# Design of a Low-Power Low-Leakage Standard Cell Library for ASIC

Institute for Integrated Circuits
Hamburg University of Technology

**Master Thesis**

for the attainment of the academic degree
Master of Science

submitted by

**Rafiul Islam**
born on April 22, 1994 in Dhaka

April 11, 2022

| | |
|---|---|
| **First examiner:** | Prof. Dr.-Ing. Matthias Kuhl |
| **Second examiner:** | Prof. Dr.-Ing. Wolfgang Krautschneider |
| **Supervisor** | M. Sc. -Ing. Mohamed Faragalla |

# Declaration of authorship

I certify that I have written this work independently and that I have not used any sources or aids other than those indicated. Both content and verbatim content have been marked as such. The work has not been submitted in the same or similar form to any other examination office. I agree to the dissemination of my work for scientific purposes.

| | |
|---|---|
| ..... | ..... |
| Place, Date | Signature |

# Acknowledgement

I want to express my gratitude to my supervisor M. Sc. -Ing. Mohamed Faragalla for supporting me with immense guidance from my project work to my thesis. I am extremely lucky to have him as my supervisor not only for academic support but also for career advice and encouragement. I would like to thank Prof. Dr. -Ing. Matthias Kuhl for giving me the opportunity of working on this thesis and for the lectures during the earlier semesters.
Last but not least, I would like to thank my family, friends, and colleagues who continuously supported and motivated me in all aspects.

# Abstract

This work presents the development of the low-leakage and low-power standard cell library. The standard cell libraries are to be used in the logical synthesis process for implementing large and complex digital blocks for medical applications. Thus, minimizing the leakage and standby currents of the logic gates is of high importance to generate low leakage and low power digital designs.

The property of the stacked effect of transistors and Dynamic Leakage suppression (DLS) are analyzed to lower the leakage currents. An inverter, 2-input NAND, 2-input NOR, tri-state buffer, and a D flip-flop have been developed by stacked and DLS techniques. Moreover, these logic cells are characterized by the Cadence Liberate tool to generate the necessary views for the synthesis flow. A Verilog testbench has been used for the synthesis flow with the generated libraries to compare the power, performance and area .

The standard cell library is developed in XFAB 180 $nm$ CMOS process. The DLS logic cells have the least leakage current almost in the range of $10^{-15}$ to $10^{-18}$ $A$ whereas stacked transistor logic gates have higher leakage current flow in the range of 100 $fA$ to 500 $fA$. The area of both DLS and stacked transistor logic occupies almost 1.5-2 times more area than the conventional logic gates with minimum current drivability in the same process technology.

# Contents

# List of Figures

# List of Tables

# 1 Introduction

A Standard cell library is a collection of precise and characterized logic gates that can be used by a logical synthesis tool so that a digital system can be implemented automatically. The logic gates in the library are the basic building blocks of that digital system [3]. So far, there are many logic families implemented in digital electronics, however, the Complementary MOS (CMOS) is the greatest technology invented considering the zero leakage current ideally and low power consumption. Since static logic has some advantages over dynamic logic due to being highly amenable to automation tools, this thesis focuses on the implementation of static logic cells. Moreover, the problem of periodic charge leakage in the dynamic logic has accelerated the current trend of increased use of static CMOS logic [4]. As the name suggests, the static logic cells always require the supplying power to keep and process the data. When a system is powered by energy storage elements such as batteries or capacitors, it becomes essential to reduce the leakage current and power consumption of the logic cells to ensure the long-lasting operation of that system. As a result, this work focuses on designing a standard cell library with low leakage current and low power consumption.

## 1.1 Basic Logic Design Parameter

There are a few basic parameters that need to be considered while designing the logic cells such as cell stability, power consumption, performance, and area. In general power consumption, performance and area are co-optimized with respect to the application of the digital system. In the case of medical applications, power criteria are pivotal over speed and area of the system. Hence there is a trade-off between power consumption, performance, and area (PPA) depending on the application.

### 1.1.1 PPA Co-optimization

There are many methods proposed to reduce the leakage current of logic cells. To reduce the short channel effect, the length of transistors is increased which increases the area

and capacitances of the transistor. Due to higher capacitance for a large area, dynamic power is increased and speed is reduced. Furthermore, due to the increase in channel length of the transistor, leakage current is reduced and hence the static power. On the other hand, if the area of the logic gate is increased by increasing the width for higher current drivability, speed is enhanced significantly with high dynamic and static power consumption. There are some other circuit-level mechanisms to reduce the leakage current and power consumption, for instance: Multi-threshold CMOS (MTCMOS), Variable-threshold CMOS (VTCMOS), Dynamic Suppression Logic (DLS), and so on [2]. These techniques are further studied and implemented in the following chapters.

### 1.1.2 Cell Stability

The parameter by which cell stability of logic cells is measured is called Static Noise Margin (SNM). It is defined as the maximum value of DC noise voltage that can be tolerated by logic cells without changing the input data. For standard cells, the SNM parameter changes with the combination of inputs. For example, in the case of a 2-input NAND gate, there is four possible Voltage Transfer Characteristic (VTC) curve that can lead to the worst-case SNM value for that NAND gate. In the case of sequential logic, for example, flip-flops and latches, the SNM parameter can be found by the butterfly curve of the cell. The higher the noise margin is, the logic cell becomes more stable [1].
The logical stable points can be found from the VTC curve of the logic cells where $dV_{out}/dV_{in} = -1$. SNM can be calculated as: $NM_H = (V_{OH} - V_{IH})$ and $NM_L = (V_{OL} - V_{IL})$ where $NM_L, NM_H$ are shown in the Fig. 1.1.

Figure 1.1 shows the definition of noise margin with respect to driving and receving device and the other terminology is described below.
**High-level input voltage ($V_{IH}$)** is the minimum high input voltage to be recognized as a logic 1.
**Low-level input voltage ($V_{IL}$)** is the maximum low input voltage to be recognized as a logic 0.
**High-level output voltage ($V_{OH}$)** is the minimum high output voltage when the output is logic 1.
**Low-level output voltage ($V_{OL}$)** is the maximum low output voltage when the output is logic 0.

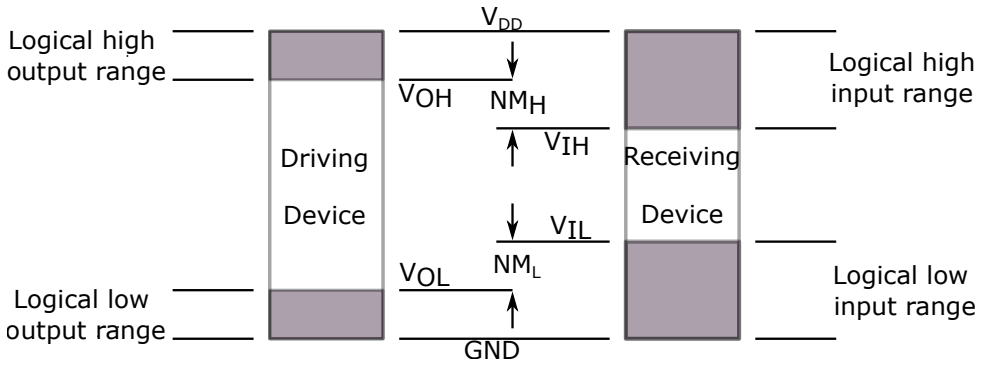



Figure 1.1: Graphical representation of SNM for logic cells [1]

### 1.1.3 Timing Metrics for sequential circuit

In general, flip-flop samples input at the clock edges. In order to capture the data properly, some timing constraints must be maintained otherwise metastability may occur [5].

**Setup Time ($t_{setup}$):** The required time for the data to be stable before the clock edge.

**Hold Time ($t_{hold}$):** The required time for the data to be stable after the clock edges.

**Propagation Delay ($t_{pcq}$):** The time required by the output data to be stable after the clock edge.

**Contamination Delay ($t_{ccq}$):** The time after clock edge when the output data is considered unstable.

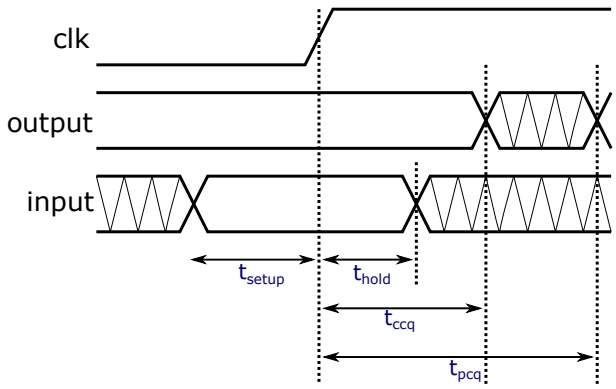



Figure 1.2: Timing metrics for sequential circuit

Figure 1.2 shows the timing metrics for the sequential circuit with respect to the positive edge of the clock. A standard cell library consists of various kinds of combinational logic cells with different driving capabilities, sequential logic cells, and some special cells like fillers, and decap cells. For this thesis, minimum logic cells required for the automation tool were designed. Later, a design testbench was verified with the designed standard cell library. The report is organized as follows:

- Chapter 2: describes leakage reduction techniques and theory behind the static logic design.
- Chapter 3: discusses the characterization process and generation of standard cell library views.
- Chapter 4: presents the verification and simulation results of the testbench with the generated library views.

# 2 Logic Core Design

This chapter describes some leakage reduction techniques in brief, the relationship between the threshold voltage and the leakage current, and finally the formation of the combinational and sequential logic cells and their characterization parameters comparison.

## 2.1 Transistor characterization

Figures 2.1 and 2.2 show the threshold voltage and threshold voltage profiles of the NMOS and PMOS respectively. The figures show the variation in threshold voltage concerning the width and length of the MOSFET.
Several effects set the value of the threshold voltage of the MOSFET. Non-uniform substrate doping distribution, short channel effects, and narrow channel effects have the most impact on the variation of the threshold voltage. The channel width also affects the threshold voltage and varies with the process as well, for example, isolation technology. The narrow width effect on $V_{th}$ is considered as [6]:

$$V_{th} = V_{th0ox} + (k_3 + k_{3b}V_{bseff})\frac{T_{ox}}{W_{eff} + W_0}\phi_s \tag{2.1}$$

where, ($k_3, k_{3b}, W_0$ parameters are the process dependent parameter. 'XFAB 180nm' uses negative value of $k_3$ which indicates the rise in threshold voltage with the increase in width of the transistor.
Transistors transfer characteristics for short channel and comparatively long channel device have been plotted in Fig. 2.3 and Fig. 2.4 respectively. It is apparent from the figures that the off-state current of the MOSFET reduces significantly in the super cut-off region. In the case of NMOS transistor, almost three magnitudes of off-state current can be reduced for a 200 mV voltage difference in the super cut-off region i.e. when the gate-source voltage ($V_{GS}$) is less than zero. PMOS transistor also faces a similar situation by the reduction in three magnitudes of off-state current (off-state current up to $10^{-18}$).

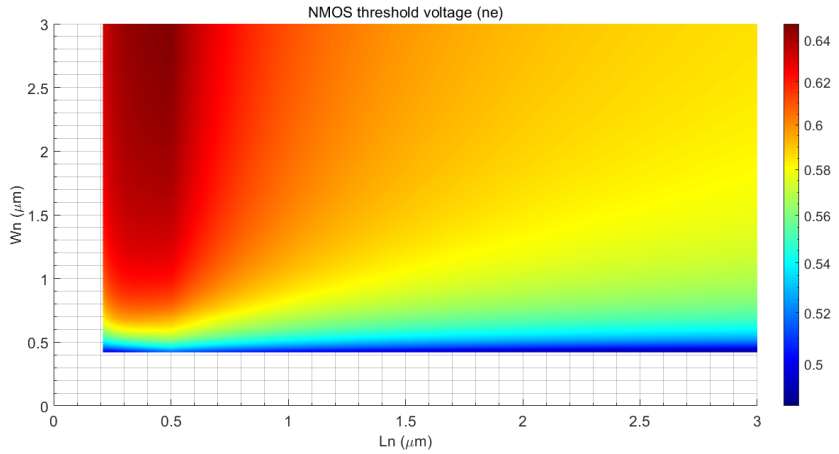



Figure 2.1: NMOS threshold voltage with respect to width and length of MOSFET

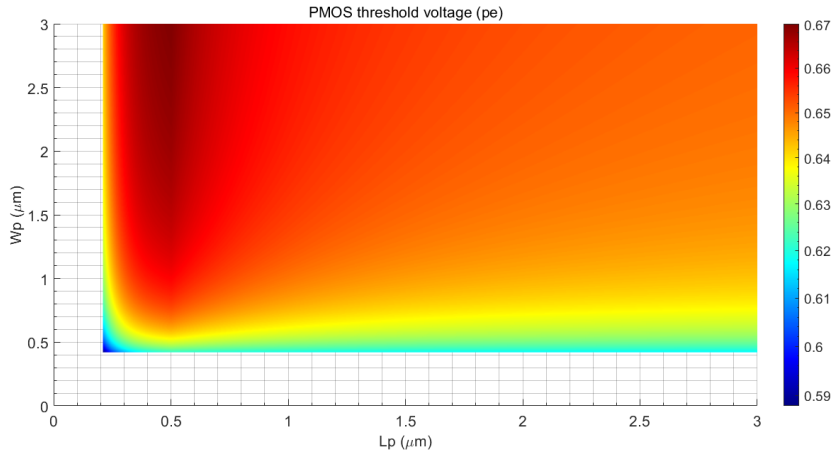



Figure 2.2: PMOS threshold voltage with respect to width and length of MOSFET

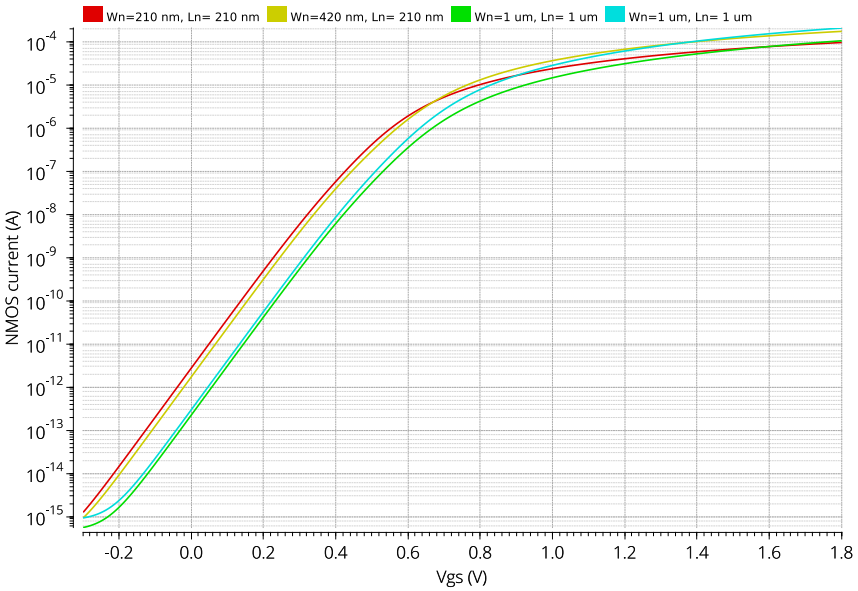



Figure 2.3: NMOS transfer characteristics

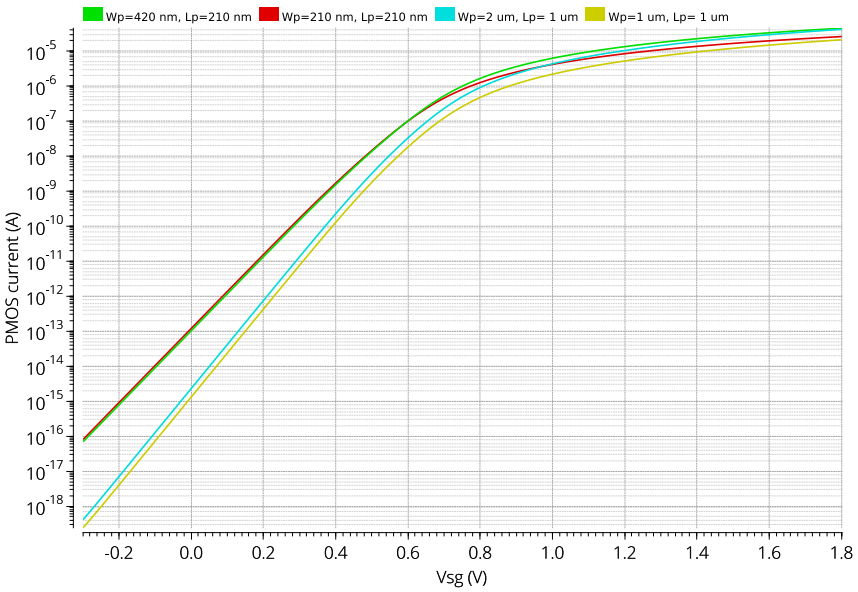
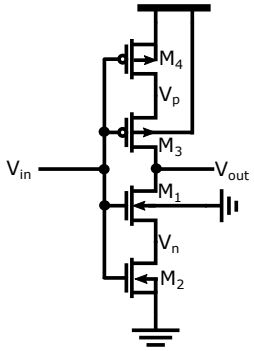
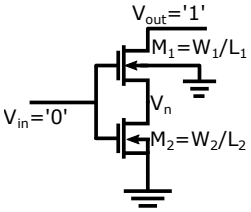


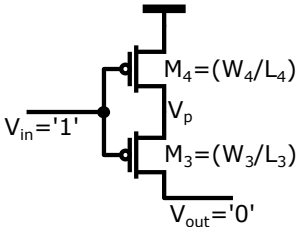

Figure 2.4: PMOS transfer characteristics




(a) Stacked inverter




(b) NMOS section of stacked inverter




(c) PMOS section of stacked inverter

Figure 2.5: Stacked inverter

## 2.2 Stacking Technique

The static and dynamic behavior due to the stacking effect is described in the following section.

### 2.2.1 Static Analysis

The subthreshold leakage current given in [7] can be expressed as:

$$I_{ds} = \mu_{eff} C_{ox} \frac{W}{L} (n-1) {V_T}^2 * e^{\frac{V_{gs}-V_{th}}{nV_T}} .(1 - \exp \frac{V_{ds}}{V_T}) \tag{2.2}$$

where, $\mu_{eff}$ =Effective mobility, $C_{ox}$ =Oxide capacitance, $n$ =subthreshold slope factor and $V_T$ =Thermal Voltage The stacking effect reduces the $V_{gs}$ of the transistor M2 which exponentially reduces the subthreshold current. Additionally, body to source voltage of transistor M2 is reduced which in turn increases the threshold voltage of the transistor and supports to reduce of the leakage current. Figure 2.5 shows a stacked inverter with NMOS and PMOS sections separately for the analysis [8].
When $V_{in} = 0$, it can be written that $V_{GS1} = -V_{DS2} = -V_{SB1} = -V_n$ and $V_{GS2} = V_{SB2} = 0$ Due to the body effect, there is a difference in the threshold voltages in two stacked NMOS transistors and two stacked PMOS transistors separately considering other conditions for

the threshold voltage variation are the same. The threshold voltage of two stacked transistor can be expressed as the difference and mean of two stacked NMOS and PMOS.

$$\Delta V_{th} = V_{th1} - V_{th2} \tag{2.3}$$

$$V_{t0} = \frac{V_{th1} + V_{th2}}{2} \tag{2.4}$$

Equations 2.3 and 2.4 can be expressed as:

$$V_{th1} = V_{t0} + \frac{\Delta V_{th}}{2} \tag{2.5}$$

$$V_{th2} = V_{t0} - \frac{\Delta V_{th}}{2} \tag{2.6}$$

The subthreshold leakage current equation can be rearranged by putting the value of $V_{th1}$ and $V_{th2}$ from equation 2.5 and 2.6 and considering the body effect. So, the subthreshold leakage current for transistor M1 is following:

$$I_{dn1} = \beta_1 (n-1) V_T^2 \exp(\frac{V_{GS1} - (V_{t0} + \frac{\Delta V_{th}}{2}) - \gamma V_{SB1}}{n.V_T})(1 - \exp\frac{-V_{DS1}}{V_T}) \tag{2.7}$$

$$I_{dn1} = \beta_1 (n-1) V_T^2 \exp(\frac{-V_n - V_{t0} - \frac{\Delta V_{th}}{2} - \gamma V_n}{n.V_T}) \tag{2.8}$$

Similarly, the subthreshold leakage current for transistor M2 can be written as:

$$I_{dn2} = \beta_2 (n-1) V_T^2 \exp(\frac{V_{GS2} - (V_{t0} - \frac{\Delta V_{th}}{2}) - \gamma V_{SB2}}{n.V_T})(1 - \exp\frac{-V_{DS2}}{V_T}) \tag{2.9}$$

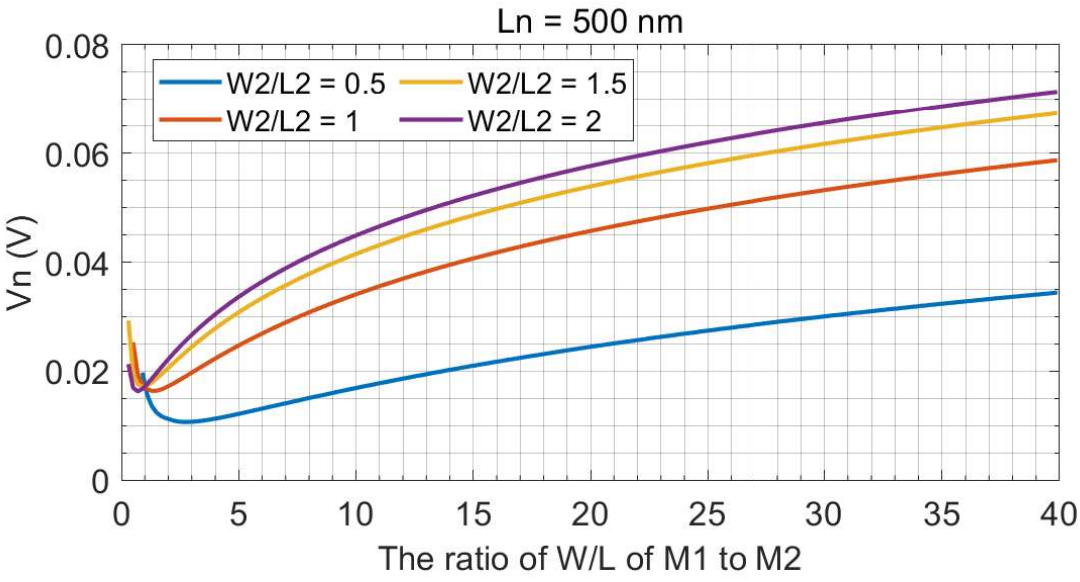
$$I_{dn2} = \beta_2 (n-1) V_T^2 \exp(\frac{-V_{t0} + \frac{\Delta V_{th}}{2}}{n.V_T})(1 - \exp\frac{-V_n}{V_T}) \tag{2.10}$$

Since, $I_{dn1} = I_{dn2}$, The value of $V_n$ can be determined by equating equation 2.8 and 2.10

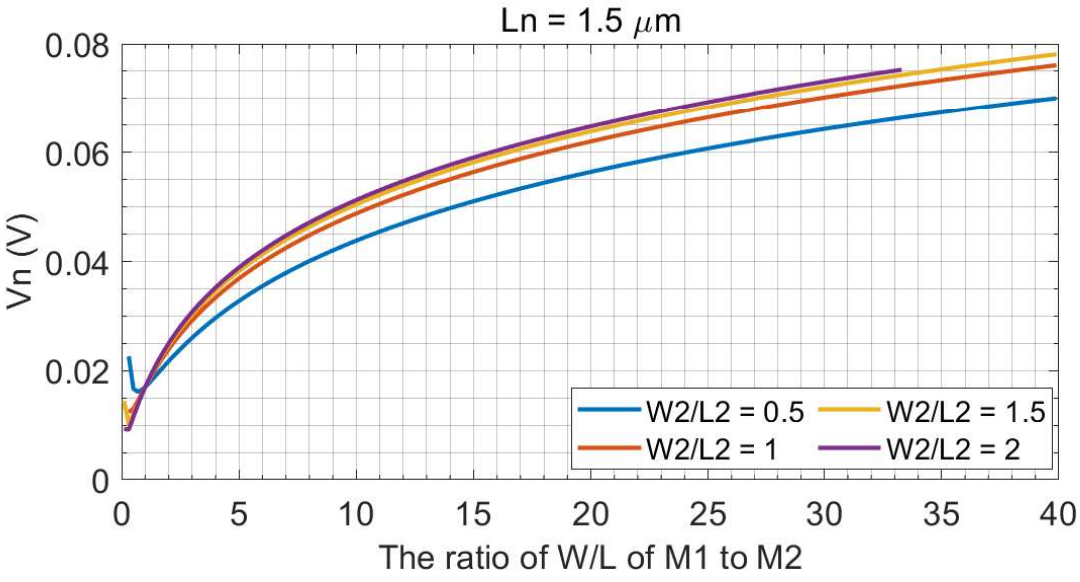
$$(1 - \exp\frac{-V_n}{V_T})V_n(1+\gamma) = nV_T \ln\frac{W_1/L_1}{W_2/L_2} - \Delta V_{th} \tag{2.11}$$

$$(1 - \exp\frac{-|V_p - V_{dd}|}{V_T})(V_p - V_{dd})(1+\gamma) = nV_T \ln\frac{W_3/L_3}{W_4/L_4} - \Delta V_{th} \tag{2.12}$$

Numerically solving the equation 2.12 in Matlab, we get the Fig. 2.7. Fig. 2.7a refers to the comparatively short channel device and Fig. 2.7b represents the analysis for the long channel device.

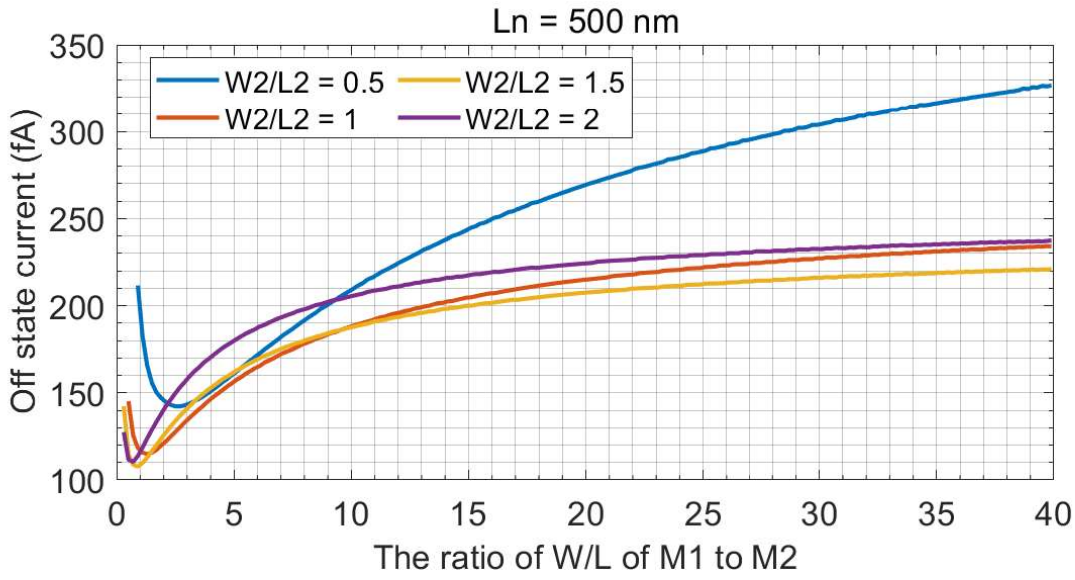



(a) Short channel device

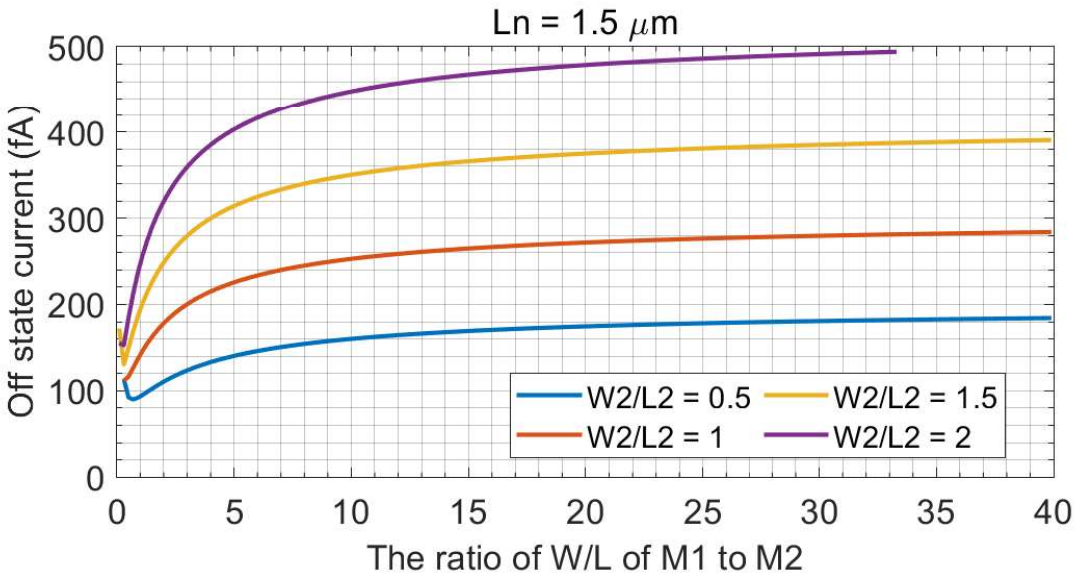



(b) Long channel device

Figure 2.6: Stacked inverter static mid-point Vn calculation




(a) Short channel device




(b) Long channel device

Figure 2.7: Stacked inverter static current calculation

Figure 2.8 shows the differences in threshold voltage for the single transistor and stacked transistor due to the body biasing. The dashed line and the solid line represent the single and stacked transistor respectively.

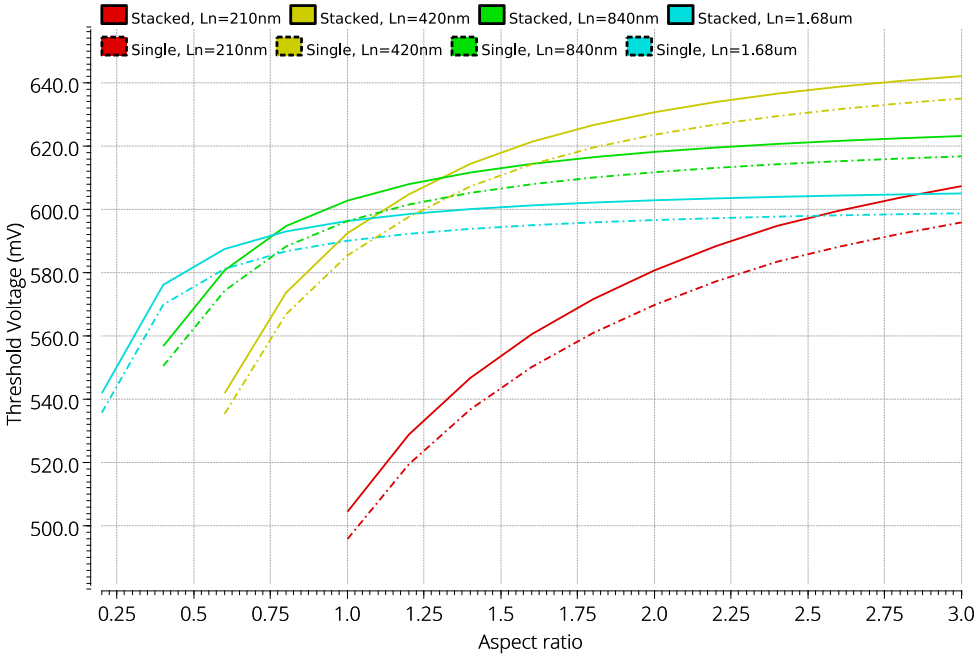
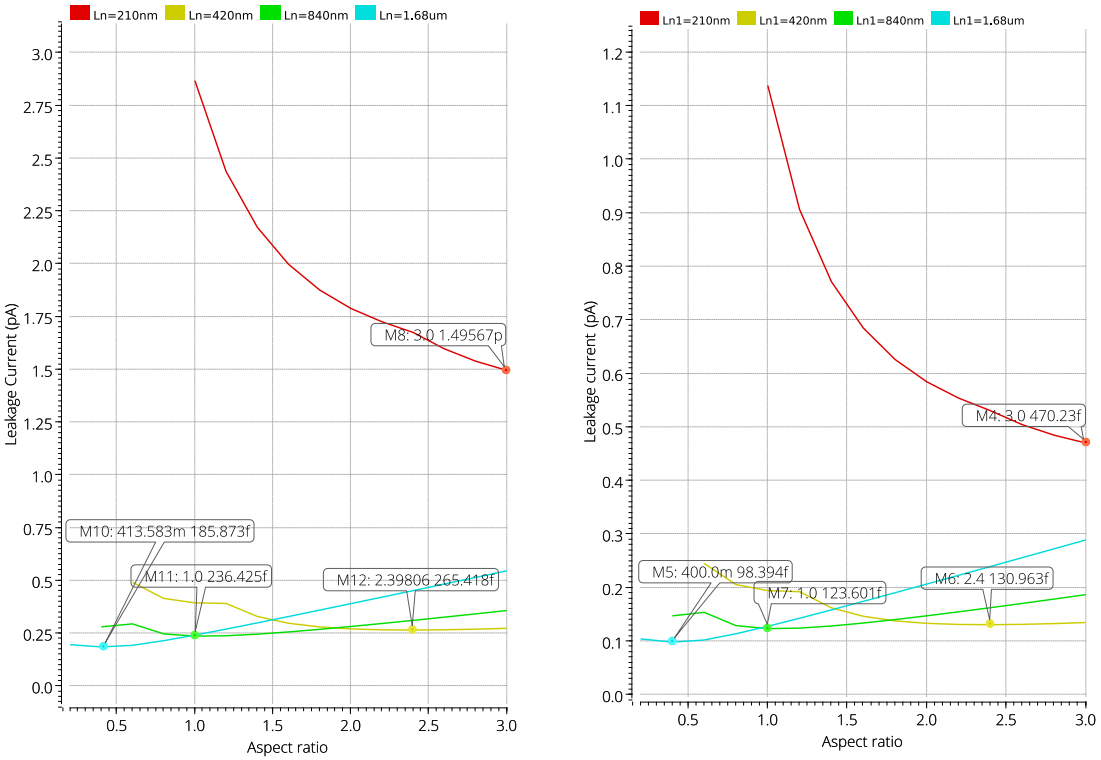



Figure 2.8: Threshold Voltage with respect to $(W/L)$ ratio of MOSFET




(a) NMOS

(b) Stacked NMOS

Figure 2.9: Leakage current versus $W/L$ ratio of the MOSFET

### 2.2.2 Stacking Technique over Single Transistor

Theoretically, two stacking transistors in series is equivalent to having one device with channel length equals to the sum of the channel-leghts of the stacked transistors. However, difference in off-state current is obsereved in comparatively small channel devices. Figure 2.9 shows the spice simulation leakage current result between stacked and single transistor. Figure 2.9a exhibits the result for the single NMOS transistor and Fig. 2.9b for the stacked NMOS. Different points for different length of MOSFET between Fig. 2.9a and 2.9b can be compared and it is found that stacked transistor can reduce leakage current 2-4 times than the single transistor. Another analysis is held where the single transistor length is doubled that of the stacked transistor and plotted in Fig. 2.10. A theoretical expression for this current exploration can be achieved by the following analysis.
For a single transistor, the subthreshold leakage current can be expressed as

$$I_{ds,single} = \beta_{single}(n-1){V_T}^2 \exp\frac{V_{gs}-V_{t0}}{nV_T}.(1-\exp\frac{V_{ds}}{V_T}) \tag{2.13}$$

For a stacked transistor, the subthreshold leakage current can be equal to equation 2.8.

$$I_{ds,stack} = \beta_{stack}(n-1)V_T^2 \exp(\frac{-V_n - V_{t0} - \frac{\Delta V_{th}}{2} - \gamma V_n}{n.V_T}) \tag{2.14}$$

Suppose, $L_{single} = 2 * L_{stack}$ and $V_{gs} = 0$. Now, Dividing the equation 2.13 by 2.14, we get,

$$\frac{I_{ds,stack}}{I_{ds,single}} = 2 * \exp(\frac{-V_n - \frac{\Delta V_{th}}{2} - \gamma V_n}{n.V_T}) \tag{2.15}$$

From equation 2.15, it can be acknowledged that a stacked transistor gives an advantage in leakage reduction based on the threshold voltage difference due to the body effect and super-threshold region of the MOSFET.

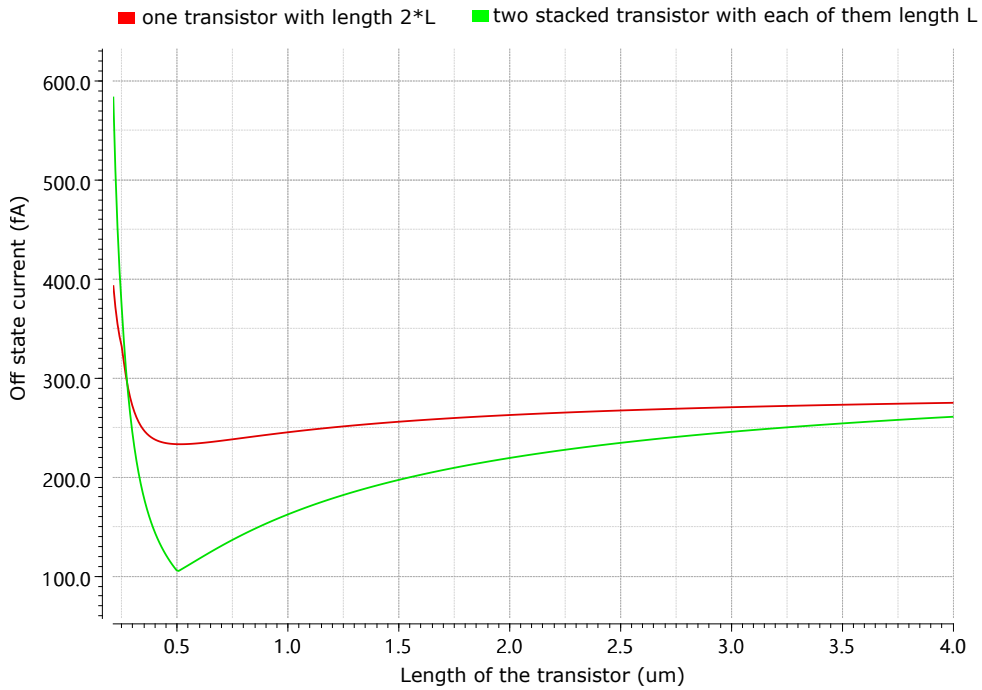



Figure 2.10: Off state current of a single transistor with double length and two stacked transistor

### 2.2.3 Dynamic Behavior Analysis

Dynamic behavior depends on the capacitance of the transistor. Both speed and dynamic power depend on the capacitance of the driving and receiving logic cells. Figure 2.11 shows how the capacitance is distributed over the driving and receiving cells.

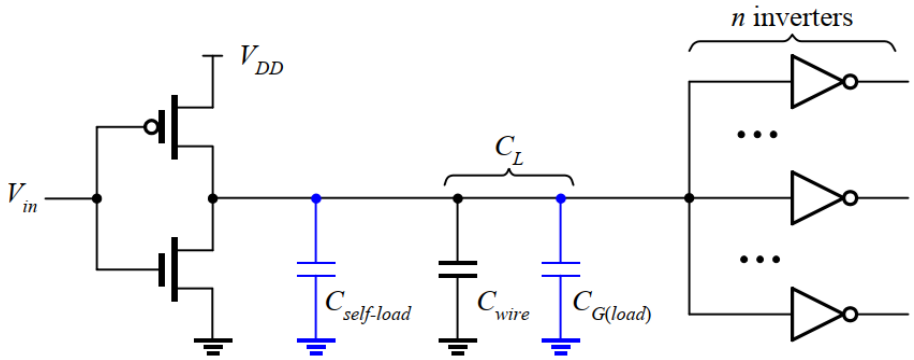



Figure 2.11: Loading capacitance components [2]

### Loading Capacitance

The loading capacitance of a logic gate consists of three types of capacitances [2]. They are:

**Gate Capacitance** consists of all the oxide-related capacitances such as $C_{GS}, C_{DS}, C_{GB}$ and gate-source and gate-drain overlap capacitances. Apart from overlap capacitances, all the other types of gate capacitances are a function of $V_{GS}$ which means overall gate capacitance can be different in different MOS operating regions.

$$C_G \approx C_{ox} W L \tag{2.16}$$

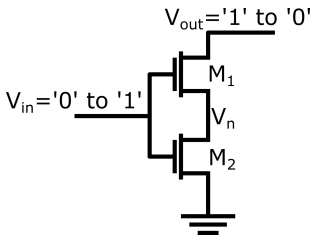
where, $C_{ox}$ is the gate-oxide capacitance in units of $\mu F/cm^2$.

**Interconnect Capacitance** can be divided depending on the length of the wires. In general, capacitance per unit area is used delivered by the foundry which is later multiplied by the length of the wire to calculate the interconnect capacitance.

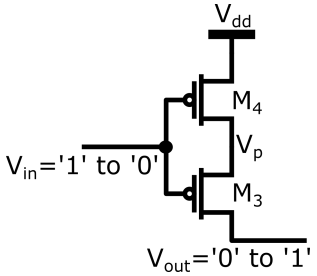
**Junction Capacitance** consists of the capacitance between the drain and source terminal and substrate of that transistor.

### Delay Calculation

Figure 2.12 shows the stacked inverter pull down and up network. Since $V_n$ is always less than $V_{dd}/2$, M2 will always be in linear region and M1 will be in saturation region until $V_{out}$ drops from $V_{dd}$ to $V_{th1} + V_n$. When $V_{out} < V_{th1} + V_n$, M1 enters into linear region.




(a) Inverter pull-down network




(b) Inverter pull-up network

Figure 2.12: Stacked Inverter pull up and down network

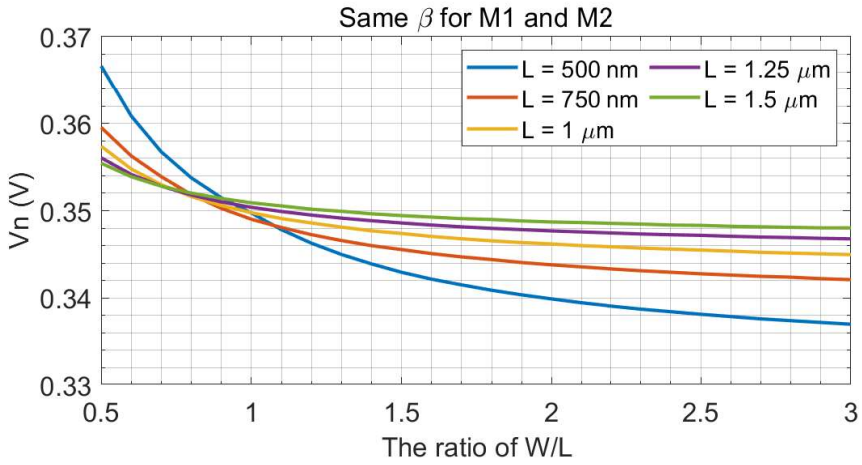



Figure 2.13: Midpoint value in a stacked design in case of dynamic behavior for different aspect ratio

Saturation current equation for the stacked NMOS transistor can be expressed as follows.

$$I_{dn1} = \frac{\beta_1}{2}(V_{dd} - V_n - V_{thn1})^2 \tag{2.17}$$

$$I_{dn2} = \frac{\beta_2}{2}[2(V_{dd} - V_{thn2})V_n - V_n^2] \tag{2.18}$$

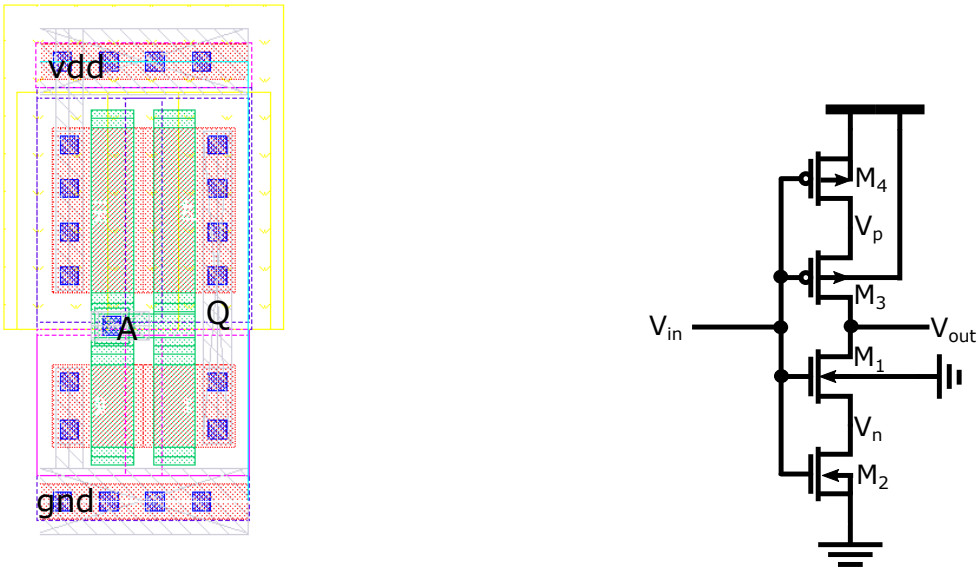
Equating $I_{dn1} = I_{dn2}$ and assuming $\beta_1 = \beta_2$ for simplicity, solviing for $V_n$

$$V_n = V_{dd} - \frac{V_{thn1}}{2} - \frac{V_{thn2}}{2} - \frac{\sqrt{(2V_{dd}^2 - 4V_{dd}V_{thn2} - V_{thn1}^2 + 2V_{thn1}V_{thn2} + V_{thn2}^2)}}{2} \tag{2.19}$$

Solving this equation numerically in MATLAB by taking the threshold voltage values from the SPICE simulation, we get the Fig. 2.13 and make sure that the M2 is in the saturation region for propagation delay calculation.
We know, $I\Delta t = \Delta Q = C\Delta V$ where, $\Delta t = t_{phl}$ and $\Delta V = V_{dd}/2$
Solving the equtaion for $t_{phl}$ and by putting the value of equation 2.17,

$$t_{phl} = \frac{2\frac{V_{dd}}{2}C_L}{\beta(V_{dd} - V_n - V_{thn1})^2} \tag{2.20}$$

$$t_{phl} = \frac{V_{dd}C_L}{\beta(V_{dd} - V_n - V_{thn1})^2} \tag{2.21}$$




(a) Layout view

(b) Schematic view

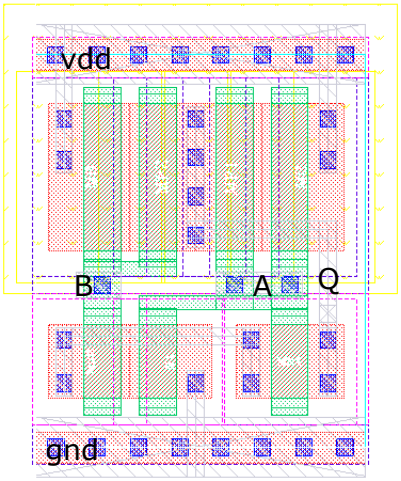
Figure 2.14: Stacked NOT gate

Low to high propagation delay can be calculated in similar way.

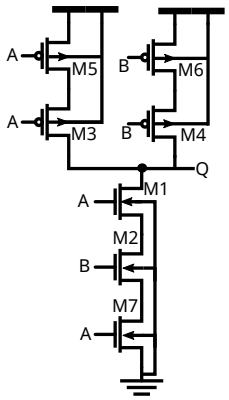
$$t_{plh} = \frac{V_{dd}C_L}{\beta(V_p + V_{thn3})^2} \tag{2.22}$$

### 2.2.4 Standard Cell Library Design Using Stacked Technique

Using the stacked technique, a logic inverter, 2-input NAND, 2-input NOR, tristate buffer, and d flip flop are designed in schematic and layout. The driving strength of these cells will be considered at a minimum. Figure 2.14, 2.15 and 2.16, 2.17 and 2.18 show the schematics and layouts for stacked standard cell library. The height of the cell is fixed at 5.32 $\mu m$. The characteristics of these logic cells have been shown at table 2.2, 2.1 and 2.3.

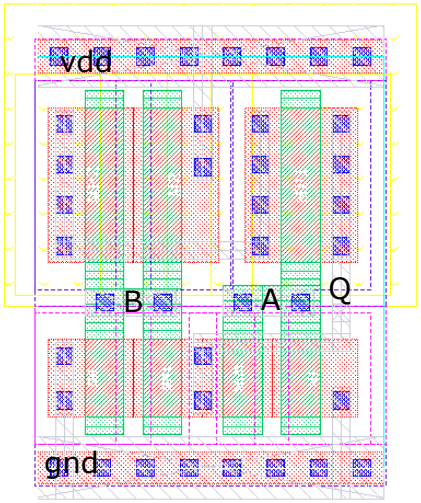



(a) Layout view

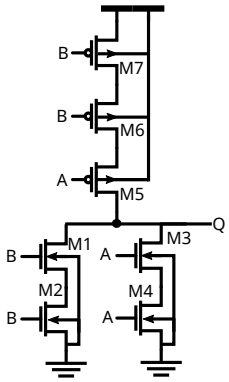



(b) Schematic view

Figure 2.15: Stacked NAND gate

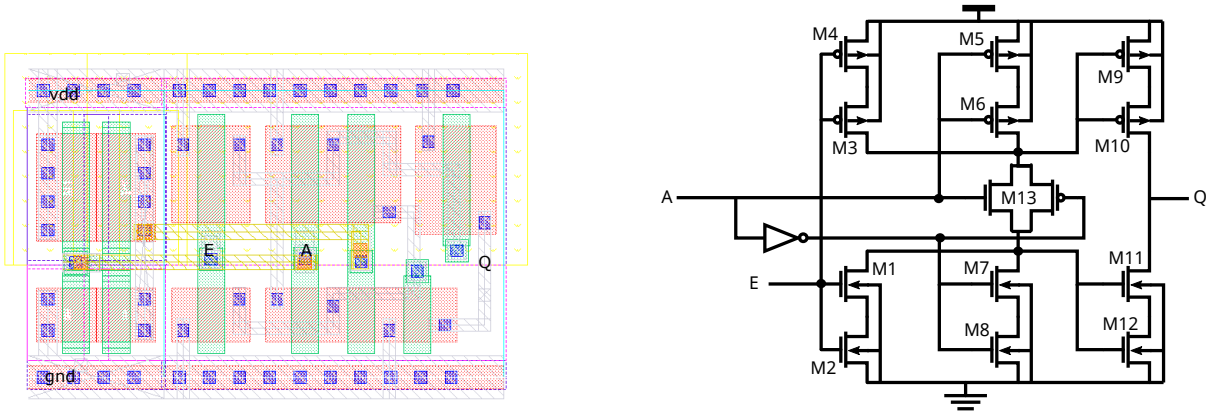



(a) Layout view



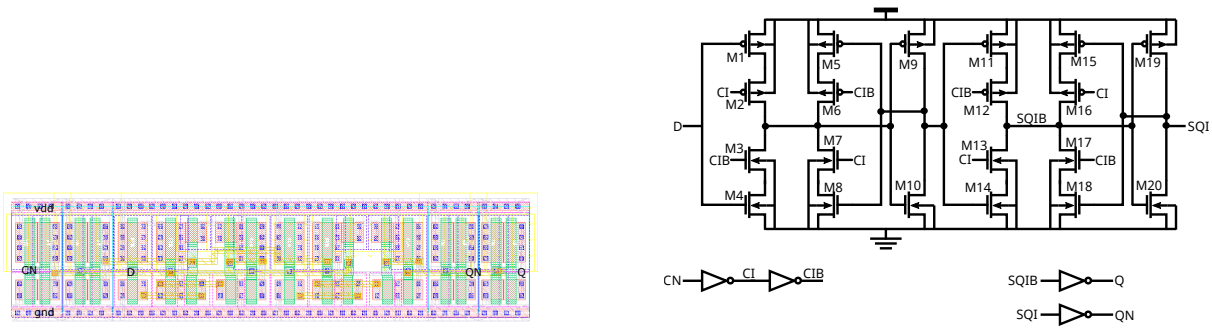

(b) Schematic view

Figure 2.16: Stacked NOR gate




(a) Layout view (b) Schematic view

Figure 2.17: Stacked Tri-state buffer gate




(a) Layout view (b) Schematic view

Figure 2.18: Stacked D-flip flop gate

## 2.3 Dynamic Leakage Suppression

### 2.3.1 Working Principle

In Dynamic Leakage Suppression (DLS) logic [9], an NMOS transistor is connected between the static CMOS logic and VDD, and a PMOS is connected between the static CMOS logic and the ground. The inputs of these top NMOS and bottom PMOS are feedback from the output. Because of this configuration, all the leaking transistors will be in the super-cutoff state i.e. having absolute gate-source voltage below zero. Figure 2.19 shows a DLS inverter and Fig. 2.20 shows the pull-up and pull-down network of the DLS inverter. As $V_{in}$ changes from logic '0' to logic '1', M1 shifts from super cut-off to weak inversion region, and M2 shifts from super cut-off to normal cut-off region. $V_{out}$ starts to discharge through M1 and M2 by the leakage current. The midpoint $V_p$ is also discharged to some extent, switching the transistor M3 and M4 to the super cut-off region. The opposite happens when $V_{in}$ changes from logic '1' to logic '0' [9].

### 2.3.2 Dynamic Behaviour Analysis

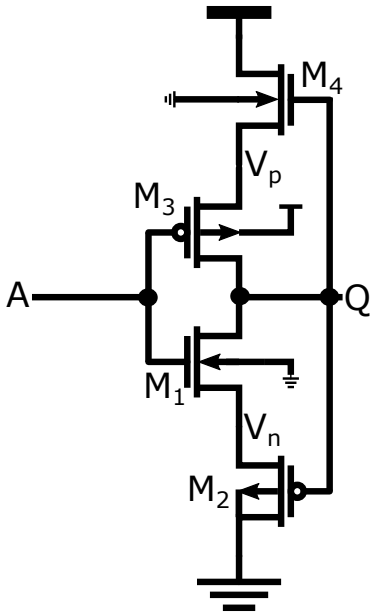
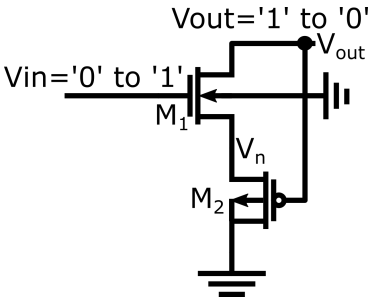
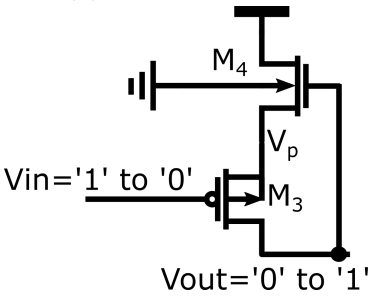



Figure 2.19: PMOS section of stack inverter

Since output switching is done by the leakage current in DLS logic, it is very slow (in the range of 2-15 Hz as in [9]) but consumes very low power (akmost 7 fW at 550 mV as in [9]). Due to the operation of the sub-threshold region, it is highly sensitive to the threshold




(a) DLS pull down




(b) DLS pull up

Figure 2.20: DLS inverter pull+up and pull+down network

voltage shift by the process and temperature variation. The rise transition in the output is limited due to the leakage current through the M4. However, due to the body effect in the transistor M4, a higher threshold voltage in that transistor is observed which can limit the leakage current during switching. Figure 2.21 shows the output and input voltage of a DLS inverter where a nominal threshold voltage transistor (ne) is used. Due to its high threshold voltage, the output voltage curve is degraded a lot. To eliminate this problem, a lower threshold voltage transistor (nel) is used so that during the switching the threshold voltage remains comparatively lower. Hence, a comparatively higher switching leakage current reduces the cell rise transition time. Figure 2.19 shows the output and input voltage of the DLS inverter where a low threshold voltage transistor (nel) M4 is used. In the following section, the characteristics of a low threshold voltage transistor have been described to find out the optimal threshold voltage point for proper operation [10].

### Low Threshold Voltage Transistor Characterization

In order to get a higher switching current and increase the switching speed, a low VT transistor has been used in the pull-up section of the transistor. Figure 2.23 and 2.24 show the threshold voltage profile of NMOS and PMOS respectively with respect to the width and length of the transistor.

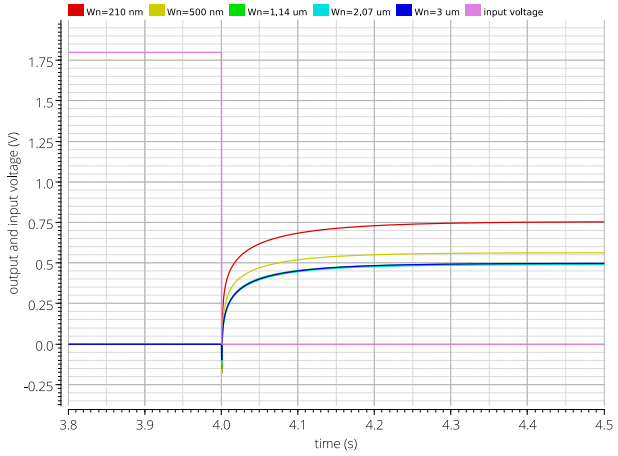



Figure 2.21: Output voltage with respect to different width of high threshold voltage transistor (ne)

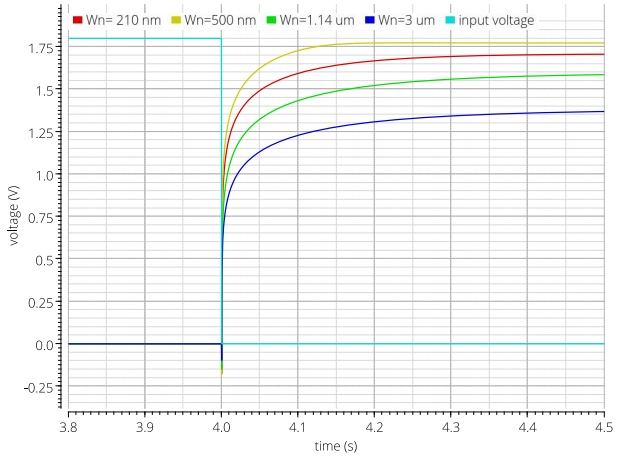



Figure 2.22: Output voltage with respect to different width of low threshold voltage transistor (nel)

If a lower length of the transistor is desirable, a lower threshold voltage can be achieved by keeping the width of the transistor smaller as well. It is also preferable because the lower capacitance of the transistor is ensured. A higher subthreshold leakage current is expected than the low VT transistors as shown in Fig. 2.25

### 2.3.3 Standard Cell Library Design Using DLS technique

Using the stacked technique, a logic inverter, 2-input NAND, 2-input NOR, tristate buffer, and d flip flop are designed in schematic and layout. The driving strength of these cells will be considered at a minimum. Figure 2.26, 2.27 and 2.28, 2.29 and 2.30 show the schematics and layouts for stacked standard cell library. The characteristics of these logic

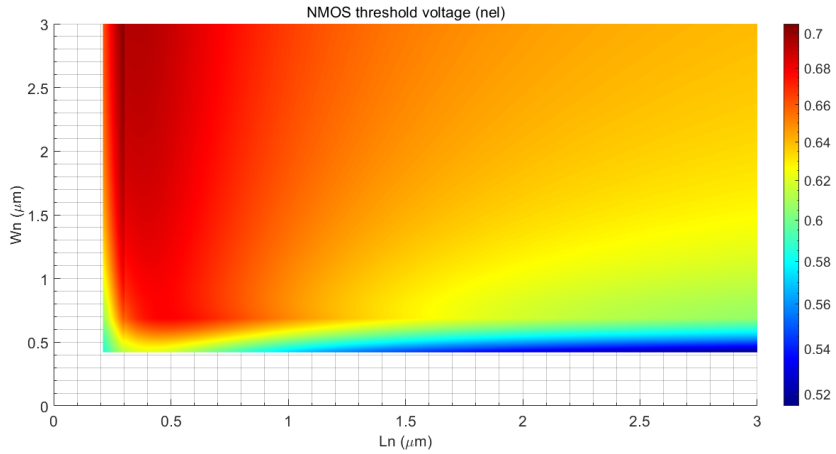



Figure 2.23: NMOS threshold voltage with respect to width and length of MOSFET

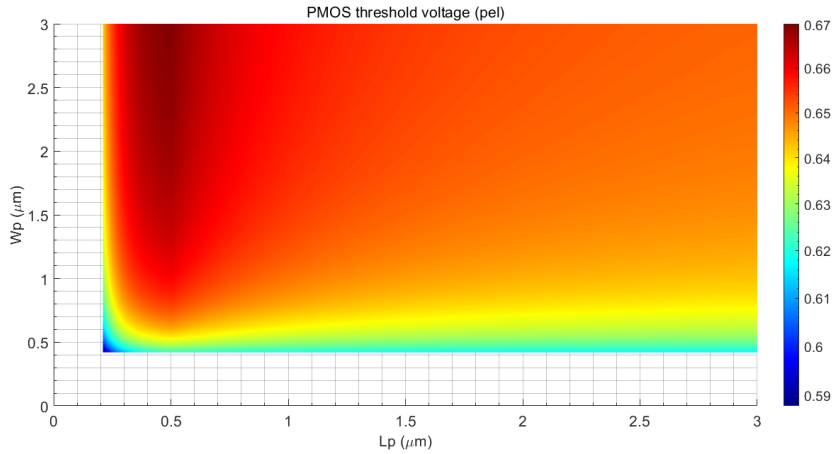



Figure 2.24: PMOS threshold voltage with respect to width and length of MOSFET

cells have been shown at table 2.2, 2.1 and 2.3.The height of the layout is fixed at 6.16 $\mu m$ for DLS logics.

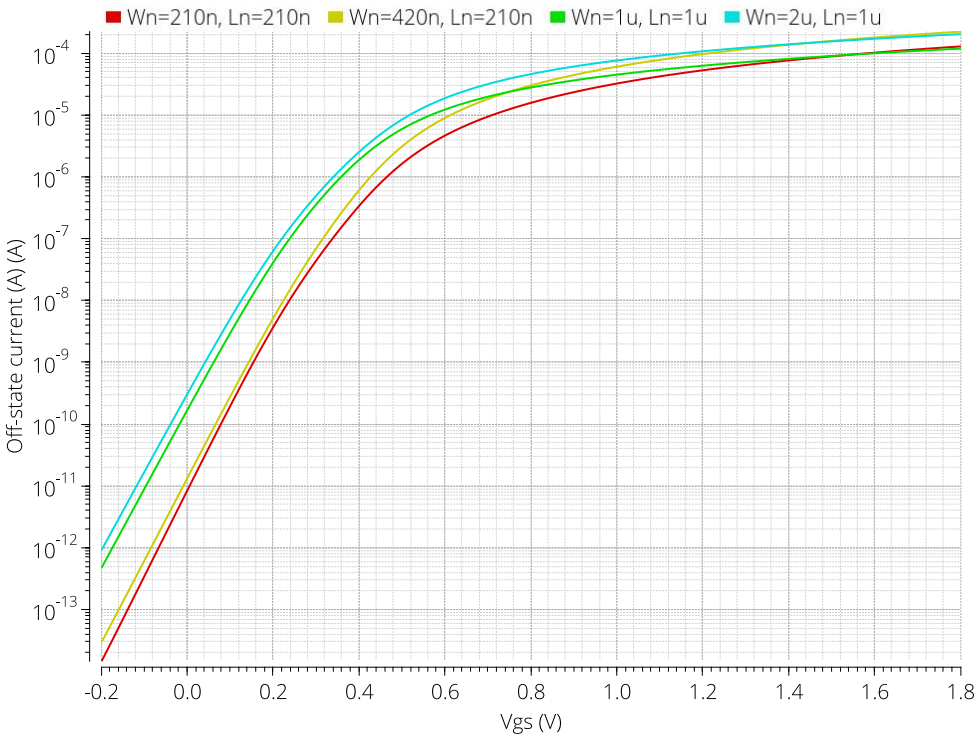
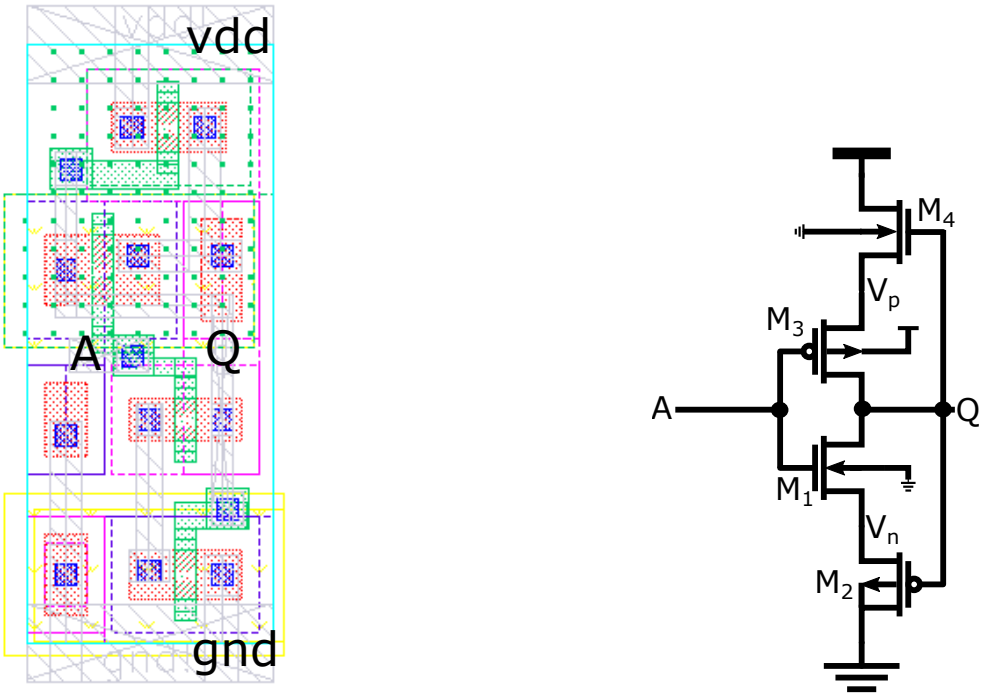


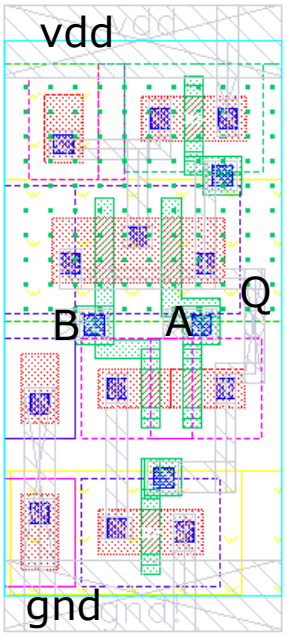

Figure 2.25: Off-state current of low VT NMOS



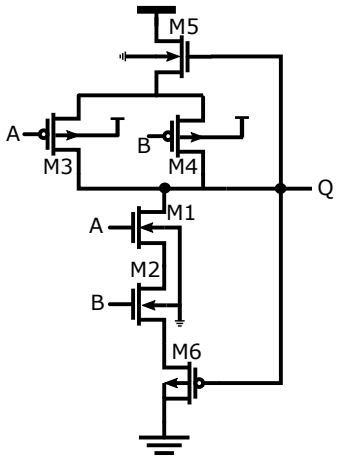

(a) Layout view

(b) Schematic view

Figure 2.26: DLS NOT gate

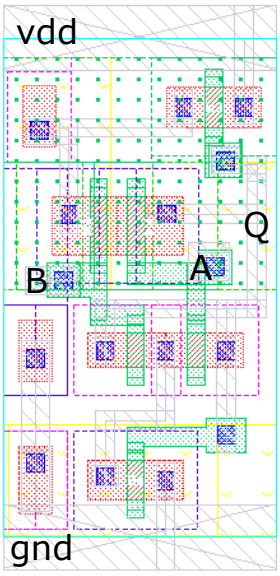



(a) Layout view

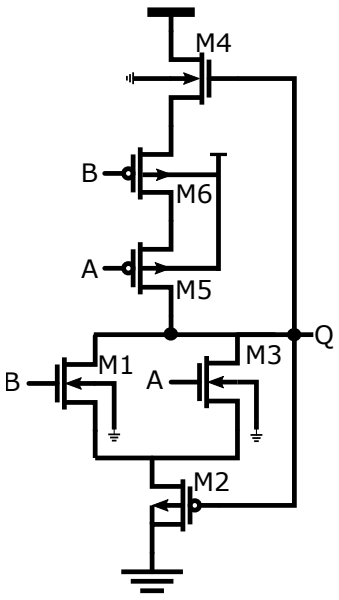



(b) Schematic view

Figure 2.27: DLS NAND gate

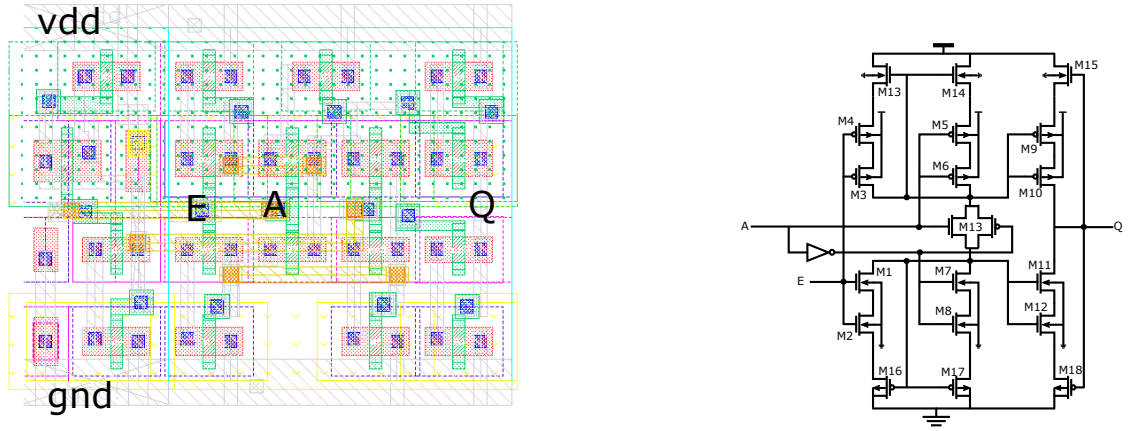



(a) Layout view



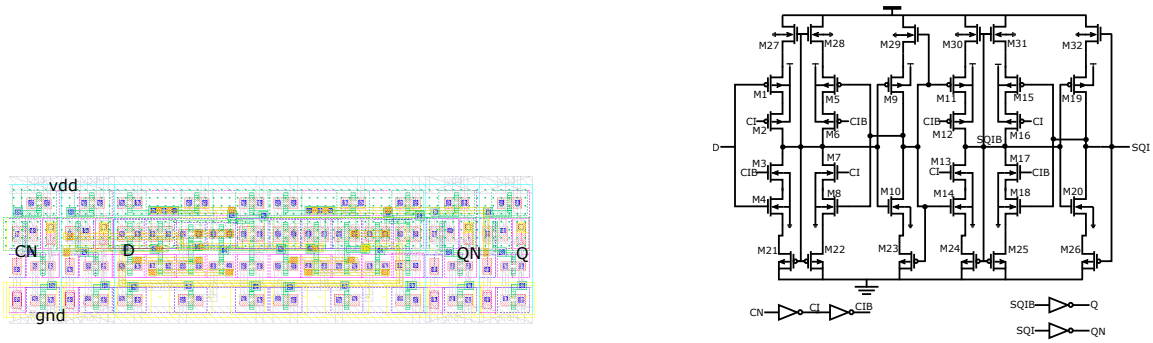

(b) Schematic view

Figure 2.28: DLS NOR gate




(a) Layout view (b) Schematic view

Figure 2.29: DLS Tri-state buffer gate




(a) Layout view (b) Schematic view

Figure 2.30: DLS flip-flop

## 2.4 Logic Gate Characteristics Comparison

Logic gates generated by stacked technique and DLS technique and conventional logic gates (minimum driving capable logic gate) from XFAB 180 nm are compared in case of power, area, and speed. Cadence Liberate tool was used to characterize all the logic cells and the process is described in the next chapter. Table 2.2 shows the comparison in static power consumption among the standard cell libraries. Static power consumption for a logic gate is different depending on the input combination of the gate. It is clear from the table that the DLS library has the least static power consumption, more than 100 times lower than the conventional library in some input combinations.
Table 2.1 shows the area comparison among the libraries. Although minimum dimensions have been used for DLS logic gates, the area is larger than the conventional gates due to its dual threshold voltage transistors. On the other hand, stacked logic gates have an almost comparable area to the DLS logic cells.
Table 2.3 shows the comparison of high to low and low to high propagation delay among the standard cell libraries. As expected, the conventional library has the least propagation delay whereas DLS logic gates have a very high propagation delay, almost to the range of mili-second which imposes the constraint to become very slow.

Table 2.1: Comparison of logic gates in case of area

| | Area (um^2) | | |
|---|---|---|---|
| Gates | **Stacked Library** | **DLS Library** | **Conventional Library** |
| **NOT** | 13.4064 | 15.5232 | 7.4264 |
| **2-input NAND** | 23.8336 | 18.9728 | 9.9352 |
| **2-input NOR** | 23.8336 | 20.6976 | 9.9352 |
| **D flip flop** | 137.5752 | 153.2608 | 57.6024 |
| **Tri-state Buffer** | 46.8958 | 52.36 | 24.988 |

Table 2.2: Comparison of logic gates in case of static Power

| | | Static Power (pW) | | |
|---|---|---|---|---|
| | | Stacked Library | DLS Library | Conventional Library |
| Gates | Input combination | | | |
| NOT | A=0 | 0.189258 | 0.015726 | 1.50523 |
| | A=1 | 0.004988 | 0.001781 | 0.110843 |
| | Avg. | 0.97123 | 0.008753 | 0.808 |
| 2-input NAND | A=0, B=0 | 0.127033 | 0.011308 | 0.479797 |
| | A=0, B=1 | 0.18953 | 0.015983 | 1.50528 |
| | A=1, B=0 | 0.368159 | 0.015741 | 1.26572 |
| | A=1, B=1 | 0.008618 | 0.002627 | 0.221324 |
| | Avg. | 0.173335 | 0.011415 | 0.868 |
| 2-input NOR | A=0, B=0 | 0.377695 | 0.020363 | 3.01015 |
| | A=0, B=1 | 0.007794 | 0.002456 | 0.104623 |
| | A=1, B=0 | 0.008209 | 0.002455 | 0.110889 |
| | A=1, B=1 | 0.004391 | 0.002455 | 0.0167439 |
| | Avg. | 0.099522 | 0.006932 | 0.8106 |
| D flip flop | D=0, Q=0 | 1.51846 | 0.095286 | 6.94041 |
| | D=1, Q=1 | 1.15932 | 1.77803 | 6.35651 |
| | D=1, Q=0 | 1.52408 | 0.091486 | 7.86107 |
| | Avg. | 1.40062 | 0.654933 | 7.053 |
| Tri-state Buffer | A=1, E=1, Q=1 | 0.394745 | 0.022403 | 1.57466 |
| | A=1, E=0, Q=0 | 0.578371 | 0.478862 | 3.30525 |
| | A=0, Q=0 | 0.76091 | 0.036244 | 3.30539 |
| | E=0, Q=1 | 0.578371 | 0.500005 | 4.38544 |
| | Avg. | 0.578099 | 0.25939 | 3.377 |

Table 2.3: Comparison of logic gates in case of delay

| | | Propagation Delay (ns) | | |
|---|---|---|---|---|
| | | Stacked Library | DLS Library | Conventional Library |
| Gates | Propagation Delay | | | |
| NOT | low to high | 0.349 | 548798 | 0.048 |
| | high to low | 0.152 | 955.667 | 0.028 |
| 2-input NAND | low to high | 0.364 | 879095 | 0.061 |
| | high to low | 0.238 | 1293.38 | 0.051 |
| 2-input NOR | low to high | 0.399 | 625786 | 0.109 |
| | high to low | 0.148 | 1123.95 | 0.032 |
| D flip flop | low to high | 2.692 | 903430 | 0.363 |
| | high to low | 4.014 | 551.843 | 0.588 |
| Tri-state Buffer | low to high | 0.510 | 6.74e+06 | 0.239 |
| | high to low | 0.815 | 3.92e+06 | 0.395 |

# 3 Standard Cell Library Characterization

This chapter describes the characterization process of the Standard cell library in Cadence Liberate. Delay models and power models for characterizing are a matter of discussion in this chapter.

## 3.1 Characterization Process

Cadence Liberate is a standard cell library characterization tool that was used for characterizing the generated cells. The tool requires a few fundamental objects to characterize standard cells. Device model files, parasitic extracted netlists of the cells, template definition, and the arc definition are the most basic ones as shown in Fig. 3.1.
The liberty file, datasheets for individual gates, and Verilog files are the product of this characterization tool. The liberty file is later used by the synthesis tool for generating the gate-level netlist for the large and complex blocks automatically. As an example, the generated Verilog file of stacked D flip-flop and datasheet of stacked D flip-flop and DLS Nand gate are shown in the appendix C.

## 3.2 Delay Model

Different kinds of industry-standard delay models are available in the Cadence Liberate tool. They are:
**Non-linear delay model** is characterized by measuring the delay and output transition when the various range of input transitions and output loads are given. Input transition can be given by many methods such as a linear ramp, pre-driver waveforms, or user-defined waveforms.
**Composete current source model** has an almost similar setup as the non-linear delay model as shown in Fig. 3.2. However, instead of only measuring the delay value, an output current is measured for every input transition and output load. Generally, this is done by attaching a voltage source at the output pin of the target gate to measure the

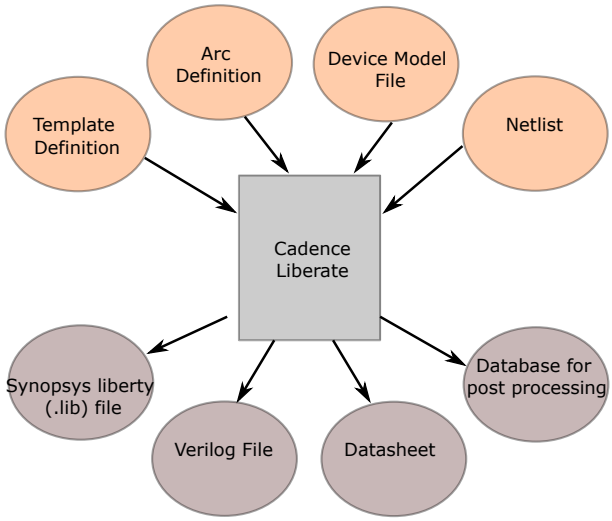



Figure 3.1: Input and output for the Cadence liberate tool

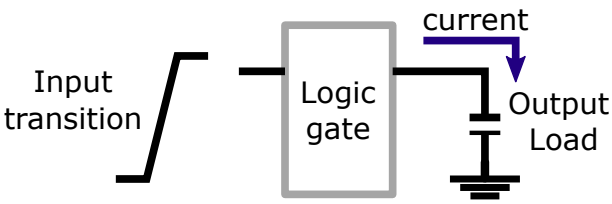



Figure 3.2: Composete current source model

current of the gate [11].

**Effective current source model** has the same setup as the composite current source model. The only difference is the characterization of the output voltage instead of out current in the CCS delay model[11].

## 3.3 Power Model

There are different kinds of power models available in the Cadence Liberate tool. They are:

- Leakage power model.
- Hidden power model.
- Switching power model.

**Leakage power model** uses all the possible combinations for a logic gate to evaluate state-dependent leakage characterization. In Cadence liberate tool, 'define_leakage'

command is used to find out the leakage current in every state.

**Hidden power model** is the energy consumption model when the input combinations are changed but there is no change in the output pin [11].

**Switching power model** is the energy consumption model when the input combinations are changed and the output voltage change is also observed [11].

## 3.4 Characterization of NAND Gate

The characterization process in the Cadence Liberate tool for a NAND gate is described in this section. Since all the static gate has an almost similar characterization process in the Cadence Liberate, only for NAND gate has been illustrated. At first, templates and arcs need to be defined. It is possible to generate the templates by the tool by putting the value of minimum input pin capacitance, and minimum and maximum output transition. In general, the template consists of index 1 and index 2 where index 1 refers to the input transition and index 2 refers to the output load capacitance. Arcs can be defined by applying vectors in different pins of the logic gate and there are multiple types of arcs. Delay calculation, hidden power, and switching power can be calculated by putting separate arcs in the logic gate. Appendix A shows the script for the generation of templates and arcs from an existing standard cell library.

## 3.5 Liberty(.lib) File

The .lib file is an ASCII representation of all the standard cell's timing and power parameters of certain semiconductor technology. Simulating the cells under a range of situations yields the timing and power parameters, and the data is saved in the .lib format. A group statement is a grouping of other statements, as demonstrated below.

- Library group.
- Cell group.
- Pin group.
- Arc group.

```
library (Stacked_Lib) {
  comment : "";
  delay_model : table_lookup;
  capacitive_load_unit (1,pf);
  current_unit : "1mA";
  leakage_power_unit : "1pW";
  pulling_resistance_unit : "1kohm";
  time_unit : "1ns";
  voltage_unit : "1V";
  voltage_map (VDD, 1.8);
  voltage_map (GND, 0);
  voltage_map (VSS, 0);
  default_cell_leakage_power : 0;
  default_fanout_load : 1;
  default_max_transition : 6.1158;
  default_output_pin_cap : 0;
  in_place_swap_mode : match_footprint;
  input_threshold_pct_fall : 50;
  input_threshold_pct_rise : 50;
  nom_process : 1;
  nom_temperature : 37;
  nom_voltage : 1.8;
  output_threshold_pct_fall : 50;
  output_threshold_pct_rise : 50;
  slew_derate_from_library : 1;
  slew_lower_threshold_pct_fall : 20;
  slew_lower_threshold_pct_rise : 20;
  slew_upper_threshold_pct_fall : 80;
  slew_upper_threshold_pct_rise : 80;
  operating_conditions (PVT_1P8V_37C) {
    process : 1;
    temperature : 37;
```

Figure 3.3: Library attribute snippet of liberty file

A group of timing and power 'arcs' is called an **arc group**, a group of 'arcs' is called a **pin group**, a group of 'pins' is called a **cell group**, and a group of 'cells' is called a **library group**.
There are different type of statements in a liberty file. They are:

- Group statement.
- Attribute statement.
- Define statement.

The structure of generated liberty file is described in brief in the next section.

### 3.5.1 liberty file for inverter cell

In the begining of a liberty file, all the library attributes are defined as shown in Fig. 3.3. Generally all the units, threshold percentage points for measuring the delays, default capacitance values and operating condition are defined in this section.
In the cell group, cell attbributes for example: area are described in the begining. A leakage power group is described under the cell group similar to Fig. 3.4 which is a leakage group for the inverter. Leakage power values are written with respect to the input

```
leakage_power () {
  value : 0.189258;
  when : "!A*Q";
  related_pg_pin : GND;
}
leakage_power () {
  value : 0.00498837;
  when : "A*!Q";
  related_pg_pin : GND;
}
```

Figure 3.4: Leakage power group snippet of liberty file

combination of the gate. Output pin group of an inverter contains timing arcs for example cell rise, cell fall, rise transition, and fall transition. Figure 3.5 shows the timing arcs for the inverter where index 1 and index 2 are the input slew and output load respectively. Depending on index 1 and index 2, all the values are calculated and written under the 'values'.

Figure 3.6 represents the internal power group in the liberty file for an inverter. There are two types of internal power recorded which are rise power and fall power. They are also calculated depending on the value of index 1 and index 2.

```
pin (Q) {
  direction : output;
  function : "!A";
  power_down_function : "(!VDD) + (GND)";
  related_ground_pin : GND;
  related_power_pin : VDD;
  max_capacitance : 0.3105;
  timing () {
    related_pin : "A";
    timing_sense : negative_unate;
    timing_type : combinational;
    cell_rise (TIMING_TEMP_206_2D) {
      index_1 ("0.0114, 0.2022, 0.393, 0.7746, 1.5372, 3.0636, 6.1158");
      index_2 ("0.001, 0.010575, 0.02025, 0.0396, 0.0783, 0.1557, 0.3105");
      values ( \
        "0.3487, 0.580817, 0.795575, 1.22168, 2.07068, 3.76601, 7.15529", \
        "0.371406, 0.61737, 0.845557, 1.28481, 2.14366, 3.84542, 7.23837", \
        "0.442575, 0.683414, 0.912429, 1.35573, 2.22058, 3.92723, 7.32282", \
        "0.597517, 0.841487, 1.06735, 1.51009, 2.37903, 4.09158, 7.49207", \
        "0.823359, 1.16587, 1.40346, 1.84116, 2.70811, 4.42091, 7.83161", \
        "1.14134, 1.65212, 2.00813, 2.53241, 3.39353, 5.10522, 8.51517", \
        "1.62455, 2.35561, 2.88117, 3.66697, 4.78381, 6.49342, 9.90644" \
      );
    }
    rise_transition (TIMING_TEMP_206_2D) {
      index_1 ("0.0114, 0.2022, 0.393, 0.7746, 1.5372, 3.0636, 6.1158");
      index_2 ("0.001, 0.010575, 0.02025, 0.0396, 0.0783, 0.1557, 0.3105");
      values ( \
        "0.233087, 0.502891, 0.777332, 1.32613, 2.42322, 4.61755, 9.0061", \
        "0.235387, 0.501572, 0.775851, 1.32522, 2.42161, 4.61767, 9.00586", \
        "0.238465, 0.501443, 0.775051, 1.32327, 2.42095, 4.61509, 9.00546", \
        "0.286851, 0.509354, 0.776255, 1.32374, 2.42015, 4.61448, 9.00468", \
        "0.394641, 0.622364, 0.831743, 1.32996, 2.42182, 4.61593, 9.00413", \
        "0.531142, 0.892839, 1.13587, 1.51199, 2.45423, 4.61517, 9.00416", \
        "0.754391, 1.27328, 1.62761, 2.13481, 2.8925, 4.70267, 9.00407" \
      );
    }
```

Figure 3.5: Output pin group snippet of liberty file

```
internal_power () {
  related_pin : "A";
  related_pg_pin : VDD;
  rise_power (INTERNAL_POWER_TEMP_205_2D) {
    index_1 ("0.0114, 0.2022, 0.393, 0.7746, 1.5372, 3.0636, 6.1158");
    index_2 ("0.001, 0.010575, 0.02025, 0.0396, 0.0783, 0.1557, 0.3105");
    values ( \
      "0.0665019, 0.0677724, 0.0678781, 0.0678544, 0.0678321, 0.0677382, 0.0681784", \
      "0.061825, 0.0639841, 0.0652042, 0.0660944, 0.0667944, 0.067389, 0.0679497", \
      "0.0603575, 0.0622176, 0.063345, 0.0648341, 0.0659207, 0.0667807, 0.0677164", \
      "0.0595838, 0.0608126, 0.0616879, 0.0630093, 0.0643282, 0.0658267, 0.067175", \
      "0.0594637, 0.0600829, 0.0604786, 0.0614523, 0.0628026, 0.0641698, 0.066176", \
      "0.0603511, 0.0604418, 0.0605208, 0.0606613, 0.0614223, 0.0627056, 0.064627", \
      "0.0633755, 0.0628814, 0.0623962, 0.0620708, 0.0616811, 0.0611078, 0.0634018" \
    );
  }
  fall_power (INTERNAL_POWER_TEMP_205_2D) {
    index_1 ("0.0114, 0.2022, 0.393, 0.7746, 1.5372, 3.0636, 6.1158");
    index_2 ("0.001, 0.010575, 0.02025, 0.0396, 0.0783, 0.1557, 0.3105");
    values ( \
      "-0.0223731, -0.0218126, -0.0217889, -0.0216537, -0.02178, -0.0217218, -0.0217261", \
      "-0.0270668, -0.0250011, -0.0240579, -0.02322, -0.0224859, -0.0220954, -0.0218785", \
      "-0.0295114, -0.0270187, -0.0256739, -0.0242927, -0.0232266, -0.0225221, -0.0221797", \
      "-0.0319794, -0.0296675, -0.0280315, -0.0262465, -0.0245639, -0.0233311, -0.0225972", \
      "-0.0336356, -0.0319989, -0.0306726, -0.0287508, -0.0265931, -0.024651, -0.0233977", \
      "-0.0340026, -0.0333139, -0.0324843, -0.0311424, -0.0289794, -0.0266648, -0.0247089", \
      "-0.031775, -0.0320172, -0.0319597, -0.0316169, -0.0305184, -0.0286242, -0.0264542" \
    );
```

Figure 3.6: Internal power group snippet of liberty file

# 4 Simulation Result with Testbench

This chapter describes the simulation results of the synthesis of a Verilog module with the generated standard cell library.

## 4.1 Testbench Setup

A 4-bit Wallace tree multiplier has been used as a testbench for synthesizing with the generated stacked standard cell library and Dynamic Leakage Suppression (DLS) cell library. A Wallace multiplier is a hardware version of a binary multiplier, which is a digital circuit for multiplying two numbers. It sums partial products in stages using a variety of full and half adders until only two numbers remain [12]. The Verilog implementation of the Wallace multiplier is shown in appendix D. Figure 4.1 and 4.2 show the synthesis report snippet for stacked library and DLS library respectively.
Table 4.1 shows the comparison among the stacked, DLS, and conventional standard cell libraries after synthesizing the Wallace multiplier Verilog implementation. For a 4 bit Wallace tree multiplier, the DLS library consumes the least static power whereas the conventional library consumes almost 40 times more static power than the DLS library. DLS logic cells have very high propagation delay limiting the frequency to a few Hz.

Table 4.1: Comparison of standard cell library

| | Leakage Power (pW) | Combinational Delay (ps) | Internal Power (nW) | Area ($\mu m^2$) |
|---|---|---|---|---|
| **Stacked Library** | 18.628 | 52090 | 76421.971 | 3192.213 |
| **DLS Library** | 1.379 | 3.212e+11 | 29277.056 | 2714.835 |
| **Conventional Library** | 69.695 | 8189 | 17510.043 | 744.422 |

| Gate | Instances | Area | Leakage Power (pW) | Internal Power (pW) | Library |
|---|---|---|---|---|---|
| INHDLLX0 | 55 | 737.352 | 5.335 | 24093336.319 | Stacked_Lib |
| NA2HDLLX0 | 41 | 977.178 | 7.093 | 22340854.810 | Stacked_Lib |
| NO2HDLLX0 | 62 | 1477.683 | 6.200 | 29987780.193 | Stacked_Lib |
| total | 158 | 3192.213 | 18.628 | 76421971.322 | |

| Type | Instances | Area | Area % | Leakage Power (pW) | Leakage Power % | Internal Power (pW) | Internal Power % |
|---|---|---|---|---|---|---|---|
| inverter | 55 | 737.352 | 23.1 | 5.335 | 28.6 | 24093336.319 | 31.5 |
| logic | 103 | 2454.861 | 76.9 | 13.293 | 71.4 | 52328635.003 | 68.5 |
| physical_cells | 0 | 0.000 | 0.0 | 0.000 | 0.0 | 0.000 | 0.0 |
| total | 158 | 3192.213 | 100.0 | 18.628 | 100.0 | 76421971.322 | 100.0 |

Figure 4.1: Synthesis report snippet for stacked library

| Gate | Instances | Area | Leakage Power (pW) | Internal Power (pW) | Library |
|---|---|---|---|---|---|
| INHDLLX0 | 41 | 636.451 | 0.369 | 6982069.959 | DLS |
| NA2HDLLX0 | 67 | 1271.178 | 0.737 | 14475950.428 | DLS |
| NO2HDLLX0 | 39 | 807.206 | 0.273 | 7819035.947 | DLS |
| total | 147 | 2714.835 | 1.379 | 29277056.334 | |

| Type | Instances | Area | Area % | Leakage Power (pW) | Leakage Power % | Internal Power (pW) | Internal Power % |
|---|---|---|---|---|---|---|---|
| inverter | 41 | 636.451 | 23.4 | 0.369 | 26.8 | 6982069.959 | 23.8 |
| logic | 106 | 2078.384 | 76.6 | 1.010 | 73.2 | 22294986.375 | 76.2 |
| physical_cells | 0 | 0.000 | 0.0 | 0.000 | 0.0 | 0.000 | 0.0 |
| total | 147 | 2714.835 | 100.0 | 1.379 | 100.0 | 29277056.334 | 100.0 |

Figure 4.2: Synthesis report snippet for DLS library

# 5 Conclusion

The first phase of the logic design was comprised of the study of the leakage current reduction techniques and finally, two approaches which are Stacked logic gate, and Dynamic logic suppression were described in chapter 2. The last phase of the standard cell library design was about the characterization of the logic gates using the Cadence Liberate tool described in chapter 3.
It was observed that Dynamic Logic Suppression (DLS) logic cells were very slow compared to stacked or high-performance logic gates delivered by XFAB. However, the main advantage is the significant leakage reduction. In the future, a study on triple-well transistors on DLS logic can be conducted which may assist remarkably in increasing the speed of the logic while having a very low leakage current.

# Bibliography


[1] X. Chen and N. A. Touba, "Chapter 2 - fundamentals of cmos design," in *Electronic Design Automation*, L.-T. Wang, Y.-W. Chang, and K.-T. T. Cheng, Eds. Boston: Morgan Kaufmann, 2009, pp. 39–95. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780123743640500096

[2] M.-B. Lin, *Introduction to VLSI Systems: A Logic, Circuit, and System Perspective*. CRC Press, 2011.

[3] A. Jambek, A. NoorBeg, and M. Ahmad, "Standard cell library development," in *ICM'99. Proceedings. Eleventh International Conference on Microelectronics (IEEE Cat. No.99EX388)*, 1999, pp. 161–163.

[4] K. Abbas, *Handbook of Digital CMOS Technology, Circuits, and Systems*. Cham: Springer International Publishing, 2020, pp. 81–109. [Online]. Available: https://doi.org/10.1007/978-3-030-37195-1_2

[5] S. Friedrichs, M. Függer, and C. Lenzen, "Metastability-containing circuits," *IEEE Transactions on Computers*, vol. 67, no. 8, pp. 1167–1183, 2018.

[6] C. Hu, "Bsim model for circuit design using advanced technologies," in *2001 Symposium on VLSI Circuits. Digest of Technical Papers (IEEE Cat. No.01CH37185)*, 2001, pp. 5–10.

[7] M. M. Khademi, A. Arjomand, and H. Saedpanah, "Dynamic calculation of leakage current and electric field of distribution polymeric insulator under pollution layer," in *2015 20th Conference on Electrical Power Distribution Networks Conference (EPDC)*, 2015, pp. 173–178.

[8] M. Moradinezhad Maryan, M. Amini-Valashani, and S. Azhari, "A new circuit-level technique for leakage and short-circuit power reduction of static logic gates in 22-nm cmos technology," in *Circuits Syst Signal Process*, vol. 40, 2021, pp. 3536–3560.

[9] D. Bol, R. Ambroise, D. Flandre, and J.-D. Legat, "Building ultra-low-power low-frequency digital circuits with high-speed devices," in *2007 14th IEEE International Conference on Electronics, Circuits and Systems*, 2007, pp. 1404–1407.

[10] W. Lim, I. Lee, D. Sylvester, and D. Blaauw, "8.2 batteryless sub-nw cortex-m0+ processor with dynamic leakage-suppression logic," in *2015 IEEE International Solid-State Circuits Conference - (ISSCC) Digest of Technical Papers*, 2015, pp. 1–3.

[11] I. Keller, K. H. Tam, and V. Kariat, "Challenges in gate level modeling for delay and si at 65nm and below," in *2008 45th ACM/IEEE Design Automation Conference*, 2008, pp. 468–473.

[12] N. Sureka, R. Porselvi, and K. Kumuthapriya, "An efficient high speed wallace tree multiplier," in *2013 International Conference on Information Communication and Embedded Systems (ICICES)*, 2013, pp. 1023–1026.

# A Appendix

Script for generating template and arc:

```
set rundir $env(PWD)
set_operating_condition -voltage 1.8 -temp 37

read_library ${rundir}/.xkit/.../D_CELLS_HDLL_LPMOS_typ_1_80V_25C.lib

#read_library ${rundir}/liberty_ex/LIBRARY/inver.lib

write_template -verbose -use_lu_table_name -dir ↩
    ${rundir}/liberty_ex/templates define_tmp.tcl
write_userdata_library ${rundir}/liberty_ex/user_data/user_data.lib
write_datasheet -format html -dir ${rundir}/liberty_ex/user_data "data_cap"
```

Result of running arc generation script for NAND gate:

```
define_cell \
      -input { a b } \
      -output { out } \
      -pinlist { a b out } \
      -delay delay_template_7x7 \
      -power power_template_7x7 \
      NANDX1

define_leakage -when "!a*b*out" NANDX1
define_leakage -when "a*!b*out" NANDX1
define_leakage -when "!a*!b*out" NANDX1
define_leakage -when "a*b*!out" NANDX1

# power arcs from => a hidden
define_arc \
      -type hidden \
```

```
17        -when "(!b * out)" \
18        -vector {Rxx} \
19        -pin a \
20        NANDX1
21
22  # power arcs from => a hidden
23  define_arc \
24        -type hidden \
25        -when "(!b * out)" \
26        -vector {Fxx} \
27        -pin a \
28        NANDX1
29
30  # power arcs from => b hidden
31  define_arc \
32        -type hidden \
33        -when "(!a * out)" \
34        -vector {xRx} \
35        -pin b \
36        NANDX1
37
38  # power arcs from => b hidden
39  define_arc \
40        -type hidden \
41        -when "(!a * out)" \
42        -vector {xFx} \
43        -pin b \
44        NANDX1
45
46  # delay arcs from a => out negative_unate combinational
47  define_arc \
48        -vector {FxR} \
49        -related_pin a \
50        -pin out \
51        NANDX1
52
53  # delay arcs from a => out negative_unate combinational
54  define_arc \
```

```
55        -vector {RxF} \
56        -related_pin a \
57        -pin out \
58        NANDX1
59
60  # delay arcs from b => out negative_unate combinational
61  define_arc \
62        -vector {xFR} \
63        -related_pin b \
64        -pin out \
65        NANDX1
66
67  # delay arcs from b => out negative_unate combinational
68  define_arc \
69        -vector {xRF} \
70        -related_pin b \
71        -pin out \
72        NANDX1
73  }
```

# B Appendix

Script for characterization:

```
set rundir $env(PWD)
# Create the directories Liberate will write to.

exec mkdir -p ${rundir}/liberty_ex/LDB
exec mkdir -p ${rundir}/liberty_ex/LIBRARY
exec mkdir -p ${rundir}/liberty_ex/DATASHEET
exec mkdir -p ${rundir}/liberty_ex/VERILOG
set_operating_condition -voltage 1.8 -temp 37

set_var extsim_cmd_option "+spice +lorder MMSIM:PRODUCT"
set_var extsim_deck_header "simulator \lang=spectre\nOpt1 options ↩
    reltol=1e-4\nsimulator \lang=spice"
set_var extsim_leakage_option "method=gear gmin=1e-20 ↩
    redefinedparams=ignore rabsshort=1m"
set_var extsim_option "method=gear gmin=1e-20 redefinedparams=ignore ↩
    rabsshort=1m"
set_var extsim_tran_append "lteratio=10"
set_var extsim_reuse_ic 3

set_var extsim_save_passed all
set_var extsim_deck_dir ↩
    "home/et5/cll2046/Cadence/Standard_Cell/liberty_ex/DATASHEET/"

set_var slew_lower_rise 0.2
set_var slew_lower_fall 0.2
set_var slew_upper_rise 0.8
set_var slew_upper_fall 0.8

set_var measure_slew_lower_rise 0.2
```

```
26 set_var measure_slew_lower_fall 0.2
27 set_var measure_slew_upper_rise 0.8
28 set_var measure_slew_upper_fall 0.8
29
30 set_var delay_inp_rise 0.5
31 set_var delay_inp_fall 0.5
32 set_var delay_out_rise 0.5
33 set_var delay_out_fall 0.5
34
35 set_var pin_based_leakage 0
36 set_var pin_based_power 0
37 set_var def_arc_msg_level 0
38 set_var process_match_pins_to_ports 1
39 set_var max_transition 6.1158e-09
40 set_var min_transition 1.14e-11
41 set_var min_output_cap 1e-15
42 set_units -leakage_power 1pw
43
44 source ${rundir}/liberty_ex/TEMPLATE/INHDLLX0_template.tcl
45 read_spice -format spectre ${rundir}/liberty_ex/NETLIST/DLS/INHDLLX0.scs
46
47 #Models
48 read_spice -format spectre ↩
      /home/et5/cll2046/Cadence/Standard_Cell/liberty_ex/MODELS/model_files.scs
49 read_spice -format spectre ↩
      /designtools/eda/cadence/kits/.../v8_1_3/lpmos/models/mos/tm_mos.scs
50 read_spice -format spectre ↩
      /designtools/eda/cadence/kits/.../v8_1_3/lpmos/models/mos/pe.scs
51 read_spice -format spectre ↩
      /designtools/eda/cadence/kits/.../lpmos/models/mos/pel.scs
52 read_spice -format spectre ↩
      /home/et5/cll2046/Cadence/Standard_Cell/liberty_ex/MODELS/ne.scs
53 read_spice -format spectre ↩
      /home/et5/cll2046/Cadence/Standard_Cell/liberty_ex/MODELS/nel.scs
54
55
56 char_library -user_arcs_only -extsim spectre
57
```

```
58 #Output Files
59 write_ldb ${rundir}/liberty_ex/LDB/INHDLLX0.ldb
60 write_library -overwrite ${rundir}/liberty_ex/LIBRARY/INHDLLX0.lib
61 write_datasheet -format html -dir ${rundir}/liberty_ex/DATASHEET "Lib"
62 write_verilog ${rundir}/liberty_ex/VERILOG/INHDLLX0.v
```

# C Appendix

This is an example of generated verilog code after characterization by the tool.

```
1  // type:
2  `timescale 1ns/10ps
3  `celldefine
4  module DFFHDLLX0 (Q, QN, D, CN);
5    output Q, QN;
6    input D, CN;
7    reg notifier;
8    wire delayed_D, delayed_CN;
9
10   // Function
11   wire int_fwire_clk, int_fwire_IQ, int_fwire_IQN;
12   wire xcr_0;
13
14   not (int_fwire_clk, delayed_CN);
15   altos_dff_err (xcr_0, int_fwire_clk, delayed_D);
16   altos_dff (int_fwire_IQ, notifier, int_fwire_clk, delayed_D, xcr_0);
17   buf (Q, int_fwire_IQ);
18   not (int_fwire_IQN, int_fwire_IQ);
19   buf (QN, int_fwire_IQN);
20
21   // Timing
22   specify
23     (negedge CN => (Q+:D)) = 0;
24     (negedge CN => (QN-:D)) = 0;
25     $setuphold (negedge CN, posedge D, 0, 0, notifier,,, delayed_CN, ↩
         delayed_D);
26     $setuphold (negedge CN, negedge D, 0, 0, notifier,,, delayed_CN, ↩
         delayed_D);
27     $width (posedge CN, 0, 0, notifier);
```

```
28      $width (negedge CN, 0, 0, notifier);
29    endspecify
30  endmodule
31  `endcelldefine
```

Following datasheet were generated during characterization of stacked D flip flop and Dynamic Suppression Logic (DLS) NAND gate respectively.

# DFFHDLLX0

***Lib Cell Library: Process , Voltage 1.80, Temp 37.00***

## Truth Table

| INPUT | | OUTPUT | |
|---|---|---|---|
| D | CN | Q | QN |
| 0 | F | 0 | 1 |
| 1 | F | 1 | 0 |
| x | x | IQ | IQN |

## Footprint

| Cell Name | Area |
|---|---|
| DFFHDLLX0 | 137.5752 |

## Pin Capacitance Information

| Cell Name | Pin Cap(pf) | | Max Cap(pf) | |
|---|---|---|---|---|
| | D | CN | Q | QN |
| DFFHDLLX0 | 0.01244 | 0.02393 | 0.38000 | 0.38000 |

## Leakage Information

| Cell Name | Leakage(pW) | | |
|---|---|---|---|
| | Min. | Avg | Max. |
| DFFHDLLX0 | 1.15932 | 1.40062 | 1.52408 |

## Delay Information

**Delay(ns) to Q rising :**

| Cell Name | Timing Arc(Dir) | Delay(ns) | | |
|---|---|---|---|---|
| | | First | Mid | Last |
| DFFHDLLX0 | CN->Q (FR) | 2.69221 | 4.19029 | 13.81190 |

**Delay(ns) to Q falling :**

| Cell Name | Timing Arc(Dir) | Delay(ns) | | |
|---|---|---|---|---|
| | | First | Mid | Last |
| DFFHDLLX0 | CN->Q (FF) | 4.01473 | 5.08818 | 10.43290 |

**Delay(ns) to QN rising :**

| Cell Name | Timing Arc(Dir) | Delay(ns) | | |
|---|---|---|---|---|
| | | First | Mid | Last |
| DFFHDLLX0 | CN->QN (FR) | 4.65002 | 6.03485 | 15.64230 |

**Delay(ns) to QN falling :**

| Cell Name | Timing Arc(Dir) | Delay(ns) | | |
|---|---|---|---|---|
| | | First | Mid | Last |
| DFFHDLLX0 | CN->QN (FF) | 3.25978 | 4.08746 | 9.29156 |

## Constraint Information

**Constraints(ns) for D rising :**

| Cell Name | Timing Check | Ref Pin(trans) | Reference Slew Rate(ns) | | |
|---|---|---|---|---|---|
| | | | first | mid | last |
| DFFHDLLX0 | hold | CN (F) | 1.53651 | 2.39440 | 3.55151 |
| | setup | CN (F) | -0.67530 | -1.65484 | -2.83091 |

**Constraints(ns) for D falling :**

| Cell Name | Timing Check | Ref Pin(trans) | Reference Slew Rate(ns) | | |
|---|---|---|---|---|---|
| | | | first | mid | last |
| DFFHDLLX0 | hold | CN (F) | -0.31945 | -0.16177 | 0.14498 |
| | setup | CN (F) | 0.36340 | 0.23025 | -0.04621 |

**Constraints(ns) for CN rising :**

| Cell Name | Timing Check | Ref Pin(trans) | Reference Slew Rate(ns) | | |
|---|---|---|---|---|---|
| | | | first | mid | last |
| DFFHDLLX0 | min_pulse_width | CN () | 1.28232 | 5.05615 | 10.09160 |

**Constraints(ns) for CN falling :**

| Cell Name | Timing Check | Ref Pin(trans) | Reference Slew Rate(ns) | | |
|---|---|---|---|---|---|
| | | | first | mid | last |
| DFFHDLLX0 | min_pulse_width | CN () | 2.71647 | 5.05615 | 10.09160 |

## Power Information

**Internal switching power(pJ) to Q rising :**

| Cell Name | Input | Power(pJ) | | |
|---|---|---|---|---|
| | | first | mid | last |
| DFFHDLLX0 | CN | 0.17886 | 0.18155 | 0.18239 |

**Internal switching power(pJ) to Q falling :**

| Cell Name | Input | Power(pJ) | | |
|---|---|---|---|---|
| | | first | mid | last |
| DFFHDLLX0 | CN | 0.18018 | 0.18268 | 0.18376 |

**Internal switching power(pJ) to QN rising :**

| Cell Name | Input | Power(pJ) | | |
|---|---|---|---|---|
| | | first | mid | last |
| DFFHDLLX0 | CN | 0.18050 | 0.18314 | 0.18453 |

**Internal switching power(pJ) to QN falling :**

| Cell Name | Input | Power(pJ) | | |
|---|---|---|---|---|
| | | first | mid | last |
| DFFHDLLX0 | CN | 0.17859 | 0.18106 | 0.18144 |

**Passive power(pJ) for D rising :**

| Cell Name | Power(pJ) | | |
|---|---|---|---|
| | first | mid | last |
| DFFHDLLX0 | 0.07782 | 0.07351 | 0.07667 |

**Passive power(pJ) for D falling :**

| Cell Name | Power(pJ) | | |
|---|---|---|---|
| | first | mid | last |
| DFFHDLLX0 | 0.13302 | 0.12994 | 0.13418 |

**Passive power(pJ) for CN rising :**

| Cell Name | Power(pJ) | | |
|---|---|---|---|
| | first | mid | last |
| DFFHDLLX0 | 0.10012 | 0.09480 | 0.08825 |

**Passive power(pJ) for CN falling :**

| Cell Name | Power(pJ) | | |
|---|---|---|---|
| | first | mid | last |
| DFFHDLLX0 | 0.17507 | 0.17082 | 0.15737 |

# NA2HDLLX0

***Lib Cell Library: Process , Voltage 1.80, Temp 37.00***

## Truth Table

| INPUT | | OUTPUT |
|---|---|---|
| A | B | Q |
| 0 | x | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

## Footprint

| Cell Name | Area |
|---|---|
| NA2HDLLX0 | 18.9728 |

## Pin Capacitance Information

| Cell Name | Pin Cap(pf) | | Max Cap(pf) |
|---|---|---|---|
| | A | B | Q |
| NA2HDLLX0 | 0.00319 | 0.00311 | 0.02000 |

## Leakage Information

| Cell Name | Leakage(pW) | | |
|---|---|---|---|
| | Min. | Avg | Max. |
| NA2HDLLX0 | 0.00263 | 0.01142 | 0.01598 |

## Delay Information

**Delay(ns) to Q rising :**

| Cell Name | Timing Arc(Dir) | Delay(ns) | | |
|---|---|---|---|---|
| | | First | Mid | Last |
| NA2HDLLX0 | A->Q (FR) | 879095.00000 | 2413310.00000 | 5140990.00000 |
| | B->Q (FR) | 1080290.00000 | 2559900.00000 | 5329180.00000 |

**Delay(ns) to Q falling :**

| Cell Name | Timing Arc(Dir) | Delay(ns) | | |
|---|---|---|---|---|
| | | First | Mid | Last |
| NA2HDLLX0 | A->Q (RF) | 1293.38000 | 170629.00000 | 842387.00000 |
| | B->Q (RF) | 1297.56000 | 174637.00000 | 861409.00000 |

## Power Information

**Internal switching power(pJ) to Q rising :**

| Cell Name | Input | Power(pJ) | | |
|---|---|---|---|---|
| | | first | mid | last |
| NA2HDLLX0 | A | 0.01532 | 0.01407 | 0.01295 |
| | B | 0.01846 | 0.01716 | 0.01601 |

**Internal switching power(pJ) to Q falling :**

| Cell Name | Input | Power(pJ) | | |
|---|---|---|---|---|
| | | first | mid | last |
| NA2HDLLX0 | A | 0.00572 | 0.00668 | 0.00682 |
| | B | 0.00562 | 0.00663 | 0.00678 |

# D Appendix

Half adder:

```
module half_adder(
    Data_in_A,
    Data_in_B,
    Data_out_Sum,
    Data_out_Carry );

    //what are the input ports.
    input Data_in_A;
    input Data_in_B;
    //What are the output ports.
    output Data_out_Sum;
     output Data_out_Carry;
     //Implement the Sum and Carry equations using Verilog Bit operators.
     assign Data_out_Sum = Data_in_A ^ Data_in_B; //XOR operation
     assign Data_out_Carry = Data_in_A & Data_in_B; //AND operation
endmodule
```

Full adder:

```
module full_adder(
    Data_in_A, //input A
    Data_in_B, //input B
    Data_in_C, //input C
     Data_out_Sum,
     Data_out_Carry
    );
    //what are the input ports.
    input Data_in_A;
    input Data_in_B;
     input Data_in_C;
```

```
12      //What are the output ports.
13      output Data_out_Sum;
14       output Data_out_Carry;
15       //Internal variables
16       wire ha1_sum;
17       wire ha2_sum;
18       wire ha1_carry;
19       wire ha2_carry;
20       wire Data_out_Sum;
21       wire Data_out_Carry;
22
23       //Instantiate the half adder 1
24      half_adder ha1(
25          .Data_in_A(Data_in_A),
26          .Data_in_B(Data_in_B),
27          .Data_out_Sum(ha1_sum),
28          .Data_out_Carry(ha1_carry)
29      );
30
31      //Instantiate the half adder 2
32      half_adder ha2(
33          .Data_in_A(Data_in_C),
34          .Data_in_B(ha1_sum),
35          .Data_out_Sum(ha2_sum),
36          .Data_out_Carry(ha2_carry)
37      );
38      //sum output from 2nd half adder is connected to full adder output
39      assign Data_out_Sum = ha2_sum;
40      //The carry's from both the half adders are OR'ed to get the final ↩
            carry./
41      assign Data_out_Carry = ha1_carry | ha2_carry;
42  endmodule
```

Wallace Tree implementation:

```
1  module wallace (prod,a,b);
2  input [3:0] a,b;
3  output [7,0] prod;
4
```

```
5  wire ↩
       s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,c0,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11;
6  reg p [3:0][3:0];
7
8  integer i,j;
9  always @ (a or b)
10 begin
11  for (i=o;i<=3;i=i+1)
12       for(j=0;j<=3;j=j+1)
13           p[j][i]<=a[j] & b[i];
14 end
15
16 half_adder h1(s0,c0,p[0][1],p[1][0]);
17 full_adder f1(s1,c1,p[0][2],[1][1],p[2][0]);
18 full_adder f2(s2,c2,p[0][3],p[1][2],p[2][1]);
19 full_adder f3(s3,c3,p[1][3],p[2][2],1'b0);
20
21 full_adder f4(s4,c4,s1,c0,1'b0);
22 full_adder f5(s5,c5,s2,c1,p[3][0]);
23 full_adder f6(s6,c6,s3,c2,p[3][1]);
24 full_adder f7(s7,c7,p[2][3],c3,p[3][2]);
25
26 full_adder f8(s8,c8,s5,c4,1'b0);
27 full_adder f9(s9,c9,s6,c8,c5);
28 full_adder f10(s10,c10,s7,c6,c9);
29 full_adder f11(s11,c11,p[3][3],c7,c10);
30
31 assign prod[0]=p[0][0];
32 assign prod[1]=s0;
33 assign prod[2]=s4;
34 assign prod[3]=s8;
35 assign prod[4]=s9;
36 assign prod[5]=s10;
37 assign prod[6]=s11;
38 assign prod[7]=c11;
39 endmodule
```