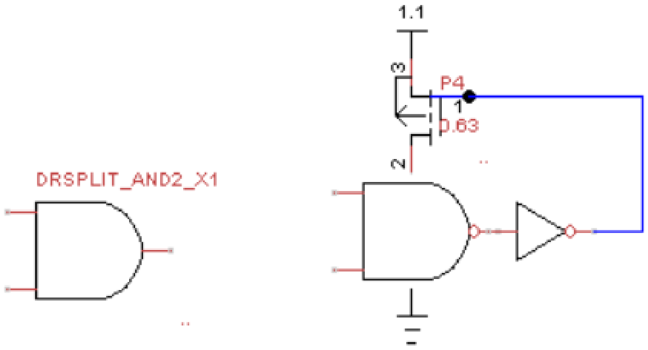


Figure 4.13 BACKP_AND2_X1



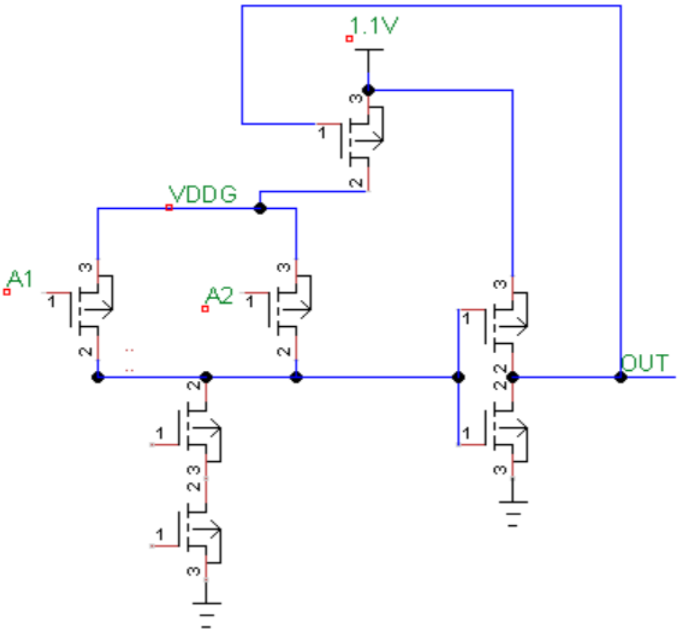


Figure 4.14 DRSPLIT_AND2_X1(or SPLIT_AND2_X1)

4.3.5 Simulate the netlist with Gated Groups using synopsys XA and obtain the leakage current

The MLV is shifted with scan chain and same scan enable signal is used as the control signal for gating PMOS transistors and partial pullups.

4.3.6 Slightly modify the algorithm so that the groups are form without driver grouping and measure the leakage power which contributes to subthreshold leakage reduction only.

4.3.7 Feeding the input vector by modified low power, state retained flops

We have used two types of flops from nangate 45nm library to synthesize the selected four ISCAS89 benchmarks. Those two types are SDFFR_X1 and SDFFR_X2. both have active low reset. And both of those scan Flops such that scan mode is activated when Scan enable signal is high. The latter one has more drive strength than first one.

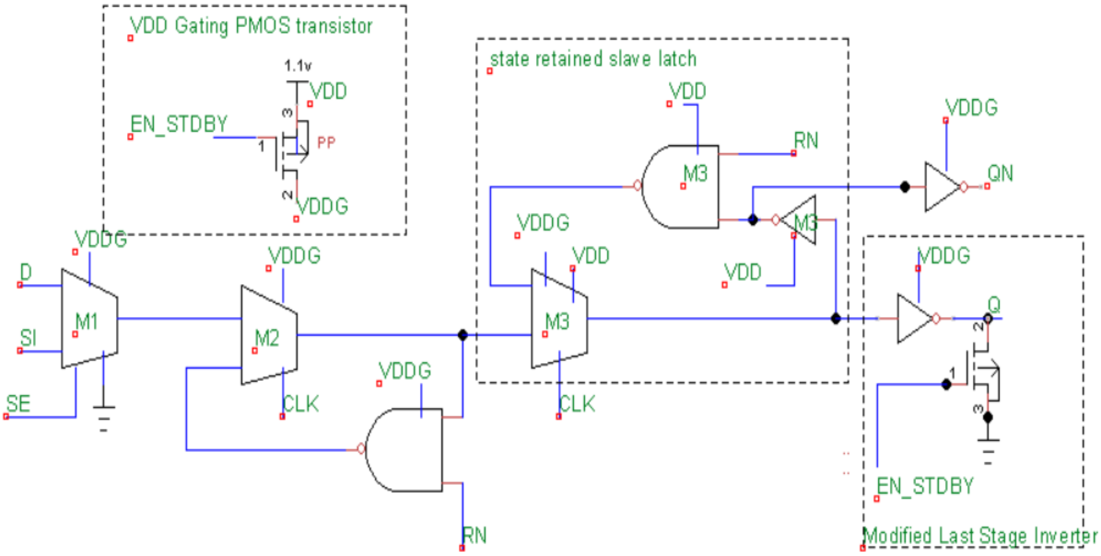


Figure 4.15 ZEROSBY_SDFFR_X1 Modified Low power Flop to feed value 0 MLV in standby mode while data is retained after standby mode

To feed the MLV, we have modified inverter which drives the output Q; That is, in the standby mode we disconnect the power with and feed the desired MLV at the flop outputs by using PMOS pullups or NMOS pull down. If we want to maintain the logic value zero as a portion of MLV at the flop output, then we do the VDD gating of the output inverter of the flop with PMOS 0.63um width and 0.05um length and output is pulled down to zero. In the Figure 4.15 the we have shown the modified last stage inverter to feed logic value ‘0’ as a part of MLV in standby mode. The logic value ‘1’ can be fed in standby mode as shown in Figure 16. The control signal EN_STDBY is asserted when system is in standby mode. We have use the same gating transistor to disconnect the power from logic gates which don’t contribute to latched value when clock is kept at logic value zero without switching. Hence We can attain some power reduction of these flops in addition to feeding the MLV values. Meanwhile since we haven’t done any power gating on the slave latch, we can retain the stored data before standby mode even after low power featured standby mode. The circuit shown in Figure 4.17 is used to stop the clocking and keep the clock at value zero as far as system is in standby mode.

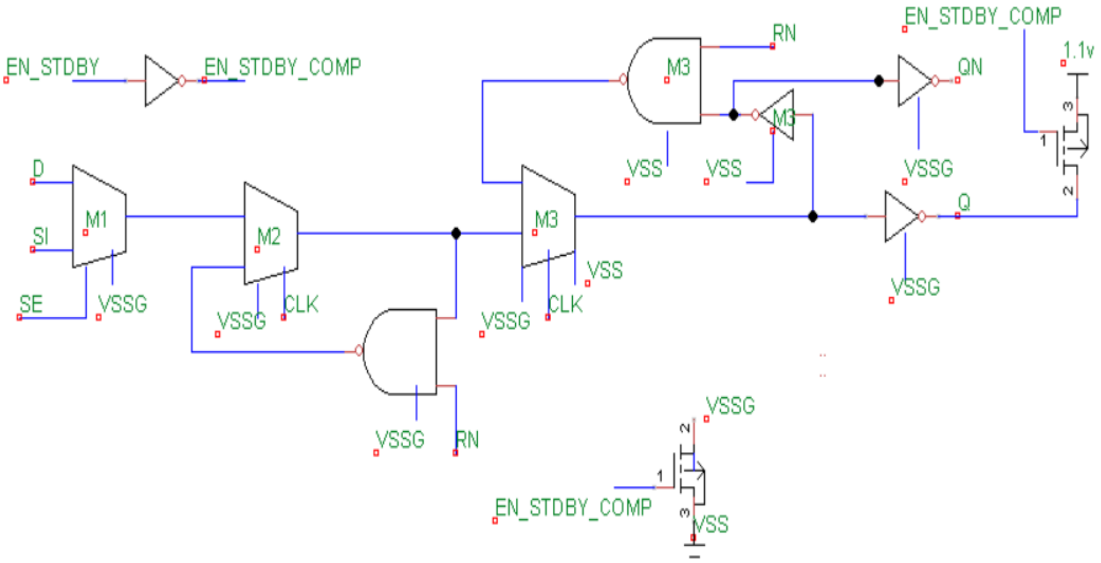


Figure 4.16 ONESBY_SDFFR_X1 Modified Low power state retain Flip flop which feed value 1 for MLV in standby mode

Figure 4.16 shows the Modified low power state retained Flip flop to feed logic value 1 for MLV in standby mode.

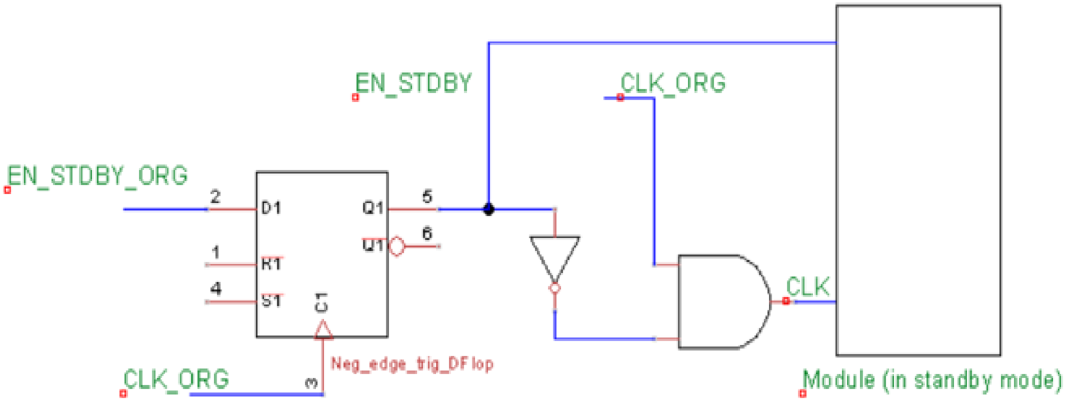


Figure 4.17 Clock Gating in Standby mode

4.3.8 Input Modification to feed MLV in Standby mode

Figure 4.18 shows the required input port modifications to feed MLV. 4.18.a shows the feeding logic value ‘0’ in standby mode and 4.18.b shows the feeding logic value ‘1’ in standby mode. EN_STDBY is the control signal which is asserted in standby mode.

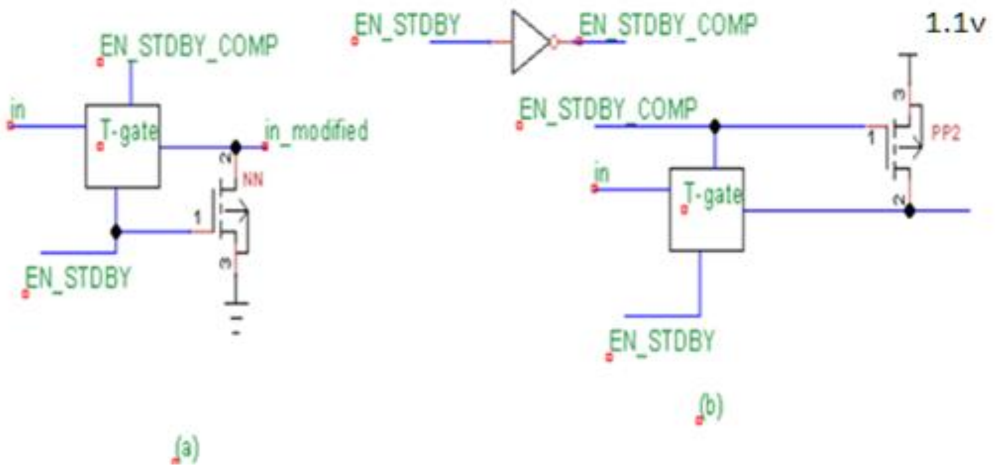


Figure 4.18 Modified Input Ports to feed MLV

4.3.9 Feeding MLV in scan mode

We can extend the MLV value feeding methods presented in 4.3.8 and 4.3.7 so that in the scan mode the combinational circuit is driven by the MLV while the scan chain is built through the inverter outputs (QN) of the scan flip flops. Here We have a constraint; that QN is not used as an input for the combinational logic circuit. And with this method the scan data should be input as like this. That is if we number the scan chain flops giving the number '1' to first flop in scan chain (which get the user data sequence directly to shift) all the scan data that is for odd numbered flops should be inverted while scan data for even numbered flops should be kept as it is. For example, consider the required scan data sequence to initialize eight flop scan chain is '01011001'. Then with this new mechanism we should feed '11110011'.

The modified last stage inverters of flops should be augmented as in Figure 4.19.a to feed value 1 and 4.19.b to feed value 0. The modified inputs should be augmented as in Figure 4.19.c to feed value 1 and as in 4.19.d to feed value 0. Dimensions of all the gating transistors and pullups/pulldowns used are such that for NMOS 0.415um width and 0.05um length and for PMOS 0.63um width and 0.05 length. In the Figure 4.19, control signal to enable Scan mode is SE.

4.4 Test Results

4.4.1 Analysis of Modified flops

Table 4.4, 4.5 and 4.6 show the output waveform of Modified Flops compared to the original flops in the library which we have used to synthesize the chosen benchmarks. The original Flops are SDFFR_X1 and SDFFR_X2 while the modified flops are ZEROSBY_SDFFR_X1, ZEROSBY_SDFFR_X2, ONESBY_SDFFR_X1, ONESBY _SDFFR_X2. But throughout the dynamic operation of the flop (As clocking data propagation across the flop) the average gated Voltage of VDD port with

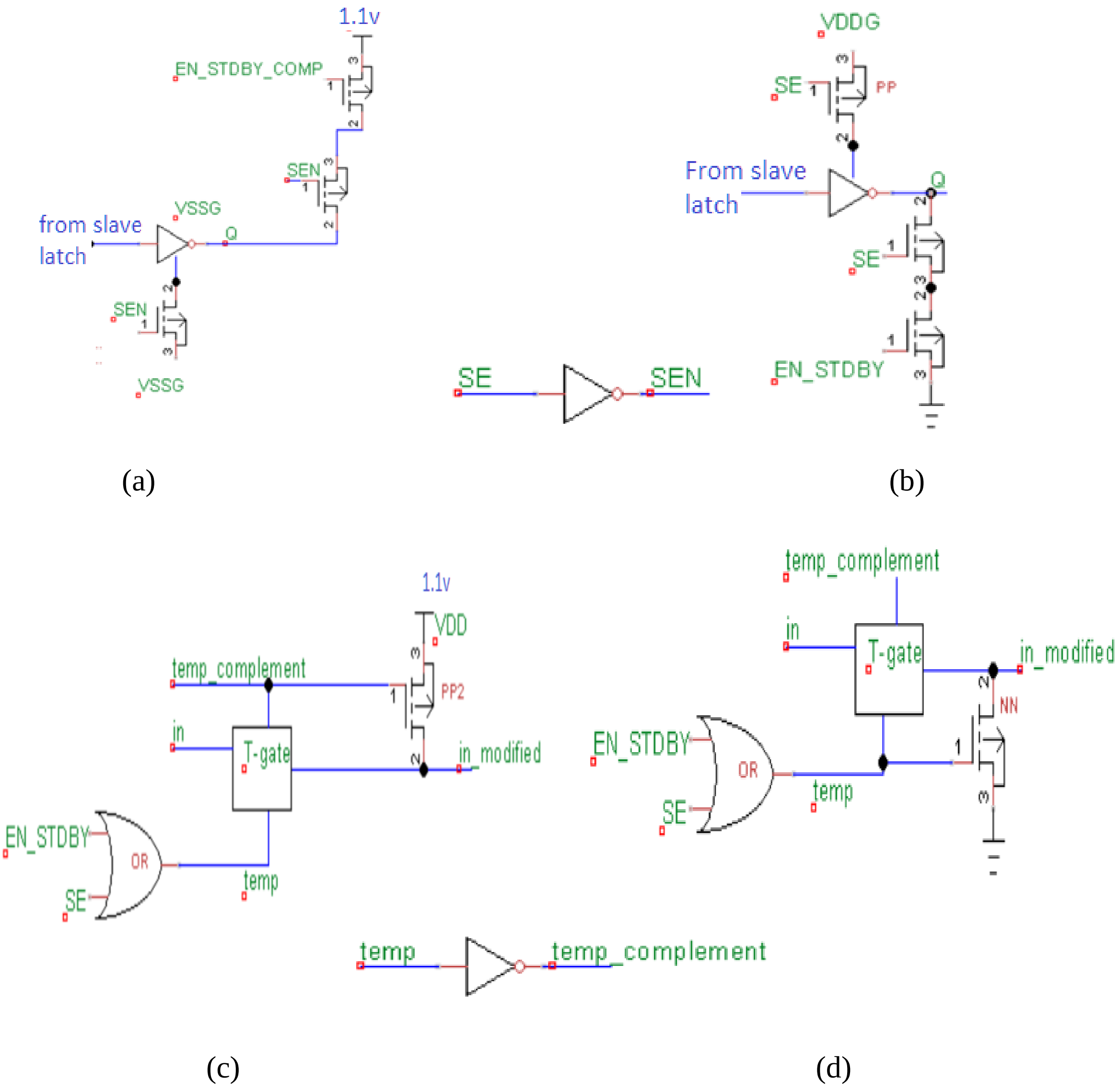


Figure 4.19 Feeding MLV in Scan Mode

VDD gating is 1.05v and with VSS gating the average voltage of VSS port when dynamic operation is 0.03v. The propagation delay increment due to gating is about 0.03ns. Table 5.3.1,5.3.2 and 5.3.3 also contain the details for propagation delay and maximum voltage dropped level in dynamic operation of the original flops and modified flops. Due to gating the maximum voltage level drop is 0.05v. Since We have synthesized the benchmarks for 0.95v supply voltage this 0.05v drop doesn't affect to system's timing requirements. According to the waveforms we can clearly see throughout the dynamic operation of the flop (As clocking data propagation across the flop) the average gated Voltage when VDD gating is almost 1.1v and average gated Voltage with VSS gating is almost 0v.

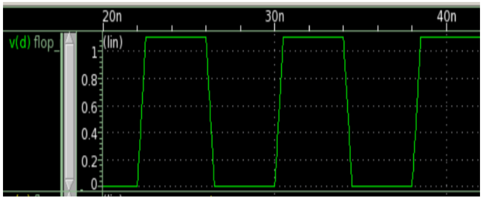
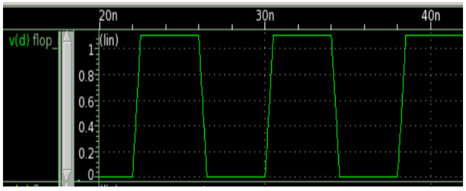
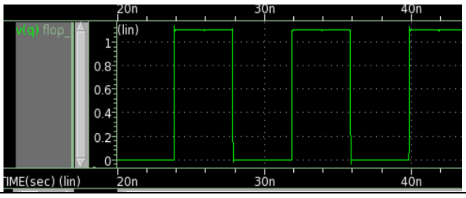
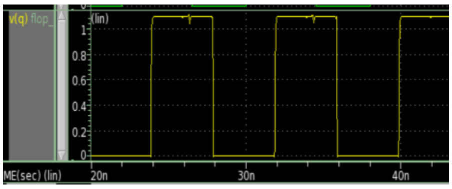
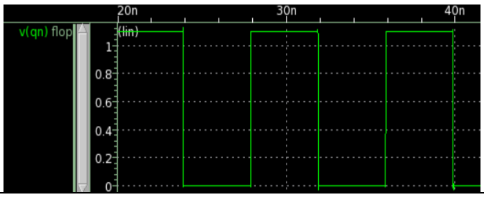
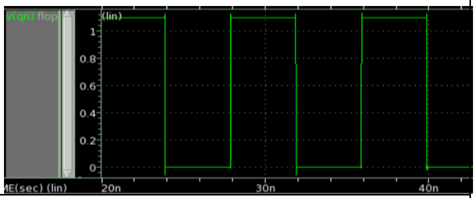
	SDFFR_X1	ZEROSBY_SDFFR_X1
Input Signal		
Normal mode output		
Scan mode output		
VDD Lower bound in functional/ dynamic mode	1.099v	1.05v
Maximum Propagation delay (1propagation/ 0 propagation)	1.498e-09 / 1.499e-09	1.525e-09 /1.513e-09

Table 4.4 Comparison of modified flop ZEROSBY_SDFFR_X1 with SDFFR_X1

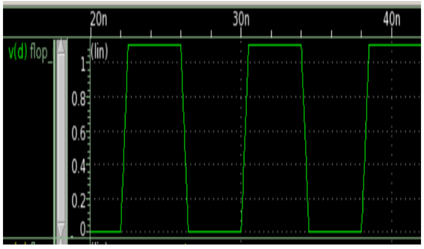
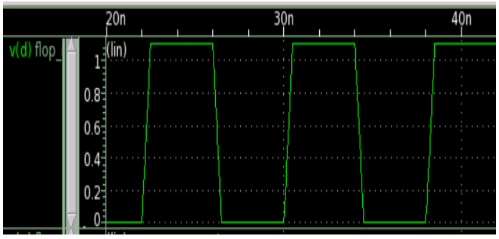
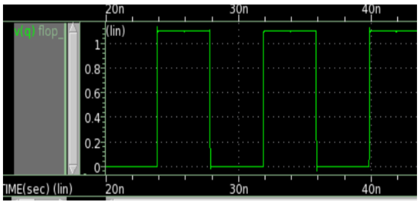
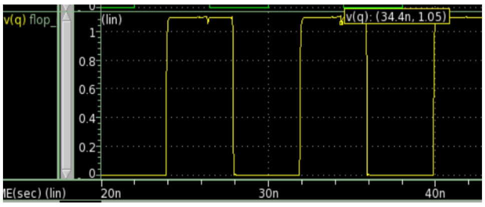
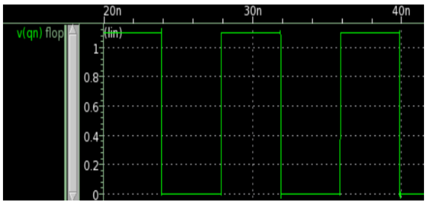
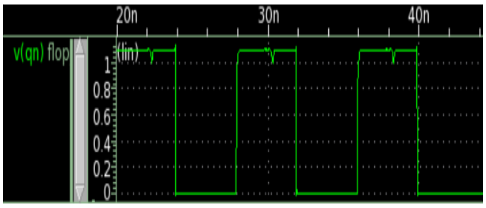
	SDFFR_X2	ZEROSBY_SDFFR_X2
Input Signal		
Normal mode output		
Scan mode output		
Supply Voltage Lower bound in functional/ dynamic mode	1.09	1.05v (in normal mode) 1.043v (in scan mode at QN)
Maximum Propagation delay (1propagation/ 0 propagation)	1.509e-09 1.530e-09	1.538e-09 1.544e-09

Table 4.5 Comparison of modified flop ZEROSBY_SDFFR_X2 with SDFFR_X2

Table 4.7 present the leakage current details of modified flops and their original Flops in standby mode. per those details modified flops shows a reduction in leakage current in standby mode compared to original flops. The leakage currents are measured when the latched value of the Flops are logic ‘1’.

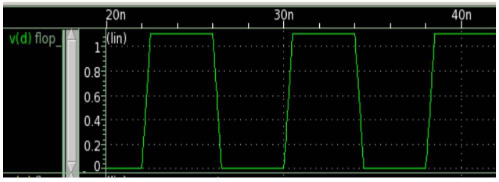
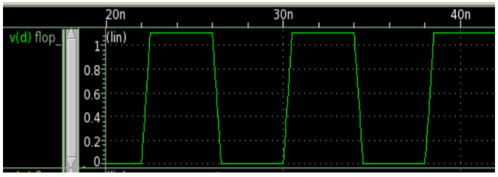
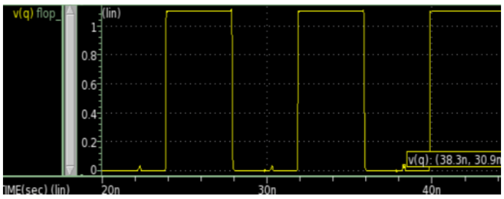
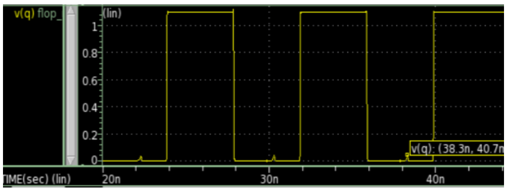
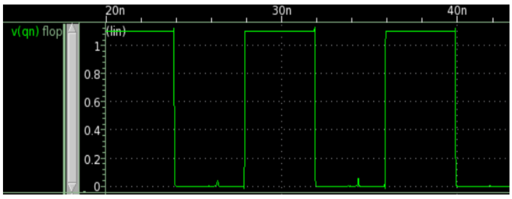
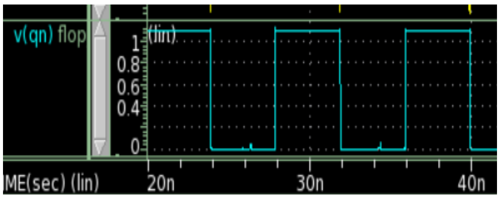
	ONESBY_SDFFR_X2	ONESBY_SDFFR_X1
		
Normal mode output		
Scan mode output		
Supply Voltage Lower bound in functional / dynamic mode and average voltage	0.031v	0.030v
Maximum Propagation delay (1propagation/ 0 propagation)	1.519e-09 / 1.551e-09	1.506e-09 /1.514e-09

Table 4.6 analysis details for ONESBY_SDFFR_X1 and ONESBY_SDFFR_X2

Flop	Leakage current (nA) (when latched value is 1)
SDFFR_X1	74.22
SDFFR_X2	108.9
ZEROSBY_SDFFR_X1	64.28
ZEROSBY_SDFFR_X2	85.83
ONESBY_SDFFR_X1	58.11
ONESBY_SDFFR_X2	75.67

Table 4.7 Leakage current details of original flops and modified flops in standby mode

4.4.2 Standby Mode Leakage power

Table 4.8 Summarizes the measured leakage current of four chosen benchmarks. Leakage current only with MLV illustrates leakage current when MLV is fed in standby mode. "MLV + subthreshold leakage reduction" means measure leakage current after forming Gated Groups so that decayed nets and partial pullups are not available in standby mode and hence no driver gating also. In this case the MLV is fed to the combination circuit using the scan shifting and. vec file. (MLV + subthreshold reduction + gate leakage reduction) refers to the situation when Gated groups are formed so that drivers of the high leaky logic gates are also included within them. And hence decayed nets and partial pullups are also present. (MLV + subthreshold + gate leak + flop power reduction) refers the situation where we use new modified flops to feed the MLV.

According to the Table 4.8, each of the newly tried mechanism shows some kind of more leakage reduction compared to initial MLV based leakage current. Table 4.9 shows percentage leakage power reduction of each newly implemented mechanism with respect to chosen MLV based leakage power.

Benchmark	Leakage current only with MLV	Leakage current (MLV + subthreshold leakage reduction)	Leakage current (MLV + subthreshold reduction + gate leakage reduction)	Leakage Current (MLV + subthreshold + gate leak + flop power reduction)
S298	2.8872uA	2.75617uA	2.527uA	2.0204651uA
S953	8.214uA	7.6379174uA	5.977116 uA (6.4064270uA)	4.5214280uA
S9234	46.758361uA	43.542278uA	36.501772uA	28.353283uA
S15850	109.9119uA	107.38596uA	92.694047uA	69.03755uA

Table 4.8 Standby Leakage Current details of benchmarks when subjected to different leakage reduction mechanisms

Benchmark	% reduction in (MLV + subthreshold leakage reduction) mechanism	% reduction in (MLV + subthreshold reduction + gate leakage reduction) mechanism	% reduction in (MLV + subthreshold + gate leak + flop power reduction) mechanism
S298	4.5	12.4	30.02
S953	7.02	27.23	44.95
S9234	6.87	21.93	39.36
S15850	2.29	15.67	37.18

Table 4.9 Percentage power reduction of each newly introduced mechanism compared to MLV power reduction

4.4.3 Area increment

Since this is transistor level simulation the rough estimate for the Area of a logic gate is obtained as number of INV2_X1 equivalents belongs to that logic gate. So the Area of INV2_X1 is considered as ‘1’. With that the Area of NAND2_X1 is 2. For inverters and buffers the number of INV_X1 equivalents are equal to drive strength. For NAND and NOR gates the number of equivalents are calculated according to Equation 4.5. For AND gates and OR gates which drive strength less than for equation 4.6 is used and When drive strength is 4 for AND or OR gates the equation 4.7 is used. For the flops total number of transistors are divided by 2 to get INV_X1 equivalents.

Number of INV2_X1 equivalents of gate G when G Is NOR or NAND gate = [number of inputs of G x drive strength of G] (eq. 4.5)

Number of INV2_X1 equivalents of gate G when G is and AND or OR gate and drive strength less than 4 = [number of inputs of G] + [driv strength of G] (eq. 4.6)

Number of INV2_X1 equivalents of gate G as G is and AND or OR gate and drive strength equal to 4 = [number of inputs of G * 2] + [driv strength of G] (eq. 4.7)

Table 4.10 Summarizes the area increment as area estimate is obtained as mentioned above.

Benchmark	Original area	Extra Area Added		Area increment %	
		For Gated Groups only	for Gated Groups + Flop modifications	Based on Added extra area for Gated Groups only	Based on Total added area including flop modifications
S298	533	4	32	0.71	6.003
S953	1294	55	113	4.25	8.03
S9234	7609	258	681	3.39	8.21
S15850	17778	496	1566	2.78	8.03

Table 4.10 Percentage Area increment due to extra circuitries for standby power reduction

So the extra added Area percentage is always less than 10. And significant area addition is shown due to the flop modifications to feed MLV. Equivalent percentage or area increment is possible even we use the blocking logic to feed MLV. If we consider extra area percentage added due to Gated Groups, it is always less than 5%.

4.4.4 Settle time/Wakeup time of Gated groups in post Standby powerup

In the Standby period the Gated Groups are VDD power gated. At the end of the standby period the EN_STANDBY signal is de-assert to power up the Gated groups. These groups are expected to settle at the their initial supply voltage which was existed before standby power gating. In Table 5.9 the settle time refers to the time it takes by Gated group to come to the initial voltage after power is up again. The measurement is taken as the time interval as falling EN_STANDBY signal crosses the 0.3v and gated supply voltage crosses 1.0 in rising. To get this measurement we have considered the Gated Group with largest number of gates in each benchmark. So we can see settle time is around 0.1ns which is a desired value. In addition, we have measured average Gated voltages of the same Gated group in normal operation mode as well as standby mode. Those details are also included in Table 4.11.

Benchmark	Average gated voltage in normal operational mode	Average gated Voltage in standby mode	Wake up time after partial gating
s298	1.090	0.499	0.247ns
S953	1.0995v	0.51	0.165ns (0.1 1.0)
S9234	1.0995v	0.51	0.153ns (0.3 1.0)
S15850	1.099	0.53	0.1615(0.3 1.0)

Table 4.11 Settle time and Average Gated voltages of Gated groups with maximum capacity in each benchmark

4.4.5 Data retention Verification of Modified Flops

The Modified Flops ZEROSBY_SDFFR_X1, ZEROSBY_SDFFR_X2, ONESBY_ SDFFR_X1, ONESBY_ SDFFR_X2 are facilitated with state retention while operating in low power in standby mode. We have verified state retention of modified flops with s298 benchmark. In Figure 4.20 we have shown the state retention activity for first seven flip flops identified with the scan chain. Initially data is shifted using scan chain so that in standby mode, each flop feeds the opposite logic value as expected by MLV. So, as standby signal desserts the previous states of the flips flops are restored.

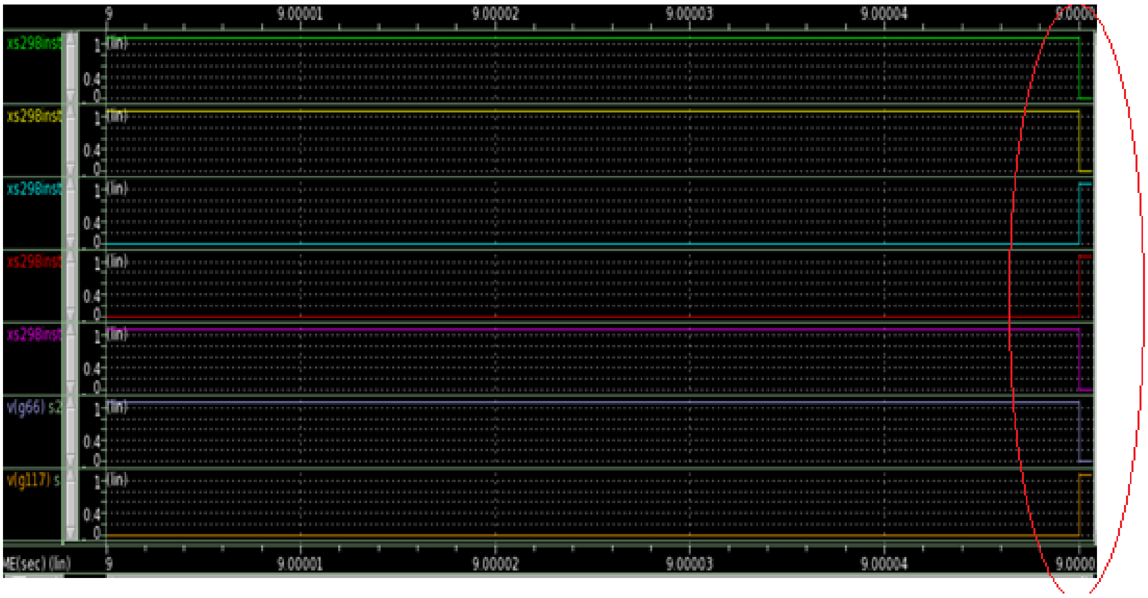


Figure 4.20 State Retention of modified flops to feed MLV and to have low leakage

4.4.6 Verification of successful Scan Mode operation

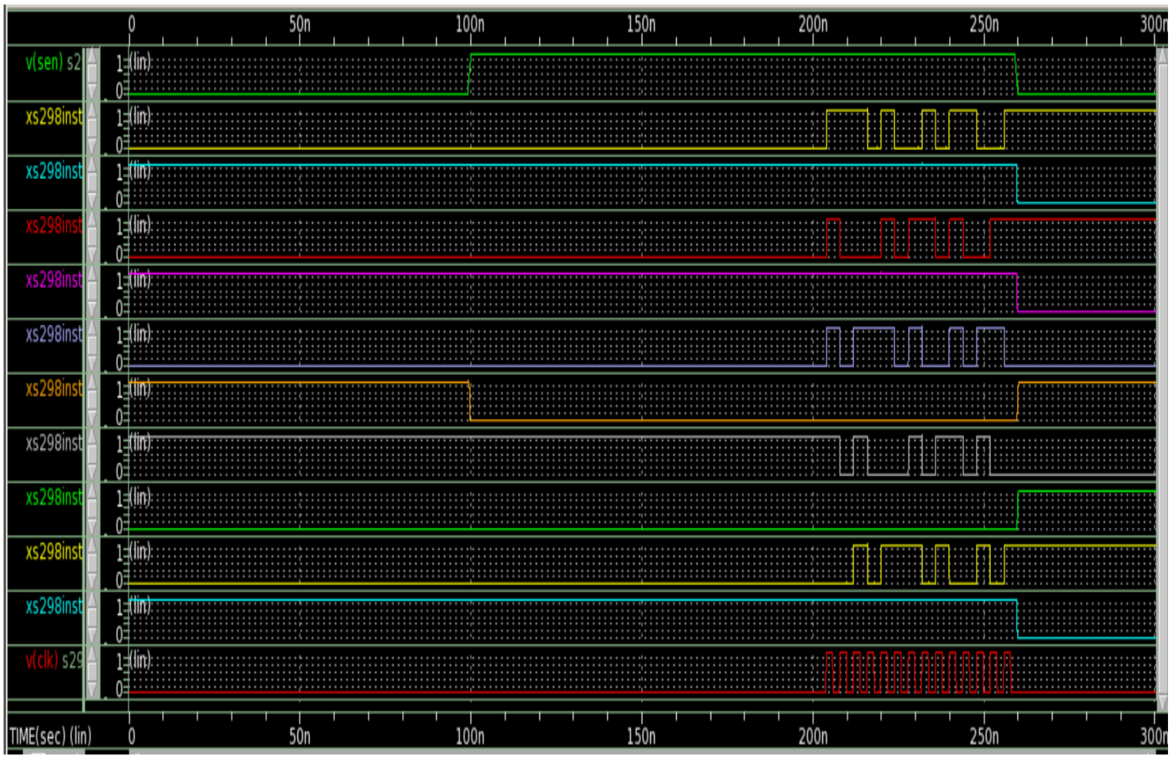


Figure 4.21 Scan Mode Operation of Modified Flops

We have further modified the Flip flops so that the MLV can be fed in the scan mode also. Here in scan mode no Gated Groups are develops and no power gating. Figure 4.20 shows the output changes of first five flops in chain in scan mode. According to the figure we can see the scan data is shifted through scan chain while the output of the flops are kept at their output values. In Figure 4.21 First graph is the time vs scan enable signal. The last one is the clock. Starting from second graph those are QN vs time and Q vs time graph pairs one for each flop. So we can see at the end of scan shifting and scan enable dessert and clock stops, the shifted values are stored in the Flops.

5. CONCLUSION AND FUTURE WORK

With the rapid scaling of silicon technology from generation to generation, the MOSFET transistor sizes are getting reduced. Although reduced transistor sizes allows higher integration density and higher operating frequency there exists a major drawback. That is simultaneous switching of millions of transistors dissipates huge power while overheating the chip reducing the reliability. In addition to power dissipation due to switching, since many modules of the SOC devices like mobile phones are idling most of the time, millions of transistors together contribute for a high standby leakage power dissipation which has become the most significant power dissipation mechanism in sub-micron and nanometer technology based CMOS designs.

From all leakage power reduction mechanism power gating is considered as a very effective mechanism as it reduces the leakage currents in the power gated module drastically. But larger wakeup times, challenges in sleep transistor designs issues with data retention are some of the noticeable drawbacks of the power gating technique.

Based on the input values of logic gates the leakage current experienced by an logic gate differs. So if we can feed the input logic levels which gives the minimum leakage when the logic gates are idling we can achieve a considerable leakage power reduction. But identifying a leakage vector for large combinational circuit is challengeable as all the individual gates in the circuits might not be at their minimum leakage levels due to a common input vector applied for the whole circuit. Hence after applying a MLV for a large combinational circuit, for the identified worse leakage gates some separate treatments are done; Gate replacements, Gate Modifications or adding new gates to feed desired input values for the worse case gates such that overall power reduction around worse leaky gates is increased.

In this thesis we focus on standby leakage power reduction. The basic idea behind this is, use of power gating around worse leaky gates which are identified after feeding MLV to the combinational logic circuit. and feed the MLV through flip flops which are enhance for low power while data is preserved.

So We developed an algorithm to form Small Gated groups around identified worse leaky gates. And for the VDD gating we use a PMOS transistor which is obtain from a inverter of drive strength 1. Instead using high Vth PMOS, we use Typical Vth PMOS. To form a Gated group we chose the immediate fanout gates of the driving net of a worse leaky gate. And we do the gating so that the output values of those gates will not be changed and hence they don't badly affect the gates which are currently with a low leakage power dissipation. In addition we do the power gating of the selected drivers of the worse leak gate due to which it causes to reduce gate leakages(in addition to subthreshold leakage) of the immediate fanouts of the driver. When we do the driver gating the output logic levels of the loads which are in Gated group might be changed if those gates are driving high capacitive load. To avoid this, we use we use high Vth partial pullups.

We have modified flip flops in the design to feed the MLV in standby mode so that with the same modification, the flip flops dissipates less power in standby mode but still the stored data is retained once come back to standby mode. Finally we extend these modifications to feed MLV in the scan mode while using inverted output(QN) of the flops for scan shifting while MLV is fed to the combinational logic through the output of the flop(Q).

We have tested our algorithm and gate and flop modifications with transistor level simulations with four selected benchmarks of ISCAS89 benchmark suite. Compared to initial MLV found by set of random vectors, we were able to achieve more than 30% leakage power reduction with 8% or less area increment. And the wakeup time after standby mode of each benchmark is less than 0.25ns. We have verified the successful operation of scan mode and data retention after these modifications with s298 benchmark.

To calculate leakage current values we use synopsys XA which is a spice based transistor level simulation tool. And in this research, We use the nangate 45nm open cell library and modifications are done in spice netlists.

As Future work we can suggest couple of things as follows.

- Input pin Reordering for partially pullup nets:

  Based on the location of the partially pullup nets the leakage power dissipation can be changed. Same as how 2 input NAND gate has two different leakage values for 01 and 10 input vectors. So In future, reordering of the partially pullups connected pins can be analyzed.

- Facilitate for Flops in Gated Groups

Currently we avoid placing flip flops in gated groups. In Future this can also be analyzed.

- Improvement for the Gated Group finding algorithm

We don’t do any sorting of the identified worst leak gates and we run the algorithm as the worse gate are identified. If the algorithm can be improved such that the worst gate should be treated first or some other manner sometimes more power reduction can be achieved

- Feasibility of Physical Implementation

Currently this research was done with transistor level simulations. In Future this concept can be implemented in physical design tools and can have area, power and performance matrices in physical level.

References

[1] Wenxin Wang, Study of “Low Power Multi-threshold CMOS circuit optimization and CAD design”, The University of Guelph, May. 2004

[2] Paulo Francisco Butzen, and Renato Perez Ribas, “Leakage Current in sub-micrometer CMOS gates”, Universidade Federal do Rio Grande do Su

[3] Ajith Pal, “Lecture 26 : Power Disipation in CMOS Circuits”, Department of Electrical Engineering, IIT Kanpur, India

[4] Amit Singh Gaur, and Jyoti Budakoti, “Advanced Low Power CMOS Design to reduce power consumption in Deep Submicron Technologies in CMOS Circuit for VLSI Design”, International Journal of Advanced Research in Computer and Communication Engineering, Vol. 3, Issue 6, June 2014

[5] Nandita Dasgupta, “Lecture - 1 Introduction on VLSI Design”, Department of Electrical Engineering, IIT Madras, Jan 2009

[6] Sheu, B. J. et al. “BSIM: Berkeley Short-Channel IGFET Model for MOS Transistors”, IEEE Journal of Solid State Circuits, New York, v.SC-22, n.4, p. 558- 566, Aug. 1987

[7] R. Saleh, “Lecture 6 Leakage and Low-Power Design”, Dept. of ECE University of British Columbia

[8] Chris H. Kim, “Short Channel MOS Transistor”, University of Minnesota Dept. of ECE, USA

[9] N. Hanchate ; N. Ranganathan, “LECTOR: a technique for leakage reduction in CMOS circuits,” IEEE Transactions on Very Large Scale Integration (VLSI) systems, Volume: 12 , Issue: 2 , Feb. 2004

[10] Lin Yuan and Gang Qu, “A Combined Gate Replacement and Input Vector Control Approach for Leakage Current Reduction, IEEE Transactions on Very Large Scale Integration (VLSI) Systems” ,Volume: 14 , Issue: 2 , Feb. 2006

[11] Jonathan P. Halter and Farid N. Najm, “A Gate-Level Leakage Power Reduction Method for Ultra-Low-Power CMOS Circuitst”, Proceedings of CICC 97 - Custom Integrated Circuits Conference, May 1997

[12] A.J.Prasad, 2ABbadasari, “Input Vector Control Approach for Leakage Reduction and Low Power Consumption”, International journal of electronics and communication technology(IJECT)-VoL II,SP-1, Dec 2011

[13] Lei cheng, Liangdeng,”A Fast Simultaneous Input Vector Generation and Gate Replacement Algorithm for LeakagePower Reduction.”,In DAC july 24-28 2006

[14] V.LeelaRani, M.Madhavi Latha, “Implementation of genetic algorithm for Minimum leakage vector in input vector control approach”, IEEE conference,SPACES,Jan2015

[15] V.Leela Rani and M.Madhavi Latha, “particle Swarm Optimization Algorithm for Leakage Power Reduction in VLSI Circuits”, INTL journal of electronics and telecommunications,2016,Vol. 62, PP. 179-186 Manuscript received July 4, 2015; revised June, 2016

[16] Afshin Abdollahi, Farzan Fallah, “Leakage Current Reduction in CMOS VLSI Circuits by Input VectorControl”, IEEE Transactions on Very Large Scale Integration (VLSI) Systems archive Volume 12 Issue 2, February 2004

[17] Swarup Bhunia, Saibal Mukhopadhyay, “Low-Power Scan Design Using First-Level Supply Gating”, IEEE Transactions on very Large scale integration (VLSI) systems,Vol. 13,No. 3,March 2005

[18] Youngsoo Shin, Seungwhun Paik, Hyung-Ock Kim, “Semicustom Design of Zigzag Power-Gated Circuits in Standard Cell Elements”, IEEE Trascations on CAD Design of integrated Circuits and Systems, Vol. 28, No 03,March 2009.

[19] Akhila Abba, K Amarender, “Improved Power Gating Technique for Leakage Power Reduction”, Research Inventy: International Journal Of Engineering And Science, Vol.4, Issue 10, October 2014

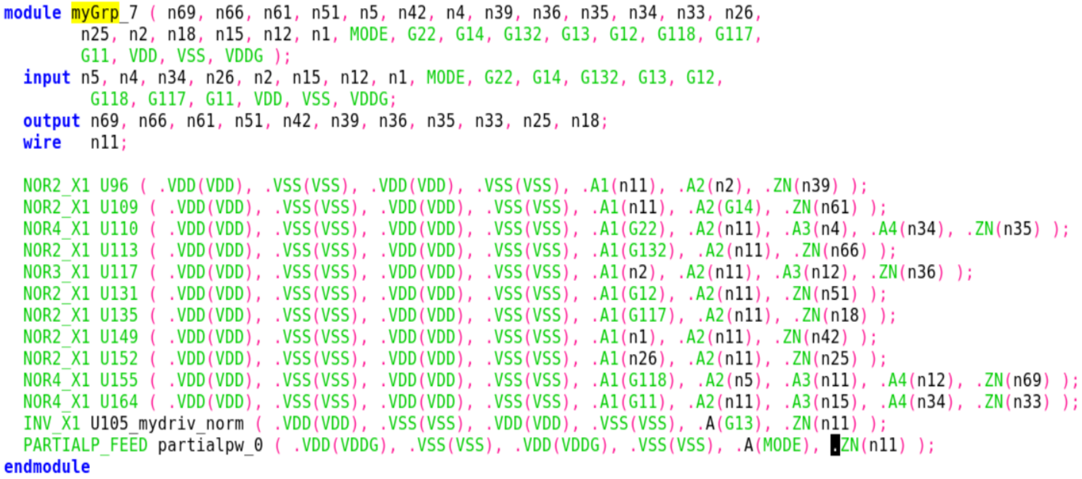
[20] Ismo Hänninen, “Low Power System-On-Chip Design, Chapter13: Retention Register Design”, TAMPERE University of Technology, Aug 2009

[21] “Chapter 5 Super-Cutoff Power Gating TCAM Design with Leakage Current Reduction”
[online]. Available: https://ir.nctu.edu.tw/bitstream/11536/80561/9/164909.pdf
[accessed: 25-12-2018]

[22] Silvaco Nangate 45nm open cell library
[online] Available:
https://www.silvaco.com/products/nangate/Library_Creator_Platform/index.html [accessed: 2-12-2018]

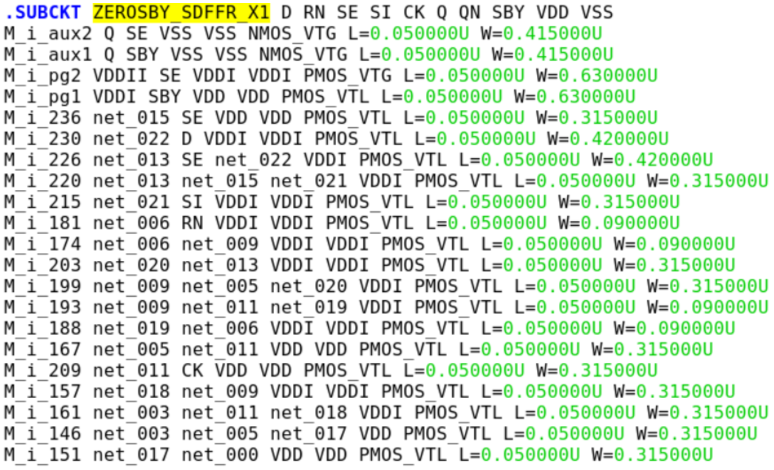
APPENDIX A: Verilog netlist of synthesized a Gated Group and hspice netlist of Modified FlipFlop

(i) Form Group in a synthesize netlist

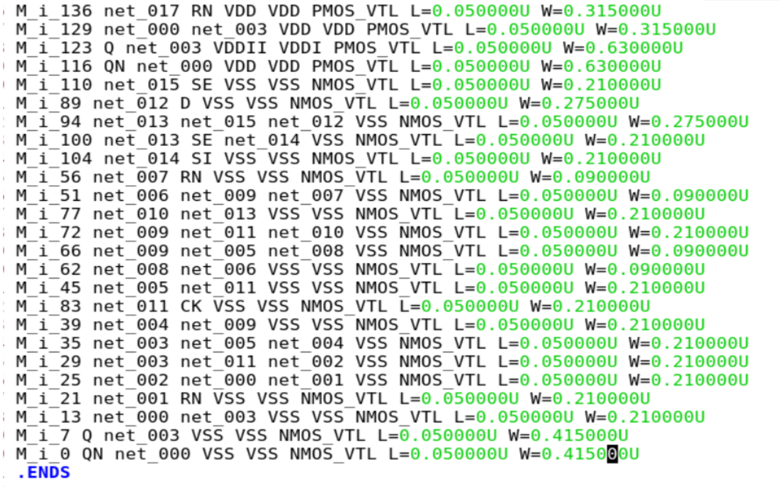


```
module myGrp_7 ( n69, n66, n61, n51, n5, n42, n4, n39, n36, n35, n34, n33, n26,
        n25, n2, n18, n15, n12, n1, MODE, G22, G14, G132, G13, G12, G118, G117,
        G11, VDD, VSS, VDDG );
  input n5, n4, n34, n26, n2, n15, n12, n1, MODE, G22, G14, G132, G13, G12,
         G118, G117, G11, VDD, VSS, VDDG;
  output n69, n66, n61, n51, n42, n39, n36, n35, n33, n25, n18;
  wire   n11;

  NOR2_X1 U96 ( .VDD(VDD), .VSS(VSS), .VDD(VDD), .VSS(VSS), .A1(n11), .A2(n2), .ZN(n39) );
  NOR2_X1 U109 ( .VDD(VDD), .VSS(VSS), .VDD(VDD), .VSS(VSS), .A1(n11), .A2(G14), .ZN(n61) );
  NOR4_X1 U110 ( .VDD(VDD), .VSS(VSS), .VDD(VDD), .VSS(VSS), .A1(G22), .A2(n11), .A3(n4), .A4(n34), .ZN(n35) );
  NOR2_X1 U113 ( .VDD(VDD), .VSS(VSS), .VDD(VDD), .VSS(VSS), .A1(G132), .A2(n11), .ZN(n66) );
  NOR3_X1 U117 ( .VDD(VDD), .VSS(VSS), .VDD(VDD), .VSS(VSS), .A1(n2), .A2(n11), .A3(n12), .ZN(n36) );
  NOR2_X1 U131 ( .VDD(VDD), .VSS(VSS), .VDD(VDD), .VSS(VSS), .A1(G12), .A2(n11), .ZN(n51) );
  NOR2_X1 U135 ( .VDD(VDD), .VSS(VSS), .VDD(VDD), .VSS(VSS), .A1(G117), .A2(n11), .ZN(n18) );
  NOR2_X1 U149 ( .VDD(VDD), .VSS(VSS), .VDD(VDD), .VSS(VSS), .A1(n1), .A2(n11), .ZN(n42) );
  NOR2_X1 U152 ( .VDD(VDD), .VSS(VSS), .VDD(VDD), .VSS(VSS), .A1(n26), .A2(n11), .ZN(n25) );
  NOR4_X1 U155 ( .VDD(VDD), .VSS(VSS), .VDD(VDD), .VSS(VSS), .A1(G118), .A2(n5), .A3(n11), .A4(n12), .ZN(n69) );
  NOR4_X1 U164 ( .VDD(VDD), .VSS(VSS), .VDD(VDD), .VSS(VSS), .A1(G11), .A2(n11), .A3(n15), .A4(n34), .ZN(n33) );
  INV_X1 U105_mydriv_norm ( .VDD(VDD), .VSS(VSS), .VDD(VDD), .VSS(VSS), .A(G13), .ZN(n11) );
  PARTIALP_FEED partialpw_0 ( .VDD(VDDG), .VSS(VSS), .VDD(VDDG), .VSS(VSS), .A(MODE), .ZN(n11) );
endmodule
```

(ii) A Modified Flop in spice Netlist



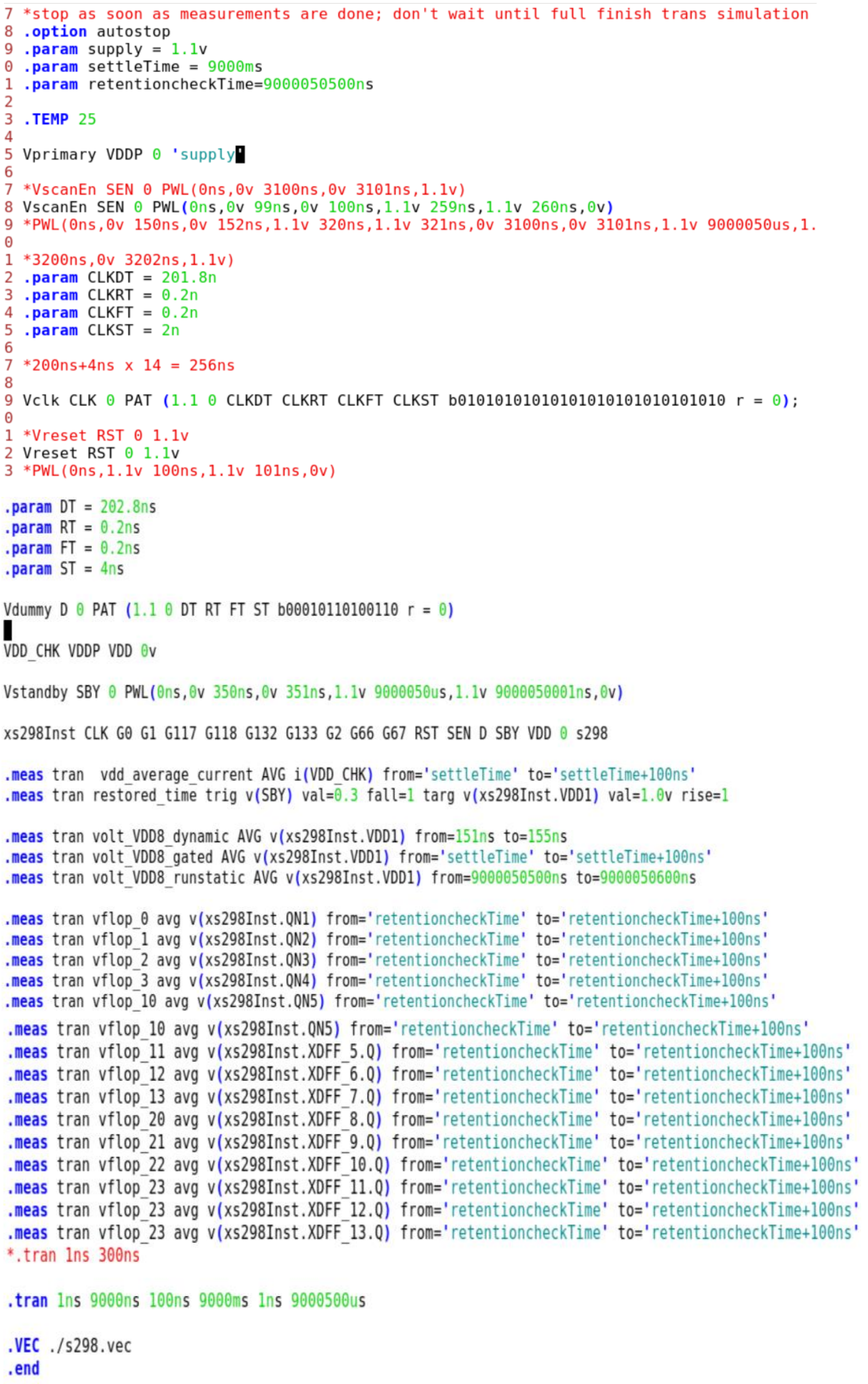
```
.SUBCKT ZEROSBY_SDFFR_X1 D RN SE SI CK Q QN SBY VDD VSS
M_i_aux2 Q SE VSS VSS NMOS_VTG L=0.050000U W=0.415000U
M_i_aux1 Q SBY VSS VSS NMOS_VTG L=0.050000U W=0.415000U
M_i_pg2 VDDII SE VDDI VDDI PMOS_VTG L=0.050000U W=0.630000U
M_i_pg1 VDDI SBY VDD VDD PMOS_VTL L=0.050000U W=0.630000U
M_i_236 net_015 SE VDD VDD PMOS_VTL L=0.050000U W=0.315000U
M_i_230 net_022 D VDDI VDDI PMOS_VTL L=0.050000U W=0.420000U
M_i_226 net_013 SE net_022 VDDI PMOS_VTL L=0.050000U W=0.420000U
M_i_220 net_013 net_015 net_021 VDDI PMOS_VTL L=0.050000U W=0.315000U
M_i_215 net_021 SI VDDI VDDI PMOS_VTL L=0.050000U W=0.315000U
M_i_181 net_006 RN VDDI VDDI PMOS_VTL L=0.050000U W=0.090000U
M_i_174 net_006 net_009 VDDI VDDI PMOS_VTL L=0.050000U W=0.090000U
M_i_203 net_020 net_013 VDDI VDDI PMOS_VTL L=0.050000U W=0.315000U
M_i_199 net_009 net_005 net_020 VDDI PMOS_VTL L=0.050000U W=0.315000U
M_i_193 net_009 net_011 net_019 VDDI PMOS_VTL L=0.050000U W=0.090000U
M_i_188 net_019 net_006 VDDI VDDI PMOS_VTL L=0.050000U W=0.090000U
M_i_167 net_005 net_011 VDD VDD PMOS_VTL L=0.050000U W=0.315000U
M_i_209 net_011 CK VDD VDD PMOS_VTL L=0.050000U W=0.315000U
M_i_157 net_018 net_009 VDDI VDDI PMOS_VTL L=0.050000U W=0.315000U
M_i_161 net_003 net_011 net_018 VDDI PMOS_VTL L=0.050000U W=0.315000U
M_i_146 net_003 net_005 net_017 VDD PMOS_VTL L=0.050000U W=0.315000U
M_i_151 net_017 net_000 VDD VDD PMOS_VTL L=0.050000U W=0.315000U
```



```
M_i_136 net_017 RN VDD VDD PMOS_VTL L=0.050000U W=0.315000U
M_i_129 net_000 net_003 VDD VDD PMOS_VTL L=0.050000U W=0.315000U
M_i_123 Q net_003 VDDII VDDI PMOS_VTL L=0.050000U W=0.630000U
M_i_116 QN net_000 VDD VDD PMOS_VTL L=0.050000U W=0.630000U
M_i_110 net_015 SE VSS VSS NMOS_VTL L=0.050000U W=0.210000U
M_i_89 net_012 D VSS VSS NMOS_VTL L=0.050000U W=0.275000U
M_i_94 net_013 net_015 net_012 VSS NMOS_VTL L=0.050000U W=0.275000U
M_i_100 net_013 SE net_014 VSS NMOS_VTL L=0.050000U W=0.210000U
M_i_104 net_014 SI VSS VSS NMOS_VTL L=0.050000U W=0.210000U
M_i_56 net_007 RN VSS VSS NMOS_VTL L=0.050000U W=0.090000U
M_i_51 net_006 net_009 net_007 VSS NMOS_VTL L=0.050000U W=0.090000U
M_i_77 net_010 net_013 VSS VSS NMOS_VTL L=0.050000U W=0.210000U
M_i_72 net_009 net_011 net_010 VSS NMOS_VTL L=0.050000U W=0.210000U
M_i_66 net_009 net_005 net_008 VSS NMOS_VTL L=0.050000U W=0.090000U
M_i_62 net_008 net_006 VSS VSS NMOS_VTL L=0.050000U W=0.090000U
M_i_45 net_005 net_011 VSS VSS NMOS_VTL L=0.050000U W=0.210000U
M_i_83 net_011 CK VSS VSS NMOS_VTL L=0.050000U W=0.210000U
M_i_39 net_004 net_009 VSS VSS NMOS_VTL L=0.050000U W=0.210000U
M_i_35 net_003 net_005 net_004 VSS NMOS_VTL L=0.050000U W=0.210000U
M_i_29 net_003 net_011 net_002 VSS NMOS_VTL L=0.050000U W=0.210000U
M_i_25 net_002 net_000 net_001 VSS NMOS_VTL L=0.050000U W=0.210000U
M_i_21 net_001 RN VSS VSS NMOS_VTL L=0.050000U W=0.210000U
M_i_13 net_000 net_003 VSS VSS NMOS_VTL L=0.050000U W=0.210000U
M_i_7 Q net_003 VSS VSS NMOS_VTL L=0.050000U W=0.415000U
M_i_0 QN net_000 VSS VSS NMOS_VTL L=0.050000U W=0.415000U
.ENDS
```

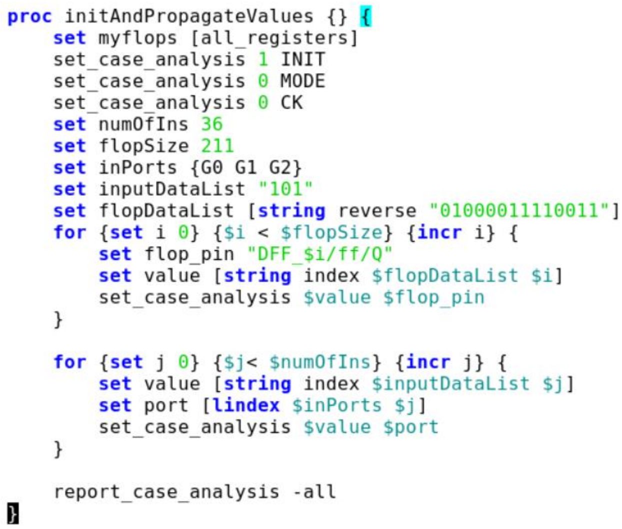
APPENDIX B Some Computer Programs Used in this research

(i) Hspice program use for leakage current value calculation , scan mode check and retention check for s298 design

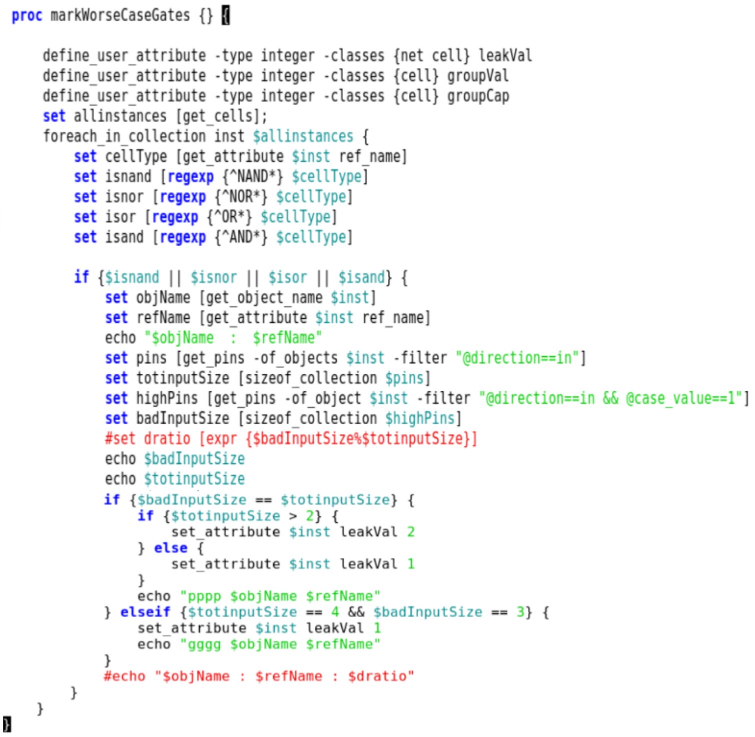


```
*stop as soon as measurements are done; don't wait until full finish trans simulation
.option autostop
.param supply = 1.1v
.param settleTime = 9000ms
.param retentioncheckTime=9000050500ns

.TEMP 25

Vprimary VDDP 0 'supply'

*VscanEn SEN 0 PWL(0ns,0v 3100ns,0v 3101ns,1.1v)
VscanEn SEN 0 PWL(0ns,0v 99ns,0v 100ns,1.1v 259ns,1.1v 260ns,0v)
*PWL(0ns,0v 150ns,0v 152ns,1.1v 320ns,1.1v 321ns,0v 3100ns,0v 3101ns,1.1v 9000050us,1.

*3200ns,0v 3202ns,1.1v)
.param CLKDT = 201.8n
.param CLKRT = 0.2n
.param CLKFT = 0.2n
.param CLKST = 2n

*200ns+4ns x 14 = 256ns

Vclk CLK 0 PAT (1.1 0 CLKDT CLKRT CLKFT CLKST b010101010101010101010101010 r = 0);

*Vreset RST 0 1.1v
Vreset RST 0 1.1v
*PWL(0ns,1.1v 100ns,1.1v 101ns,0v)

.param DT = 202.8ns
.param RT = 0.2ns
.param FT = 0.2ns
.param ST = 4ns

Vdummy D 0 PAT (1.1 0 DT RT FT ST b00010110100110 r = 0)

VDD_CHK VDDP VDD 0v

Vstandby SBY 0 PWL(0ns,0v 350ns,0v 351ns,1.1v 9000050us,1.1v 9000050001ns,0v)

xs298Inst CLK G0 G1 G117 G118 G132 G133 G2 G66 G67 RST SEN D SBY VDD 0 s298

.meas tran  vdd_average_current AVG i(VDD_CHK) from='settleTime' to='settleTime+100ns'
.meas tran restored_time trig v(SBY) val=0.3 fall=1 targ v(xs298Inst.VDD1) val=1.0v rise=1

.meas tran volt_VDD8_dynamic AVG v(xs298Inst.VDD1) from=151ns to=155ns
.meas tran volt_VDD8_gated AVG v(xs298Inst.VDD1) from='settleTime' to='settleTime+100ns'
.meas tran volt_VDD8_runstatic AVG v(xs298Inst.VDD1) from=9000050500ns to=9000050600ns

.meas tran vflop_0 avg v(xs298Inst.QN1) from='retentioncheckTime' to='retentioncheckTime+100ns'
.meas tran vflop_1 avg v(xs298Inst.QN2) from='retentioncheckTime' to='retentioncheckTime+100ns'
.meas tran vflop_2 avg v(xs298Inst.QN3) from='retentioncheckTime' to='retentioncheckTime+100ns'
.meas tran vflop_3 avg v(xs298Inst.QN4) from='retentioncheckTime' to='retentioncheckTime+100ns'
.meas tran vflop_10 avg v(xs298Inst.QN5) from='retentioncheckTime' to='retentioncheckTime+100ns'
.meas tran vflop_10 avg v(xs298Inst.QN5) from='retentioncheckTime' to='retentioncheckTime+100ns'
.meas tran vflop_11 avg v(xs298Inst.XDFF_5.Q) from='retentioncheckTime' to='retentioncheckTime+100ns'
.meas tran vflop_12 avg v(xs298Inst.XDFF_6.Q) from='retentioncheckTime' to='retentioncheckTime+100ns'
.meas tran vflop_13 avg v(xs298Inst.XDFF_7.Q) from='retentioncheckTime' to='retentioncheckTime+100ns'
.meas tran vflop_20 avg v(xs298Inst.XDFF_8.Q) from='retentioncheckTime' to='retentioncheckTime+100ns'
.meas tran vflop_21 avg v(xs298Inst.XDFF_9.Q) from='retentioncheckTime' to='retentioncheckTime+100ns'
.meas tran vflop_22 avg v(xs298Inst.XDFF_10.Q) from='retentioncheckTime' to='retentioncheckTime+100ns'
.meas tran vflop_23 avg v(xs298Inst.XDFF_11.Q) from='retentioncheckTime' to='retentioncheckTime+100ns'
.meas tran vflop_23 avg v(xs298Inst.XDFF_12.Q) from='retentioncheckTime' to='retentioncheckTime+100ns'
.meas tran vflop_23 avg v(xs298Inst.XDFF_13.Q) from='retentioncheckTime' to='retentioncheckTime+100ns'
*.tran 1ns 300ns

.tran 1ns 9000ns 100ns 9000ms 1ns 9000500us

.VEC ./s298.vec
.end
```

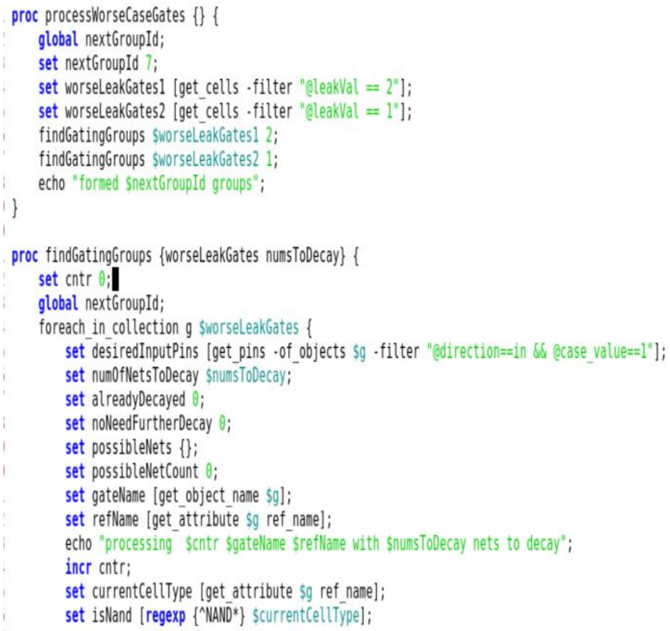
(ii) Part of the tcl program used to implement Gated Group formation algorithm



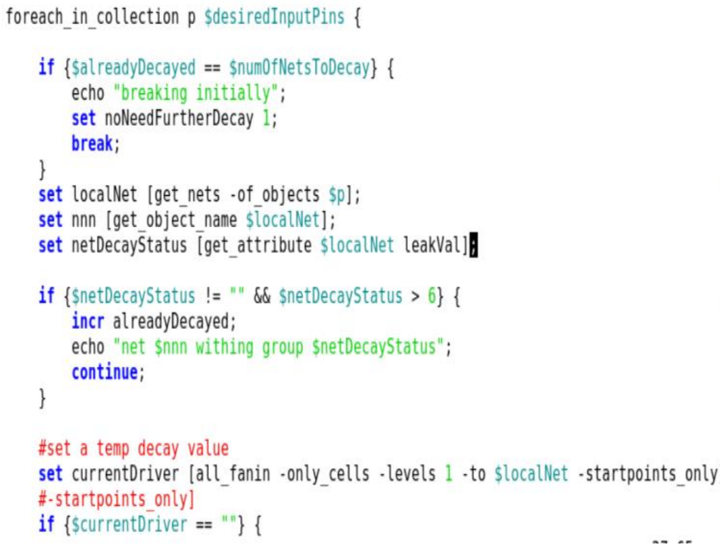
```
proc initAndPropagateValues {} {
    set myflops [all_registers]
    set_case_analysis 1 INIT
    set_case_analysis 0 MODE
    set_case_analysis 0 CK
    set numOfIns 36
    set flopSize 211
    set inPorts {G0 G1 G2}
    set inputDataList "101"
    set flopDataList [string reverse "01000011110011"]
    for {set i 0} {$i < $flopSize} {incr i} {
        set flop_pin "DFF_$i/ff/Q"
        set value [string index $flopDataList $i]
        set_case_analysis $value $flop_pin
    }

    for {set j 0} {$j< $numOfIns} {incr j} {
        set value [string index $inputDataList $j]
        set port [lindex $inPorts $j]
        set_case_analysis $value $port
    }

    report_case_analysis -all
}
```



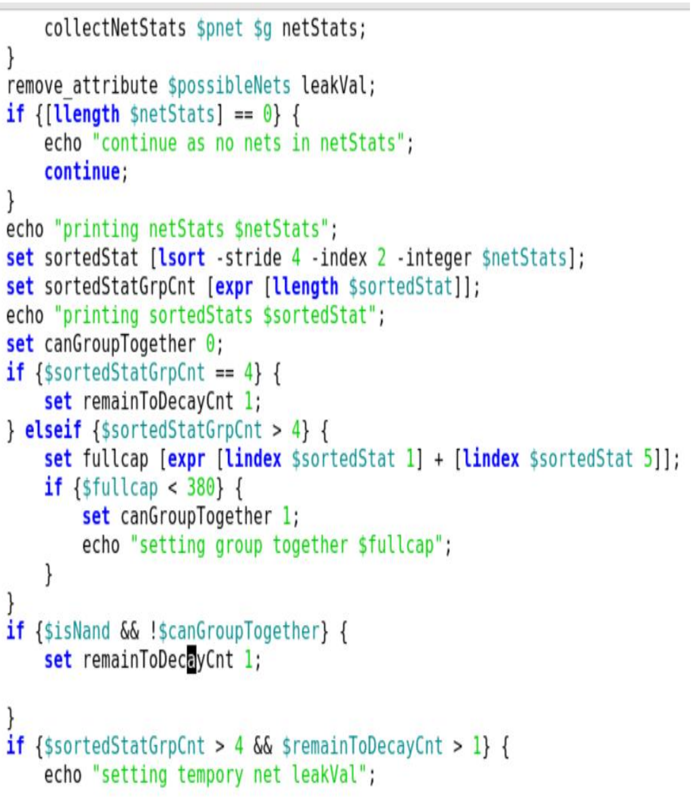
```
proc markWorseCaseGates {} {

    define_user_attribute -type integer -classes {net cell} leakVal
    define_user_attribute -type integer -classes {cell} groupVal
    define_user_attribute -type integer -classes {cell} groupCap
    set allinstances [get_cells];
    foreach_in_collection inst $allinstances {
        set cellType [get_attribute $inst ref_name]
        set isnand [regexp {^NAND*} $cellType]
        set isnor [regexp {^NOR*} $cellType]
        set isor [regexp {^OR*} $cellType]
        set isand [regexp {^AND*} $cellType]

        if {$isnand || $isnor || $isor || $isand} {
            set objName [get_object_name $inst]
            set refName [get_attribute $inst ref_name]
            echo "$objName  :  $refName"
            set pins [get_pins -of_objects $inst -filter "@direction==in"]
            set totinputSize [sizeof_collection $pins]
            set highPins [get_pins -of_object $inst -filter "@direction==in && @case_value==1"]
            set badInputSize [sizeof_collection $highPins]
            #set dratio [expr {$badInputSize%$totinputSize}]
            echo $badInputSize
            echo $totinputSize
            if {$badInputSize == $totinputSize} {
                if {$totinputSize > 2} {
                    set_attribute $inst leakVal 2
                } else {
                    set_attribute $inst leakVal 1
                }
                echo "pppp $objName $refName"
            } elseif {$totinputSize == 4 && $badInputSize == 3} {
                set_attribute $inst leakVal 1
                echo "gggg $objName $refName"
            }
            #echo "$objName : $refName : $dratio"
        }
    }
}
```



```
proc processWorseCaseGates {} {
    global nextGroupId;
    set nextGroupId 7;
    set worseLeakGates1 [get_cells -filter "@leakVal == 2"];
    set worseLeakGates2 [get_cells -filter "@leakVal == 1"];
    findGatingGroups $worseLeakGates1 2;
    findGatingGroups $worseLeakGates2 1;
    echo "formed $nextGroupId groups";
}

proc findGatingGroups {worseLeakGates numsToDecay} {
    set cntr 0;
    global nextGroupId;
    foreach_in_collection g $worseLeakGates {
        set desiredInputPins [get_pins -of_objects $g -filter "@direction==in && @case_value==1"];
        set numOfNetsToDecay $numsToDecay;
        set alreadyDecayed 0;
        set noNeedFurtherDecay 0;
        set possibleNets {};
        set possibleNetCount 0;
        set gateName [get_object_name $g];
        set refName [get_attribute $g ref_name];
        echo "processing  $cntr $gateName $refName with $numsToDecay nets to decay";
        incr cntr;
        set currentCellType [get_attribute $g ref_name];
        set isNand [regexp {^NAND*} $currentCellType];
```



```
foreach_in_collection p $desiredInputPins {

    if {$alreadyDecayed == $numOfNetsToDecay} {
        echo "breaking initially";
        set noNeedFurtherDecay 1;
        break;
    }
    set localNet [get_nets -of_objects $p];
    set nnn [get_object_name $localNet];
    set netDecayStatus [get_attribute $localNet leakVal];

    if {$netDecayStatus != "" && $netDecayStatus > 6} {
        incr alreadyDecayed;
        echo "net $nnn withing group $netDecayStatus";
        continue;
    }

    #set a temp decay value
    set currentDriver [all_fanin -only_cells -levels 1 -to $localNet -startpoints_only]
    #-startpoints_only]
    if {$currentDriver == ""} {
```



```
    collectNetStats $pnet $g netStats;
}
remove_attribute $possibleNets leakVal;
if {[llength $netStats] == 0} {
    echo "continue as no nets in netStats";
    continue;
}
echo "printing netStats $netStats";
set sortedStat [lsort -stride 4 -index 2 -integer $netStats];
set sortedStatGrpCnt [expr [llength $sortedStat]];
echo "printing sortedStats $sortedStat";
set canGroupTogether 0;
if {$sortedStatGrpCnt == 4} {
    set remainToDecayCnt 1;
} elseif {$sortedStatGrpCnt > 4} {
    set fullcap [expr [lindex $sortedStat 1] + [lindex $sortedStat 5]];
    if {$fullcap < 380} {
        set canGroupTogether 1;
        echo "setting group together $fullcap";
    }
}
if {$isNand && !$canGroupTogether} {
    set remainToDecayCnt 1;

}
if {$sortedStatGrpCnt > 4 && $remainToDecayCnt > 1} {
    echo "setting tempory net leakVal";
```