A PROJECT REPORT ON

“DESIGN AND IMPLEMENTATION OF AN INVERTER CIRCUIT USING OPEN SOURCE PDK TOOLS : INTEGRATING VLSI THEORETICAL CONCEPTS AND VERILOG CODING"

Submitted by:

V. SAI CHARAN GUPTA

B Tech’25 IIIT BHUBANESWAR

ETC 3RD YEAR (B221065)

Under the guidance of:

Mr. PRASAD BORANNAVAR (Sc ‘F’)

For GD D-FCR

L.R.D.E D.R.D.O

DECLARATION

I, V. Sai Charan Gupta student of B. Tech belonging to IIIT Bhubaneswar, declare that this Project Report entitled “DESIGN AND IMPLEMENTATION OF AN INVERTER CIRCUIT USING OPEN SOURCE PDK TOOLS” is the Project work done by me under the supervision of Mr. Prasad Borannawar Sir.

I am submitting this Project Work Report in partial fulfillment of the requirements for the award of the Internship at L.R.D.E D.R.D.O, Bangalore during the academic year 2024.

(Signature of candidate)

Date:

Certified that this project work submitted by V. Sai Charan Gupta has been carried out under my guidance and the declaration made by the candidate is true to the best of my knowledge.

Signature of Guide **Signature of Director**

ACKNOWLEDGMENT

I would like to express my sincere gratitude to Mr. Prasad Borannawar Sir for their invaluable guidance, support, and expertise throughout the course of this project. Their insightful feedback and encouragement have been instrumental in shaping my understanding and skills in VLSI design.

I am incredibly grateful to Mrs. P. Rajini Mam for their unwavering belief in my abilities and their generous support that opened the doors to this internship opportunity. Their encouragement and mentorship have not only been invaluable but also profoundly inspiring, propelling my professional growth in ways I never imagined possible.

I am grateful to my parents and all members of my family for their constant encouragement given to me.

I also thank my friends for their cooperation.

Above all, I thank the almighty, for giving me the resource and strength to do this project.

ABSTRACT

This abstract provides an overview of the internship project report focused on the design and implementation of an inverter circuit using open-source Process Design Kit (PDK) tools. The project involves the complete workflow from schematic design through to layout verification. Initially, the circuit was designed using XSchem, followed by extensive simulations using Ng-Spice to verify its functionality under various conditions. Adjustments were made to key characteristics of the circuit based on simulation results to optimize performance.

Subsequently, the layout of the inverter circuit was created using Magic VLSI tool, ensuring adherence to design rules and functional correctness. Verification of the layout was conducted by analyzing the netlist output, confirming the physical realization matched the intended schematic design. This project highlights the practical application of VLSI design concepts and the effectiveness of open-source tools in achieving design objectives.

Additionally, this project provided an opportunity to learn the basics of VLSI theoretical concepts and Verilog coding. This foundational knowledge helped in gaining a better perspective of the theoretical concepts covered in my college curriculum, bridging the gap between academic learning and practical application.

TABLE OF CONTENTS

MODULE 1 : INTRODUCTION

What is VLSI ?

Very-Large-Scale Integration (VLSI) is a process used to create integrated circuits (IC's) by combining thousands to millions of transistors onto a single chip. This technology is essential in electronics and computer engineering because it allows for the development of complex electronic devices and systems that are more compact, faster, and energy-efficient. VLSI plays a pivotal role in advancing modern electronics, making it possible to produce high-performance and cost-effective products such as microprocessors, memory chips, and mobile devices.

By integrating a large number of transistors into a single chip, VLSI allows for the production of high-performance electronic components that drive modern technological advancements. This high level of integration leads to significant improvements in computational power, storage capacity, and overall device functionality, which are essential for contemporary electronics.

So there are 'DESIGN STEPS' in this VLSI Design to be followed to achieve the correct output as expected by the user.

VLSI Design Process :

The VLSI design process encompasses several critical stages, each contributing to the successful creation of an integrated circuit (let's see these steps with an example) :

1. **Specification**: This initial stage involves defining the requirements and functionalities of the chip, outlining what the IC needs to accomplish.

   Let's say, we need to get **Y** = **(a * b)** ^ **c**

2. **Design**: During this phase, engineers create a schematic representation of the circuit using hardware description languages (HDL's) such as Verilog or VHDL. This step translates the specified functionalities into a design blueprint.

So, the Verilog code for the above equation is :

VERILOG CODE :

```
module example(input a, b, c,
output y);
assign y = (a*b)^c;
endmodule
```

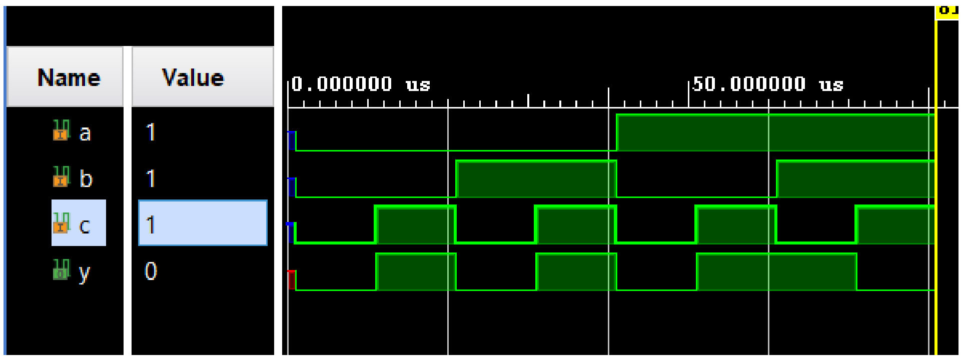
3 **Simulation**: The design undergoes rigorous testing through simulations to ensure it meets the specified requirements. Simulations help identify and rectify potential issues before proceeding to the next stage.

It includes some sub-steps that get executed :

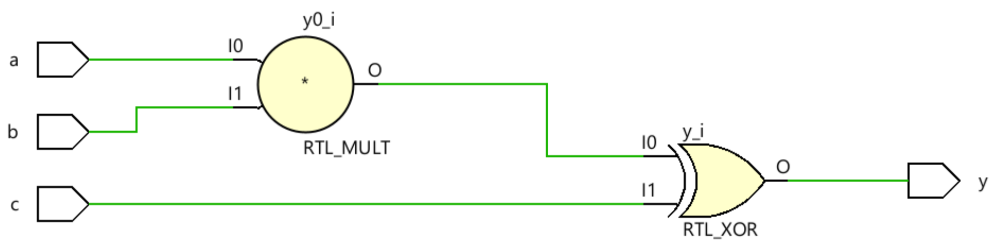
i. we can say as it executes our code & get the output by changing the force constants.

ii. Syntax check happens here and RTL analysis [i.e. The RTL design on the logic & functional description of the hardware (or) It directly maps the operations described in the Verilog code to logical components.

The Result we get in VIVADO :



Name	Value
a	1
b	1
c	1
y	0



4 **Synthesis**: This stage involves converting the high-level design into a gate-level representation, mapping the design onto actual hardware components.

(or) It also includes some sub-steps :

i. It helps in converting the RTL description into a gate level netlist, which is more detailed representation of circuit at the transistor level.

ii. Also optimizes the design for the specific target hardware, considering factors such as timing, area, & power consumption.

As we are going with the example here it undergoes the K-MAP simplification as we do in our theory :

The TRUTH TABLE is as follows :

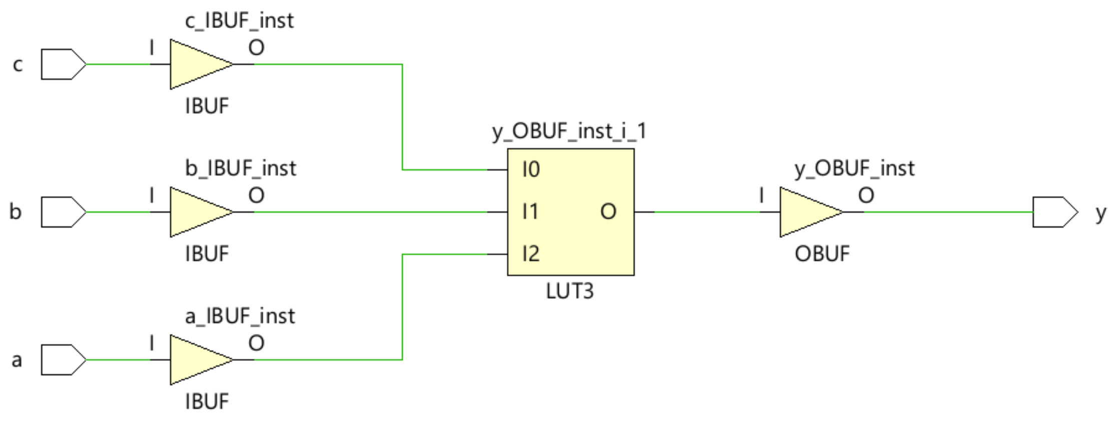
a	**b**	**c**	**a*b**	**(a*b)^c**	**Column No.**
0	0	0	0	0	0
0	0	1	0	**1**	**1**
0	1	0	0	0	2
0	0	1	0	**1**	**3**
1	0	0	0	0	4
1	0	1	0	**1**	**5**
1	1	0	1	**1**	**6**
1	1	1	1	0	7

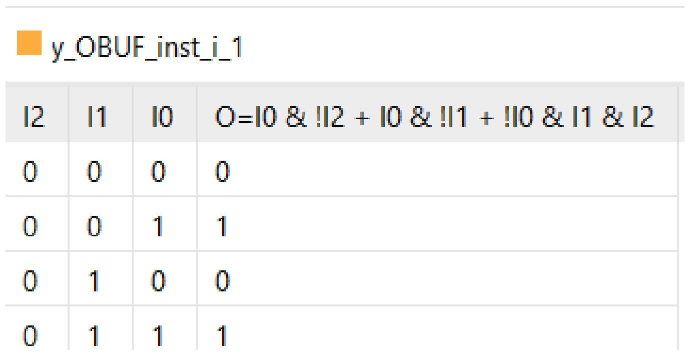
The K-MAP is as follows :

	(~b)(~c)	(~b)c	bc	b(~c)
~a		1	1	
a		1		1

The result is : Y = (~b)c + (~a)c + ab(~c)

After Implementing the same code in VIVADO we get the result as follows :





🟧 y_OBUF_inst_i_1

I2	I1	I0	O=I0 & !I2 + I0 & !I1 + !I0 & I1 & I2
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1

5 **Implementation:** It consists of various sub-steps :

i. Optimizes logic based on design constraints

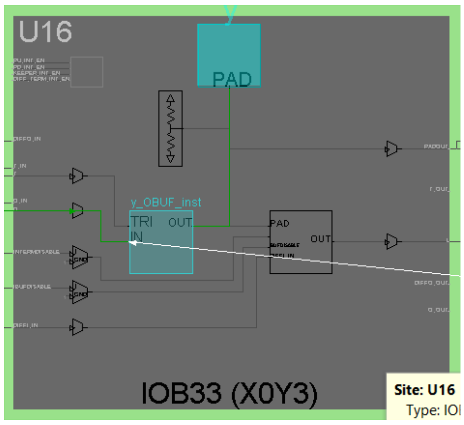
ii. Maps the synthesized netlist onto the resources available in the target hardware (eg. LUT's and flip-flops) in an FPGA.

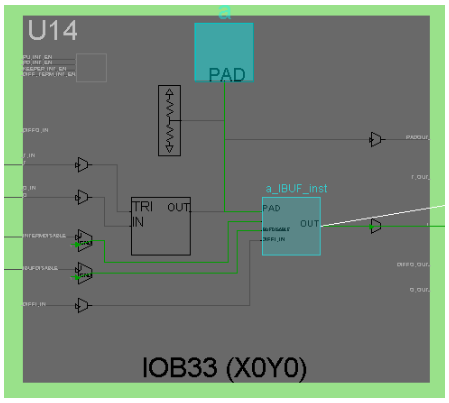
iii. Layout: Engineers then proceed to physically arrange the components on the chip, adhering to design rules and ensuring optimal performance and functionality which includes the placing and routing.

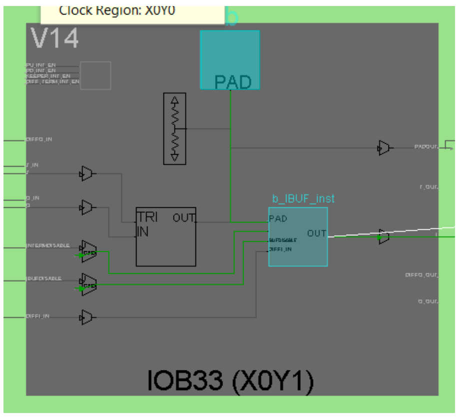
Placing : Determines physical location of each component on the FPGA (or) other hardware.

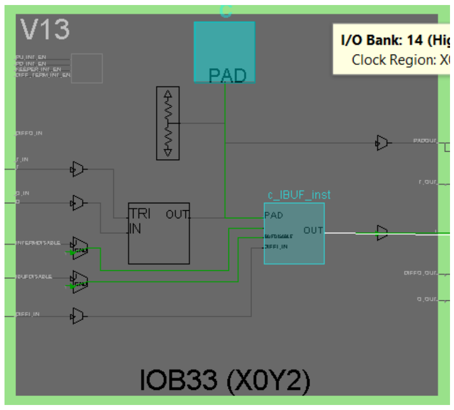
Routing : Establishes physical connection between the placed components.

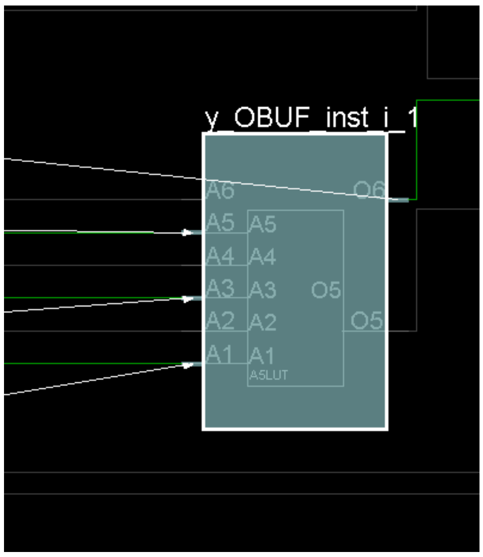
The Result we get in VIVADO :











6. **Fabrication**: The physical manufacturing of the IC is carried out using semiconductor technology, where the layout design is etched onto a silicon wafer.
7. **Testing**: Finally, the manufactured IC undergoes comprehensive testing to verify its performance and functionality, ensuring it operates as intended.

If we want to implement the same example we have taken above then we will just try to remove the last two steps mentioned above, and we will generate the Bitstream.

So what is a **Bitstream** ?

Bitstream contains all the data needed to implement a specific design on the FPGA board, i.e., it involves translating HDL to format that an FPGA can use to configure it's logic blocks.

*** After the implementation step we also perform the timing analysis which broadly includes **static timing analysis (STA)** and **path analysis.**

MODULE 2 : I. DESIGN

1. VLSI DESIGN TYPES

In Very Large Scale Integration (VLSI) design, several methodologies and types are used to create integrated circuits (IC's). Here are the primary types:

1. Full-Custom Design

- ***Description:*** In full-custom design, every part of the circuit is designed and optimized manually. This approach allows for maximum performance optimization and area utilization.
- ***Applications:*** High-performance microprocessors, specialized IC's in aerospace, and military applications.
- ***Advantages:*** Optimized for performance, power, and area.
- ***Disadvantages:*** Very time-consuming and costly; requires extensive design expertise.

2. Semi-Custom Design

Semi-custom design is a mix between full-custom and standard cell design methodologies.

2.1. Standard Cell Design

- ***Description:*** Uses pre-designed and pre-characterized cells (eg., logic gates, flip-flops) provided in a library. Designers focus on the placement and routing of these standard cells.
- ***Applications:*** General-purpose microprocessors, ASIC's.
- ***Advantages:*** Faster design time, more predictable performance.
- ***Disadvantages:*** Less optimized than full-custom design.

2.2. Gate Array Design

- ***Description:*** Uses a predefined array of transistors on a wafer, and only the top-level metal layers are customized to create the final circuit. This includes variants like Mask Programmable Gate Arrays (MPGA's) and Field Programmable Gate Arrays (FPGA's).
- ***Applications:*** Prototyping, low to medium volume production.

- ***Advantages:*** Reduced time to market, lower cost for low-volume production.
- ***Disadvantages:*** Less efficient in terms of performance and area compared to full-custom and standard cell designs.

3. Field Programmable Gate Array (FPGA) Design

- ***Description:*** Uses a pre-fabricated silicon device with programmable logic blocks and interconnects that can be configured after manufacturing.
- ***Applications:*** Prototyping, low-volume production, digital signal processing, telecommunications.
- ***Advantages:*** Reconfigurability, short time to market, cost-effective for low volumes.
- ***Disadvantages:*** Larger area, higher power consumption, and lower performance compared to ASIC's.

4. Application-Specific Integrated Circuit (ASIC) Design

- ***Description:*** Custom-designed chips tailored for specific applications.
- ***Applications:*** Consumer electronics, automotive electronics, network devices.
- ***Advantages:*** Optimized for specific applications, high performance, lower power consumption.
- ***Disadvantages:*** High non-recurring engineering (NRE) costs, longer design cycle.

5. Structured ASIC Design

- ***Description:*** A compromise between full-custom ASIC's and FPGA's. They use pre-defined logic and routing structures with a high degree of customization in the top metal layers.
- ***Applications:*** Cost-sensitive high-volume applications where FPGA performance is insufficient.
- ***Advantages:*** Faster design cycle than full-custom, better performance than FPGA's.
- ***Disadvantages:*** Less flexible than FPGA's, higher NRE costs than standard cells.

6. System on Chip (SoC) Design

- ***Description:*** Integrates all components of a computer or other electronic systems into a single chip, including processors, memory, interfaces, and analog components.
- ***Applications:*** Smartphones, tablets, embedded systems.
- ***Advantages:*** High integration, reduced power consumption, cost-effective for high volumes.
- ***Disadvantages:*** Complex design and verification, longer development time.

Commonly Used VLSI Design Types

- ***Standard Cell Design:*** Widely used due to the balance between design time, cost, and performance.
- ***FPGA Design:*** Popular for prototyping, low-volume production, and applications requiring frequent updates.
- ***ASIC Design:*** Common for high-performance and high-volume applications where optimization is critical.
- ***SoC Design:*** Increasingly prevalent due to the need for integration in mobile and embedded systems.

Each design type has its own set of trade-offs, making them suitable for different applications and design requirements.

1.1. DIFFERENCES BETWEEN ASIC & FPGA

As we can see in mostly used design types we have FPGA & ASIC, let’s see the differences between them in detail:

DIFFERENCES	ASIC	FPGA
1.Design and Flexibility	*Fixed Functionality:* ASIC’s are custom-designed for a specific application. Once manufactured, their functionality cannot be changed. *Design Process:* The design process involves creating a detailed circuit design, followed by fabrication in a semiconductor foundry. This process is complex, time-consuming, and expensive. *High Initial Cost:* The initial cost includes design, verification, and manufacturing setup (mask costs), which are very high.	*Programmable:* FPGA’s are designed to be programmable by the user. They consist of an array of configurable logic blocks (CLB’s) connected via programmable interconnects. *Flexibility:* FPGA’s can be reprogrammed to perform different functions even after deployment, making them highly flexible. *Lower Initial Cost:* The initial cost is lower compared to ASIC’s because there is no need for costly manufacturing processes. However, the per-unit cost is higher.
2. Performance	*High Performance:* ASIC’s typically offer better performance in terms of speed and power efficiency because they are optimized for a specific task. *Optimization:* Designers can optimize ASIC’s for power consumption, area, and performance, making them suitable for high-volume, performance-critical applications.	*Moderate Performance:* FPGA’s generally offer lower performance compared to ASIC’s due to the overhead of programmable interconnects and logic blocks. *Versatility:* Despite being slower, FPGA’s are versatile and can handle a wide range of applications, making them suitable for prototyping and low-to-medium volume production.

3. Cost and Economics	*Economies of Scale:* ASIC's are cost-effective for high-volume production due to lower per-unit cost after the initial design and setup expenses. *High NRE Costs:* Non-recurring engineering (NRE) costs are significant, covering design, testing, and manufacturing setup.	*Higher Per-Unit Cost:* FPGA's have a higher per-unit cost compared to ASIC's because they are general-purpose and include additional resources for programmability. *Low NRE Costs:* FPGA's have minimal NRE costs, making them more economical for low-volume production and prototyping.
4. Development Time:	*Longer Development Time:* The development cycle for an ASIC is lengthy, often taking several months to years due to the complexity of design, testing, and fabrication processes. *Detailed Verification:* Extensive verification and testing are required to ensure the design works correctly before committing to manufacturing.	*Shorter Development Time:* FPGA's have a shorter development cycle since they can be reprogrammed easily, allowing for rapid prototyping and testing. *Iterative Design:* Engineers can quickly iterate on the design, test it, and make changes as needed without waiting for new hardware to be manufactured.
5. Applications	*Specific Applications:* Used in applications where high performance, low power consumption, and high-volume production are critical. Examples include processors, custom SoCs for smartphones, and specialized hardware in automotive and aerospace industries.	*Varied Applications:* Used in applications requiring flexibility and reconfigurability, such as prototyping new designs, implementing digital signal processing algorithms, network hardware, and various industrial controls.

1.2. CONFIGURABLE LOGIC BLOCKS (CLB)

A Configurable Logic Block (CLB) is a fundamental building block within a Field-Programmable Gate Array (FPGA). CLB's are the primary components that provide the programmable logic capability of FPGA's. Here is a detailed explanation of what a CLB is and its components:

Components of a CLB:

1. Look-Up Tables (LUT's):

- LUT's are small memory blocks that can implement combinational logic. Each LUT can be programmed to realize any boolean function of its inputs.

- Typically, an LUT has 4 to 6 inputs, allowing it to represent complex logic functions.

2. Flip-Flops:

- Flip-flops are used to store binary data and implement sequential logic. Each flip-flop can store one bit of data.

- They work in conjunction with LUT's to build more complex circuits like state machines or registers.

3. Multiplexers:

- Multiplexers within CLB's allow for the selection of different data paths, enabling more flexible logic implementation.

4. Carry Chains:

- Carry chains are specialized circuits designed for efficient implementation of arithmetic functions, such as addition and subtraction.

5. Interconnects:

- CLB's are connected to each other and to other parts of the FPGA via a programmable interconnect network. This network allows for flexible routing of signals between different CLB's and other components.

Functionality of CLB’s :

Programmability: CLB’s can be programmed to perform a wide variety of logic functions. This programmability is achieved through the configuration of LUT’s,flip-flops, and interconnects.

Combinational and Sequential Logic: By using LUT’s for combinational logic and flip-flops for sequential logic, CLB’s can implement complex logic functions, including state machines, counters, and custom logic circuits.

Reconfigurability: One of the key advantages of FPGA’s is that CLB’s can be reconfigured even after the FPGA has been deployed. This allows for updates and modifications to the hardware functionality without needing new hardware.

Example of CLB Usage

Consider designing a simple digital circuit, such as a 4-bit binary counter, using an FPGA. Here's how CLB’s might be used:

1. Combinational Logic with LUT’s:

- The LUT’s in CLB’s can be programmed to implement the logic for each bit of the counter. For example, the logic for incrementing each bit in a binary counter can be mapped to the LUT’s.

2. Sequential Logic with Flip-Flops:

- The flip-flops will store the current state (i.e., the current value of the counter) and update their state based on the clock signal and the combinational logic outputs from the LUT’s.

3. Interconnections:

- The programmable interconnects will route the necessary signals between the LUT’s, flip-flops, and other CLB’s to ensure the correct operation of the counter.

Conclusion:

CLB’s are the versatile and reconfigurable units within FPGA’s that allow them to implement a wide range of digital logic functions. They consist of LUT’s for combinational logic, flip-flops for sequential logic, and interconnects for routing signals. This flexibility makes FPGA’s a powerful tool for prototyping, custom hardware design, and applications requiring reconfigurable logic.

MODULE 3 : SIMULATION, SYNTHESIS & IMPLEMENTATION

1. CLOCK AND TIMING GENERATION TECHNIQUES & PARAMETERS

As we know that for any VLSI design clock and timing parameters are the main ones which need to be carefully taken care of as many factors depend on it like performance, speed, etc, of a device or specifically IC.

So let’s first discuss

Clock Generation techniques :

1. Crystal Oscillators :

How They Work:

- **Crystal Resonance:** A crystal oscillator uses the mechanical resonance of a vibrating crystal (usually quartz) to generate a consistent frequency. When an electric field is applied to the crystal, it deforms. The crystal then generates an electric signal as it returns to its original shape.
- **Piezoelectric Effect:** Quartz exhibits the piezoelectric effect, meaning it can convert mechanical vibrations into electrical signals and vice versa.
- **Oscillation Circuit:** The crystal is part of an oscillation circuit, typically involving an amplifier and feedback loop that sustain the oscillations at the crystal's natural resonant frequency.

Process:

1. **Oscillator Circuit Initiation:** A small electric charge is applied to the crystal.
2. **Vibration:** The crystal starts to vibrate at its resonant frequency.
3. **Amplification:** The oscillating signal is fed back into the amplifier, which boosts the signal.
4. **Stable Output:** The feedback loop ensures continuous oscillation, providing a stable and precise frequency output.

***Applications**:* Common in micro-controllers, microprocessors, and other digital circuits requiring stable and accurate clocks.

***Advantages**:* High stability, accuracy, and low jitter.

Disadvantages: Larger size and higher cost compared to other methods.

2. Phase-Locked Loops (PLL) :

How They Work:

- **Phase Comparator:** Compares the phase of an input reference signal with the phase of the output signal.
- **Voltage-Controlled Oscillator (VCO):** Produces an output signal whose frequency is adjusted based on the voltage applied to it.
- **Low Pass Filter:** Filters the output of the phase comparator to produce a control voltage.
- **Feedback Loop:** Ensures the VCO's output frequency locks to the reference signal's frequency.

Process:

1. **Phase Detection:** The phase comparator detects the phase difference between the input reference signal and the VCO output.
2. **Control Voltage Generation:** This phase difference generates a control voltage through the low pass filter.
3. **Frequency Adjustment:** The control voltage adjusts the VCO, changing its frequency to reduce the phase difference.
4. **Lock:** The feedback loop continues adjusting until the VCO output is phase-locked to the reference signal, producing a stable clock signal.

***Applications**:* Widely used in CPUs, GPUs, communication systems, and any system needing clock multiplication or frequency synthesis.

***Advantages**:* High frequency accuracy, flexibility in generating multiple clock frequencies.

***Disadvantages**:* Complexity in design, power consumption, and potential for jitter.

3. Voltage-Controlled Oscillators (VCO) :

How They Work:

- **Frequency Control:** The frequency of the oscillation is controlled by an input voltage.
- **Oscillator Circuit:** Typically involves an LC circuit (inductor-capacitor) or a ring oscillator whose frequency is varied by changing the control voltage.

Process:

1. **Voltage Application:** An input voltage is applied to the VCO.
2. **Frequency Variation:** The applied voltage changes the characteristics of the LC circuit or the delay of the ring oscillator.
3. **Output Signal:** The VCO produces a clock signal whose frequency is proportional to the input control voltage.

Applications*:* Often used within PLL’s and frequency synthesizers.

Advantages*:* Frequency can be easily tuned, simple design.

Disadvantages*:* Frequency stability depends on the control voltage, susceptible to noise.

4. Digitally Controlled Oscillators (DCO) :

How They Work:

- **Digital Control:** Uses digital signals to control the frequency of the oscillator.
- **Frequency Synthesis:** Often implemented using a combination of digital logic and a VCO.

Process:

1. **Digital Input:** A digital control word is input to the DCO.
2. **Frequency Adjustment:** The DCO adjusts its output frequency based on the digital control signals.
3. **Output Signal:** The DCO generates a clock signal with a frequency corresponding to the digital input.

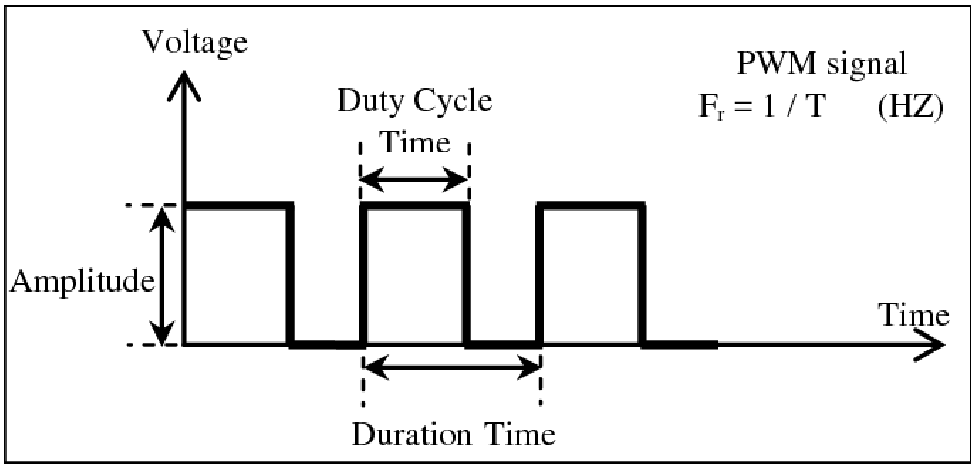
Applications*:* Digital PLL’s, modern digital systems.

Advantages*:* High frequency accuracy, easy to integrate with digital systems.

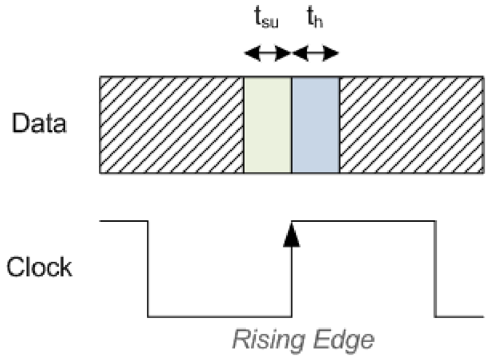
Disadvantages*:* Digital control can introduce quantization noise.

2. CLOCK PARAMETERS :

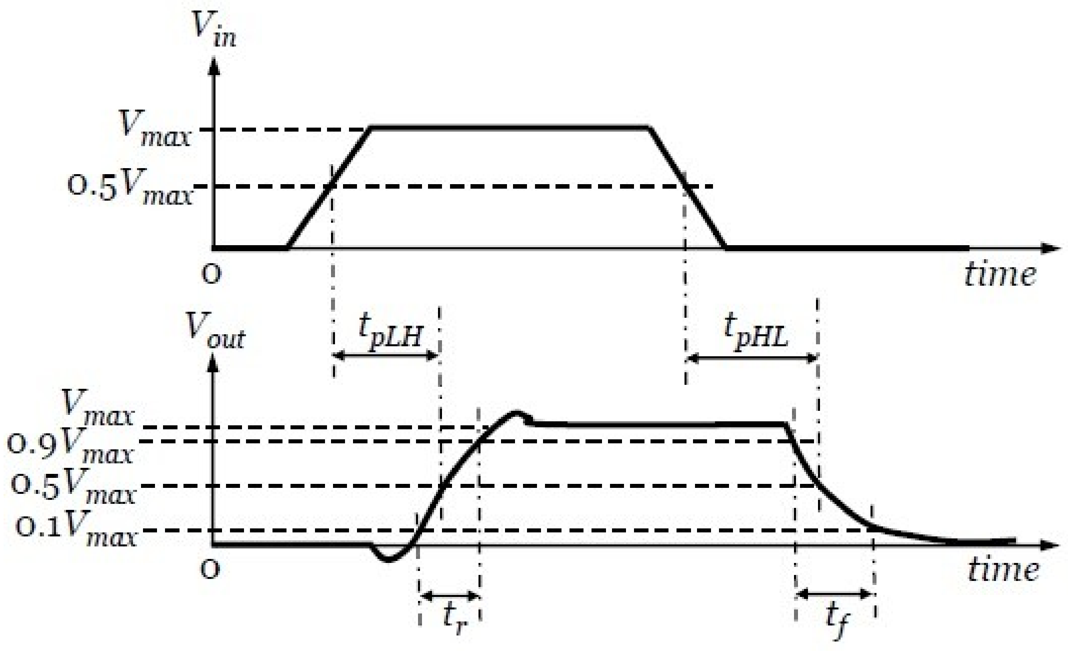
1. **CLOCK FREQUENCY** : The frequency at which the clock generator of a processor can generate pulses. Range of clock frequency over which a bistable device can be operated while maintaining stable transitions between logic levels at the outputs.
2. **PULSE DURATION** : Time interval between the specified reference points on the 2 transitions of the pulse waveforms.



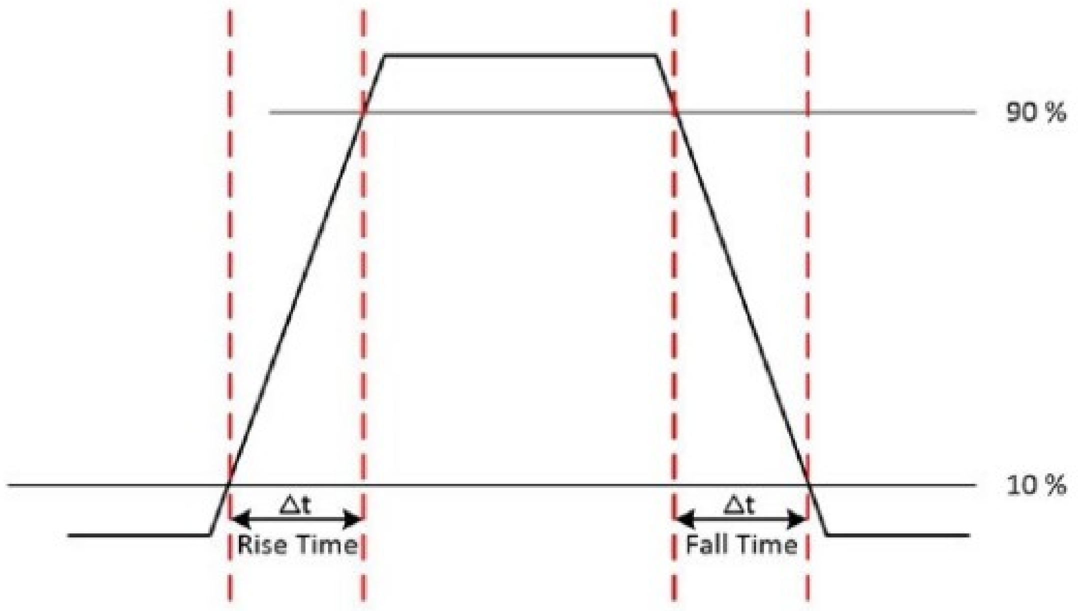
3. **SETUP TIME** : Minimum amount of time i.e., the shortest interval for which correct operation of the specified input is given to the digital circuit.
4. **HOLD TIME** : The time interval during which a signal is retained at a specified input terminal after an active transition occurs at another specified input terminal.



5. **PROPAGATION DELAY TIME** : The time interval between specified reference points on the input & output voltage waveforms with the output changing from one defined level (high or low) to the other defined level.



6. **ENABLE TIME** : Propagation time between specified reference points on the input and output voltage waveforms with the output changing from a high impedance(OFF) state to either of the defined active levels (high or low).
7. **DISABLE TIME** : The propagation time between the specified reference points on the input and output voltage waveforms with the output changing from either of the defined active levels (high or low) to the high impedance (off) state.
8. **RISE TIME** : Rise time is the time taken for a signal to transition from a specified low voltage level to a specified high voltage level. In digital circuits, this typically refers to the time required for the signal to change from 10% to 90% of its final value.
9. **FALL TIME** : Fall time is the time taken for a signal to transition from a specified high voltage level to a specified low voltage level. Typically, this refers to the time required for the signal to change from 90% to 10% of its initial value.
10. **SKEW RATE** : It is the time delta between the actual and expected arrival time of a clock signal.



11. **SLEW RATE** : The rate of change of an output voltage.
12. **PULSE SKEW** : It is the magnitude of the time difference between the high to low (t_{PHL}) & the low to high (t_{PLH}).
13. **JITTER** : It is the timing variations of a set of signals from their ideal values. These are typically caused by noise (or) other disturbances in the system.
14. **CLOCK LATENCY** : It is mainly classified into two types :

    i. ***CLOCK SOURCE LATENCY :*** Propagation delay from the origin of clock to the clock definition point.

    ii. ***CLOCK NETWORK LATENCY :*** Propagation delay from a clock definition point to a register clock pin.

    CLOCK LATENCY = CLOCK SOURCE LATENCY + CLOCK NETWORK LATENCY
15. **CLOCK UNCERTAINTY** : Clock Jitter + Clock Skew

Q. Why Setup Time and Hold Time are Needed ?

Ans. 1. **Ensuring Reliable Data Capture**:

- **Setup Time**: If data changes too close to the clock edge, it might not propagate through the flip-flop's internal circuitry in time, leading to incorrect data being captured.
- **Hold Time**: If data changes immediately after the clock edge, the flip-flop might not have sufficient time to capture the correct data, again leading to incorrect data storage.

2. **Avoiding Metastability**:
    - Metastability is a condition where a flip-flop fails to resolve its output to a stable '0' or '1' due to a violation of setup or hold times. This can cause indeterminate states and unpredictable behavior in digital circuits.
    - **Setup and hold times** ensure that the data is stable during critical periods, reducing the risk of metastability.
3. **Proper Timing Analysis**:

- Setup and hold times are essential parameters for timing analysis, which is crucial for ensuring that all timing constraints are met in a digital design.
- Timing analysis tools use setup and hold times to check if data paths meet the required timing constraints, ensuring the circuit operates correctly at the desired clock frequency.

4. **Clock Skew Management**:

- Clock skew, the difference in the arrival time of the clock signal at different flip-flops, can affect the setup and hold times.
- Properly designed setup and hold times help manage clock skew by providing margins within which data must be valid, ensuring reliable operation despite variations in clock arrival time.

5. **Interfacing with Different Clock Domains**:

- In systems with multiple clock domains, setup and hold times are crucial for designing safe data transfer mechanisms between domains.
- Ensuring proper setup and hold times helps avoid timing violations and data corruption when transferring data across different clock domains.

Practical Considerations

1. **Design Margins**:
    - Designers often add margins to setup and hold times to account for variations in manufacturing, voltage, and temperature (PVT variations).
    - These margins ensure that the circuit remains reliable under different operating conditions.

2. **Timing Closure**:
    - Achieving timing closure, where all timing constraints are met, involves ensuring that setup and hold times are satisfied for all data paths in the design.
    - Tools like static timing analysis (STA) are used to verify that setup and hold times are not violated across the entire design.
3. **Performance Optimization**:
    - Minimizing setup and hold times can lead to higher clock frequencies and better performance.
    - Careful design and optimization of the circuit can help reduce these times, allowing the system to operate faster.

3. XILINX vs ALTERA

DIFFERENCES	XILINX	ALTERA
1. LOGIC BLOCKS	Fundamental blocks in Xilinx is slice, which typically contains a pair of 4 input LUT's & a pair of flip flops. Multiple slices are grouped together into larger CLB's.	It uses Adaptive Logic Modules (ALM's) which is a granular approach. ALM contains several LUT's that share inputs & outputs to improve efficiency.
2. INTERCONNECTS	Arranges logic into columns & vertical & horizontal routing channels between each column 3D stacking technology is used for greater interconnect density.	Denser fractal based interconnect approach, faster cascade chains within logic blocks before entering interconnect.
3. TRANSCEIVERS	Implement DSP algorithms is soft logic that offers flexibility but consume more logic resources.	Builds full hard input transmission blocks with dedicated analog circuitry area & power efficient
4. DESIGN SOFTWARE	VIVADO Design Suite	Quartus Prime
5. PERFORMANCE & POWER	***Power*** : Uses techniques like adaptive voltage scaling and dynamic power management to reduce power consumption. Their 7nm Versal ACAP's (Adaptive Compute Acceleration Platform) are designed for low power and high efficiency. ***Process*** : Leverages advanced process technologies like 7nm for their Versal ACAP's and UltraScale+ series, contributing to better power efficiency and performance.	***Power*** : Also focuses on power efficiency with innovations like power-aware synthesis and low-power transceiver technology. The Agilex series, built on Intel's 10nm process, emphasizes power efficiency. ***Process*** : Uses Intel's process technology, with the Agilex series built on 10nm technology, offering a balance between power efficiency and performance.

MODULE 4 : FABRICATION

As we have seen seen the basic principles of the VLSI design clearly of the main requirements that need to be kept in mind while designing an IC. So let's see the fabrication steps as well.

1. IC Fabrication Steps :

The basic IC fabrication steps will be described in the following sections. Some of these steps may be carried out many times, in different combinations and/or processing conditions during a complete fabrication run :

1.1. Silicon Wafers :

Modern integrated circuits start with very-high-purity, single-crystal silicon, initially grown as a solid cylinder, typically 10 cm to 30 cm in diameter and up to two meters long. These cylinders are sawed into thin wafers, 400μm to 600μm thick, resembling circular discs. The wafer surfaces are polished to a mirror finish using chemical and mechanical techniques. Semiconductor manufacturers typically buy these ready-made wafers rather than starting from ingots.

The electrical and mechanical properties of the wafer depend on crystal orientation, impurity concentrations, and types of impurities, tightly controlled during growth. Doping adds specific impurities to alter silicon's electrical properties, affecting resistivity. Depending on the impurity type, either holes (in p-type silicon) or electrons (in n-type silicon) conduct electricity. Heavy doping (e.g., > ~10^18 atoms/cm^3) results in low-resistivity silicon, denoted as n+ or p+. Lightly doped silicon (e.g., < ~10^16 atoms/cm^3) is labeled n− or p−. This control over doping enables the creation of diodes, transistors, and resistors in integrated circuits.

1.2. Oxidation :

To enhance the oxidation process where silicon reacts with oxygen to form silicon dioxide (SiO2), high temperatures (around 1000–1200°C) and ultra-clean furnaces are essential. Operating in a clean room environment is crucial to prevent contamination that could alter silicon's electrical properties. Particle filters ensure dust-free airflow, while personnel wear lint-free clothing from head to toe.

Oxygen can be introduced as high-purity gas (dry oxidation) or steam (wet oxidation). Wet oxidation grows oxide layers faster, while dry oxidation offers superior electrical characteristics. Silicon dioxide, the resulting oxide, has excellent electrical insulation with a dielectric strength of about 10^7 V/cm and a dielectric constant of 3.9. It's ideal for forming MOS capacitors and acts as a barrier against impurities during doping.

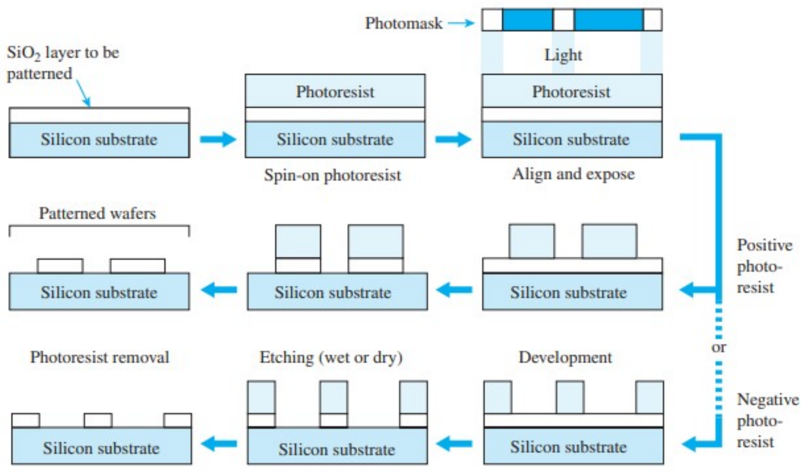
Silicon dioxide is transparent and creates reflective surfaces. When white light shines on an oxidized wafer, interference effects create colors that indicate the oxide layer's thickness. This principle is used in optical tools to measure film thickness. Finished wafers show vivid colors due to varying oxide thicknesses, visible to the naked eye.

1.3. Photolithography :

Mass production efficiency is the key driver behind the profound impact of Very Large Scale Integration (VLSI) technology on society. Photolithography enables precise definition of integrated circuit components through a series of steps. Initially, a photosensitive layer called photoresist is spun onto the wafer surface. A photographic plate with patterns is then used to selectively expose the photoresist using deep ultraviolet light. Exposed areas become softened (for positive photoresist) and are removed with a chemical developer, transferring the mask pattern onto the wafer.

This technique allows for accurate replication of intricate surface geometries. Patterns are either directly projected onto the wafer or via a photomask produced with a 10x reduction technique. The patterned photoresist serves as a mask for subsequent wet chemical etching or reactive ion etching processes, selectively removing materials like silicon dioxide, silicon nitride, polysilicon, and metal layers. After etching, the photoresist is stripped away, leaving behind the desired pattern on the wafer.

In advanced VLSI fabrication, more than 20 masking layers may be used, each requiring precise alignment with previous layers. This demands extremely tight mechanical and optical tolerances from the photolithography equipment.



1.4. Etching :

To permanently imprint the photographic patterns onto the wafer, chemical (wet) etching or RIE dry etching procedures can be used. Chemical etching is usually referred to as wet etching. Different chemical solutions can be used to remove different layers. For example, hydrofluoric (HF) acid can be used to etch SiO2, potassium hydroxide (KOH) for silicon, phosphoric acid for aluminum, and so on. In wet etching, the chemical usually attacks the exposed regions that are not protected by the photoresist layer in all directions (isotropic etching). Depending on the thickness of the layer to be etched, a certain amount of undercut will occur. Therefore, the dimension of the actual pattern will differ slightly from the original pattern. If exact dimension is critical, RIE dry etching can be used. This method is essentially a directional bombardment of the exposed surface using a corrosive gas (or ions). The cross section of the etched layer is usually highly directional (anisotropic etching) and has the same dimension as the photoresist pattern. A comparison between isotropic and anisotropic etching is given in Fig.

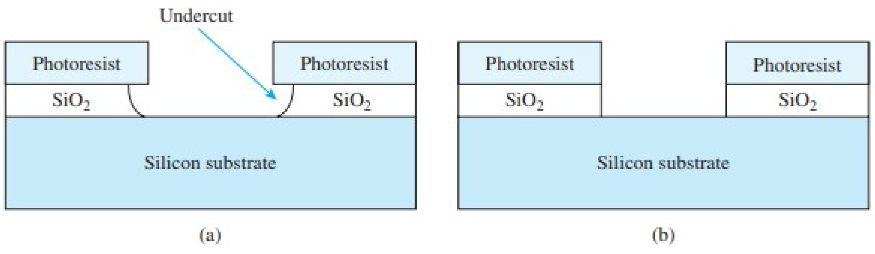


Figure (a) Cross-sectional view of an isotropic oxide etch with severe undercut beneath the photoresist layer. (b) Anisotropic etching, which usually produces a cross section with no undercut.

1.5. Diffusion :

Diffusion in VLSI involves atoms moving from high to low concentration, akin to ink spreading in water but slower in solids. It's used to introduce dopant atoms (impurities) into silicon to alter resistivity. Temperature strongly affects dopant diffusion rate, typically done at 1000–1200°C. Cooling "freezes" dopants in place. Furnaces used for oxidation are also used for diffusion. Dopant depth depends on temperature and time. Common dopants include boron (p-type), phosphorus, and arsenic (n-type). Oxide layers effectively mask these dopants. Boron in an n-type substrate forms a pn junction (diode). Heavy doping creates a low-resistivity conducting layer.

1.6. Ion Implantation :

Ion implantation introduces dopants into semiconductors by accelerating dopant ions with an electric field to embed them in the crystal lattice. The depth of penetration depends on ion beam energy, controlled by accelerating-field voltage. Ion quantity is adjusted by beam current. This method offers precise and reproducible dopant profiles compared to diffusion, as voltage and current are precisely regulated. Ion implantation is favored for applications needing exact doping control and can be conducted at room temperature.

1.7. Chemical Vapor Deposition :

Chemical vapor deposition (CVD) deposits solid materials on a substrate by reacting gases or vapors. It can deposit SiO2, Si3N4, polysilicon, etc., on silicon. Silane gas and oxygen produce silicon dioxide on silicon substrates. CVD oxide is a sufficient electrical insulator but inferior to thermally grown oxide. CVD deposits oxide faster and at lower temperatures (<500°C).

Using silane alone deposits silicon. Above 1000°C, crystalline silicon forms epitaxial layers on crystalline silicon substrates. At lower temperatures or on non-crystalline surfaces, polycrystalline silicon (poly Si) forms with randomly aligned small silicon crystals. Heavily doped polysilicon layers are highly conductive for electrical interconnections.

1.8. Metallization :

Metallization connects components in integrated circuits by depositing metal over silicon surfaces. Metal is sputtered onto wafers in a vacuum chamber using an Ar ion gun, which physically dislodges metal atoms from a pure metal disk (e.g., 99.99% aluminum). Ar ions, being inert, do not react with metal but help coat surfaces. Sputtering time (1 to 2 minutes) controls metal film thickness. Photolithography and etching define metal interconnect patterns.

1.9. Packaging :

A finished silicon wafer may contain several hundreds of finished circuits or chips. A chip may contain from 10 to more than 108 transistors; each chip is rectangular and can be up to tens of millimeters on a side. The circuits are first tested electrically (while still in wafer form) using an automatic probing station. Bad circuits are marked for later identification. The circuits are then separated from each other (by dicing), and the good circuits (dies) are mounted in packages (headers). Fine gold wires are normally used to interconnect the pins of the package to the metallization pattern on the die. Finally, the package is sealed using plastic or epoxy under vacuum or in an inert atmosphere.

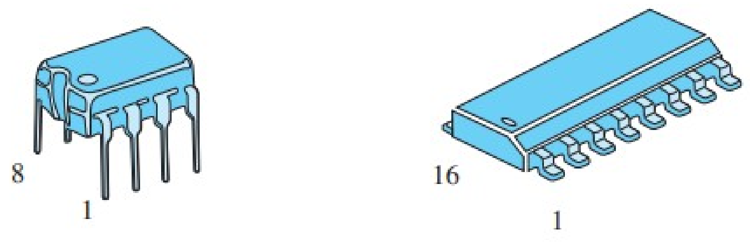


Figure Examples of an 8-pin plastic dual-in-line IC package and a 16-pin surface-mount package.

2. BEYOND 20nm TECHNOLOGY – FINFET TECHNOLOGY

Moore's Law, coined by Intel co-founder Gordon Moore in 1965, predicted that the number of transistors on VLSI chips would double approximately every two years, necessitating consistent transistor size reduction to prevent chip size increase, which would lower yield and increase cost. Scaling involves shrinking both horizontal (surface layout) and vertical dimensions (eg., gate oxide thickness, junction depths). This process enhances integration density and speed of MOSFET’s, improving overall performance. Typically, dimensions are reduced by about 50% every two generations, denoted by technology nodes like 1 µm (1990), 0.7 µm, 0.5 µm, and so forth.

Approaching the 20 nm node, challenges arise from ultra thin gate dielectrics and short channel lengths causing undesirable leakage currents (gate and drain-source). Solutions include using high-k dielectrics (eg., HfO2) for MOS gates. To further integration, new device structures like ultra-thin-body (UTB) devices are explored, with promising options such as FINFET’s, where a thin silicon fin stands vertically from the wafer surface, fully depleting during off-states to minimize leakage. As of 2014, 16 nm FINFET technology is in use for high-performance VLSI chips.

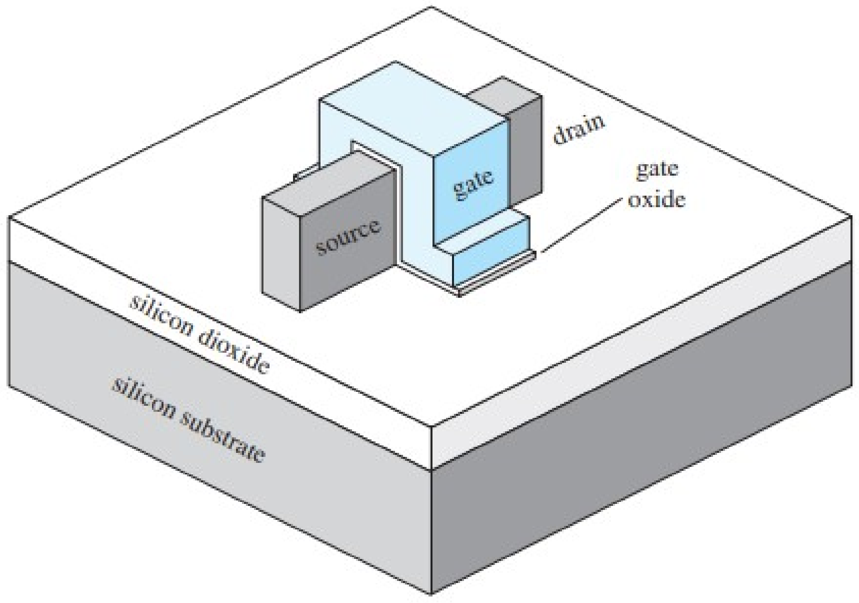


Figure A perspective view of the FINFET showing a 3D gate warped around a very thin slab of silicon fin. The source and drain contact areas are actually larger than the intrinsic device.

3. PACKAGING

Packaging plays a crucial role in the final stages of semiconductor fabrication, ensuring that integrated circuits (ICs) are protected, interconnected, and ready for use in electronic devices. It involves enclosing the delicate silicon chips in protective casings that shield them from environmental factors such as moisture, dust, and mechanical stress. Additionally, packaging facilitates the electrical connections between the IC and the external circuitry, enabling the functionality of the device.

In this report, we explore the various materials essential to the packaging process of integrated circuits. These materials are carefully selected and engineered to meet stringent requirements for reliability, thermal management, electrical performance, and miniaturization in modern electronics.

TAPE : The IC's are placed in cavities on a longitudinal strip of plastic tape. Usually made of a durable material like polystyrene (or) polycarbonate.

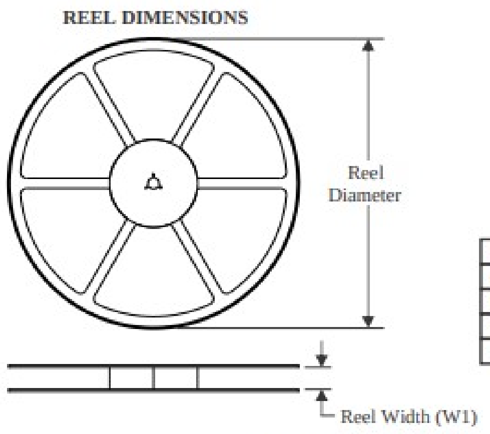
REEL : Tape is wound onto a reel, which is cylindrical spindle like structure, This allows in easy handling & transport.

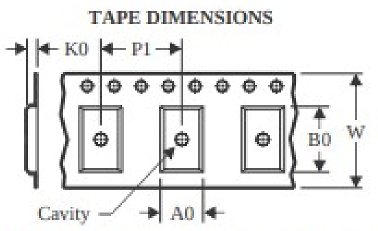
POCKET QUADRANTS : Sections (or) components on the tape where individual components are housed. Typically arranged in a grid pattern along the length of the tape, each quadrant contains 1 IC.

P1 : Pitch is the distance between the center-lines of Adjacent pocket quadrants.

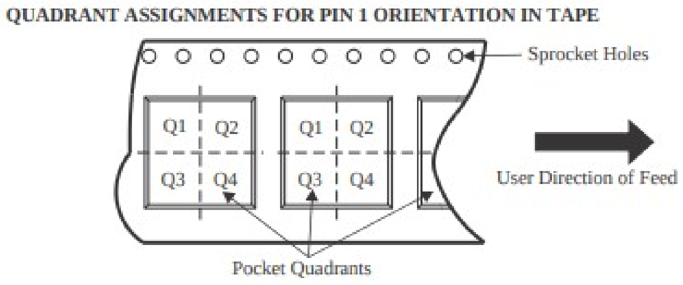
SPROCKET HOLES : These help in ensuring accurate alignment & registration of different layers of the chip during the lithography process.

TAPE AND REEL INFORMATION





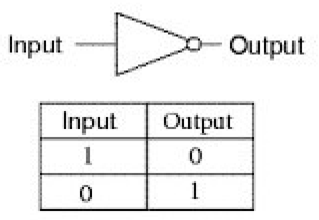
A0	Dimension designed to accommodate the component width
B0	Dimension designed to accommodate the component length
K0	Dimension designed to accommodate the component thickness
W	Overall width of the carrier tape
P1	Pitch between successive cavity centers



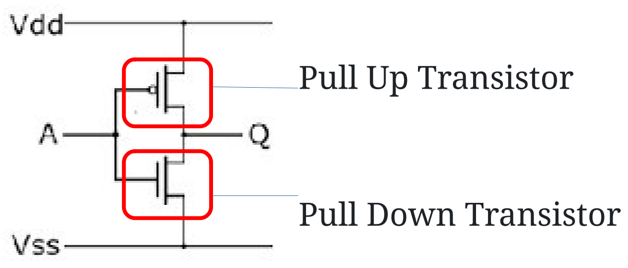
MODULE 5 : C-MOS INVERTER DESIGN & ANALYSIS USING SKY130 PDK

We know that neither N-MOS nor P-MOS can be used for design that can produce either values, HIGH and LOW. But another thing that is worth notice is how they complement each other. This is what gave rise to an idea of attaching them together. Since, P-MOS is a Strong 1, we put it between V_{DD} and V_{out} and N-MOS being a STRONG 0, it is placed between Vout and GND. This way, either can act as a load to the other transistor, since both are never ON together. The configuration looks like what we have below. This is referred to as Complimentary Metal Oxide Semiconductor(C-MOS) Configuration and it also represents the simplest circuit known as the C-MOS Inverter.

C-MOS Circuits generally consists of a network split into two parts, Upper one referred to as a pull up network and the lower half as a pull down network. The former consists of P-channel MOSFET's and later N-Channel MOSFET's. Reason is simple. As one transistor is one, another is off. This eliminates the issue of an resistive path to the ground and hence, no voltage division occurs(At least not a significant one). This way, one can easily achieve a Strong High and a Strong LOW from the same network. PULL UP is what offers a low resistance path to the VDD and PULL DOWN is what offers a low resistance path the GND.



Input	Output
1	0
0	1



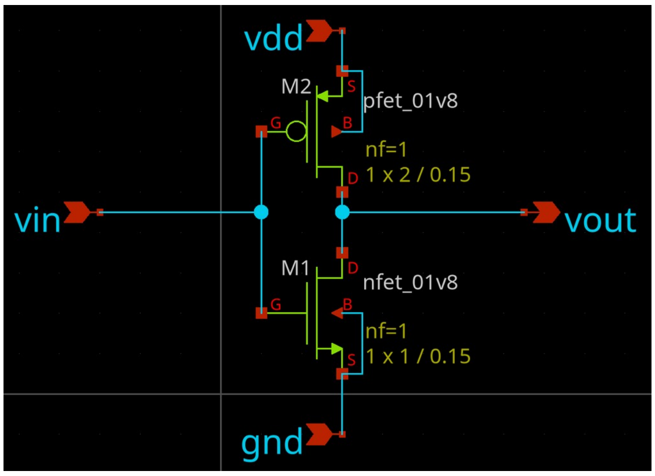
PULL – UP TRANSISTORS : This transistor is responsible for pulling the output node (usually a signal line) to a high voltage level (logic level 1). It is typically a P-Channel MOSFET connected between the output node & the positive power supply voltage (V_{CC}). When a pull-up transistor is turned on (conducting), It allows current flow from the positive supply voltage to the output node, pulling the output voltage up to a high level.

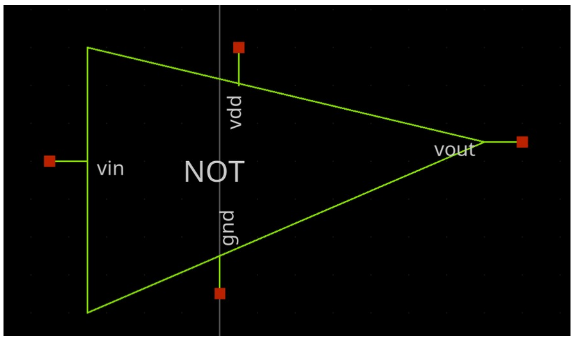
PULL-DOWN TRANSISTORS : This type of transistor is responsible for pulling the output node to a low logic level (0). Typically a N-Channel MOSFET connected between the output node & GND (Or) negative power supply voltage. When PDT is turned ON, it provides low resistive path for current to flow from the output node to ground, pulling the output voltage down to a low level.

1.1. C-MOS Inverter Analysis Pre-Layout :

In electronics it is very popularly explained as something that performs the NOT logic, that is complements the input. So, HIGH(1.8V) becomes LOW(0V) and vice versa. Ideally, the output follows the input and there is no delay or propagation issues of the circuit. But in reality, an inverter can be a real piece of work. It can have several issues like how fast can it react to the changes in the input, how much load can it tolerate before it's output breaks and so many more including noise, bandwidth, etc.

So here I have designed an Inverter where, I have chosen (W/L) of P-MOS = 2 & (W/L) of N-MOS = 1.

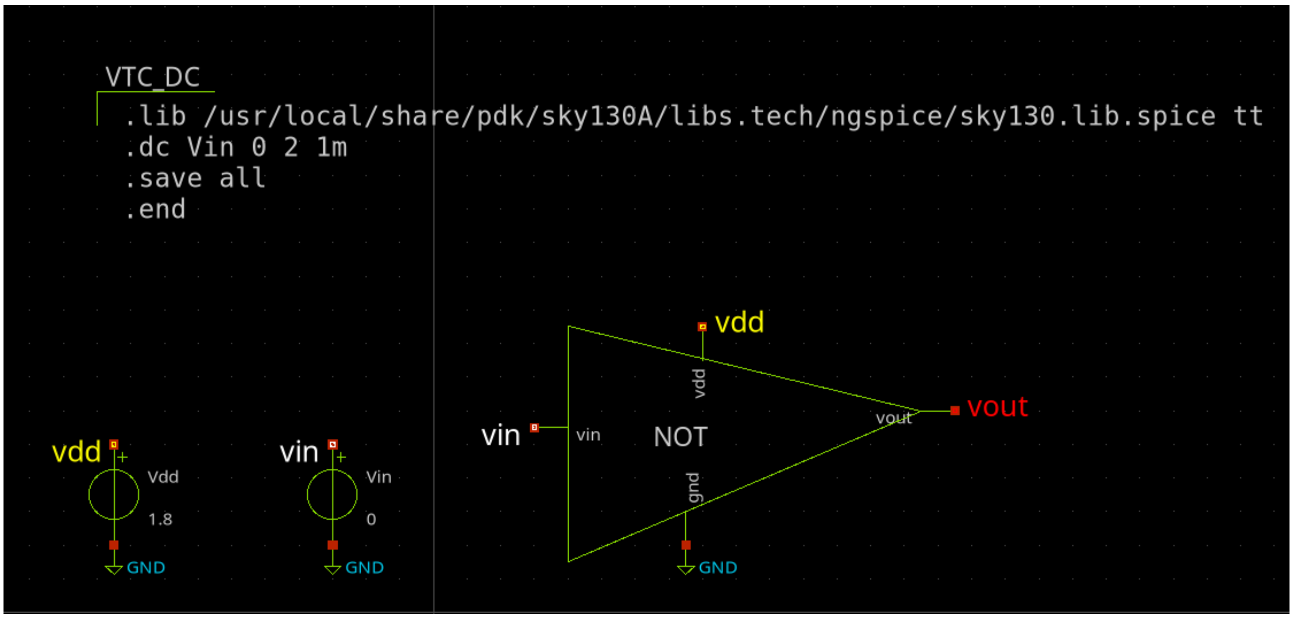




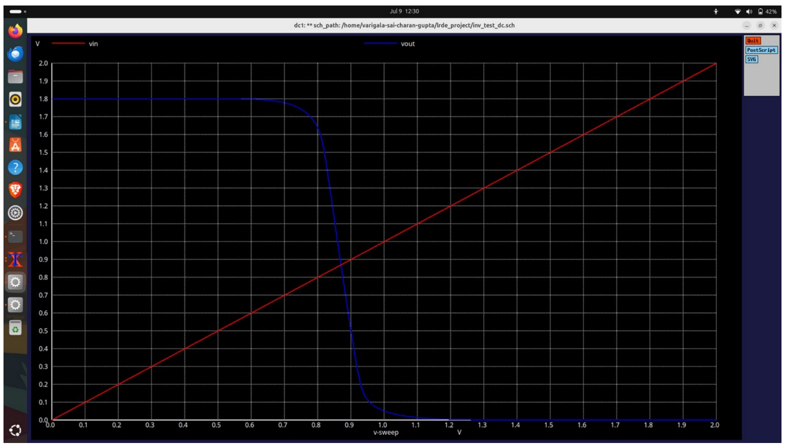
1.2. DC ANALYSIS & Important Design Parameters :

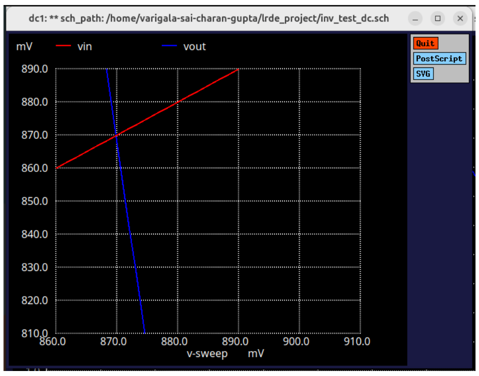
DC analysis would be used to plot a Voltage Transfer Characteristics (VTC) curve for the circuit. It will sweep the value of Vin from high to low to determine the working of circuit with respect to different voltage levels in the input.

A voltage transfer characteristics paints a plot that shows the behavior of a device when it's input is changed(full swing). It shows what happens to the output as input changes. In our case, for an inverter we can see a plot that is like a square wave(non ideal), that changes it's nature around 0.869 volts of input. So one can say that there are like 3 regions in the VTC curve, the portion where output is high, the place of transition and the one where the output goes low. But actually there are five regions of operation and they are based on the working of inverter constituents, that is the N-MOS and the P-MOS transistors with respect to the change in the input potential.



```
VTC_DC
.lib /usr/local/share/pdk/sky130A/libs.tech/ngspice/sky130.lib.spice tt
.dc Vin 0 2 1m
.save all
.end
```



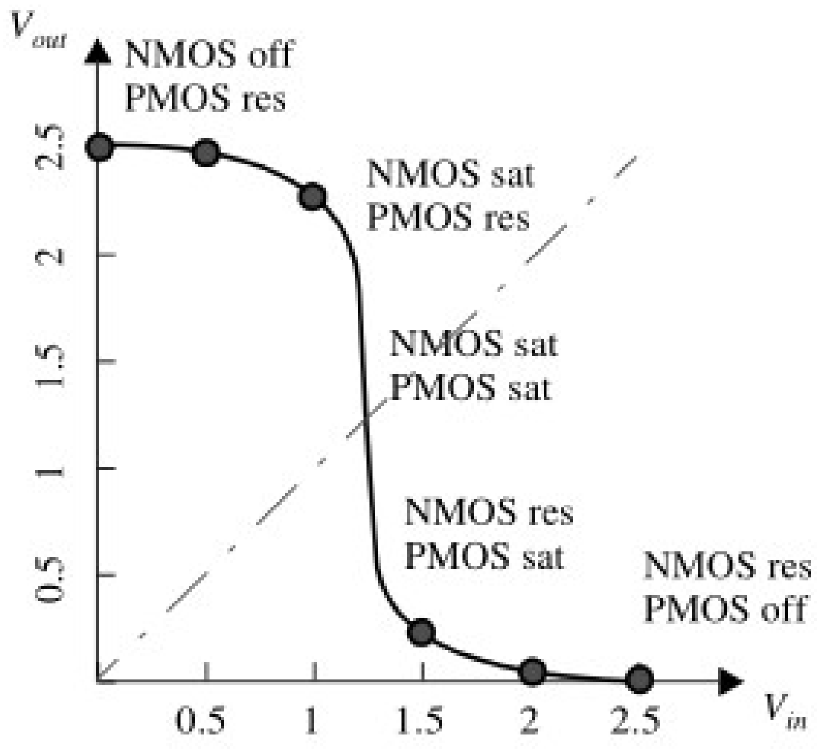


V_m (when $V_{in} = V_{out}$) = 8.698293e-01 (as per simulations done)
One can solve for them using the equations for individual transistors. Now it is time to talk about the important parameters of this device that are based off it's VTC curve.

- •V_{OH} - Maximum output voltage when it is logic '1'.
- •V_{OL} - Minimum output voltage when it is logic '0'.
- •V_{IH} - Maximum input voltage that can be interpreted as logic '0'.
- •V_{IL} - Minimum input voltage that can be interpreted as logic '1'.

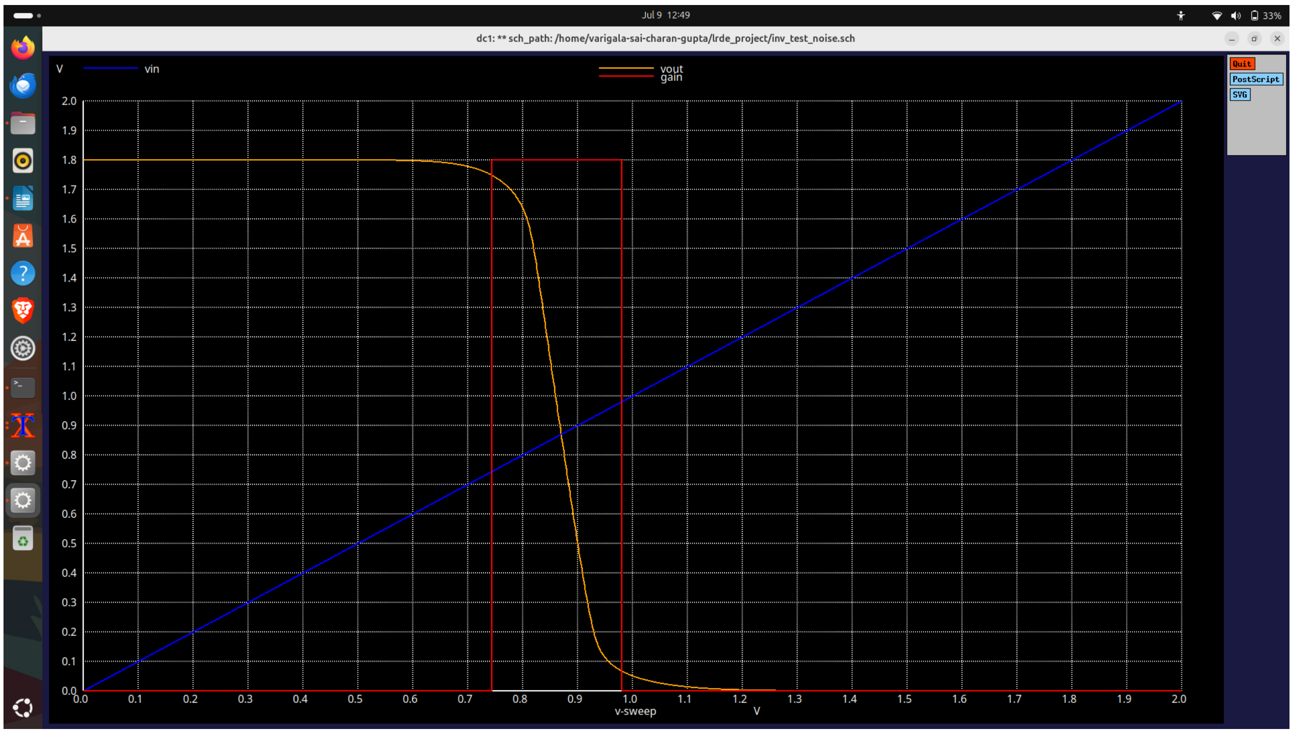
•V_{th} - Inverter Threshold voltage

Above five are critical for an Inverter and can be seen on the VTC curve of an inverter.



1.3. NOISE ANALYSIS :

Vth should be at a value of VDD/2 for maximum noise margins



to plot the above graph we need to first find the gain curve which is in red, to get this we write

```
let gain=(abs(deriv(vout)) >= 1)*1.8
plot gain vin vout
```

the value of the noise margin that is to get the values of Vil and Vih :
we use Ng-Spice commands for that as follows :

for Vil : `meas vil dc find vin when gain=1 cross=1`
we get Vil = 7.435556e-01 = 0.74 = noise margin for the lower side

for Vih : `meas vih dc find vin when gain=1 cross=last`
we get Vih = 9.804444e-01
MAX RANGE = 1.8
Noise margin for higher side = 1.8 – 0.9804444
= 0.82 (roughly)

**** as we are not using the ideal case of W/L ration of P-MOS/N-MOS = 3.5 we don't get the lower and higher noise margin symmetrical.*

Noise margins are defined as the range of values for which the device can work noise free or with high resistance to noise. This is an important parameter for digital circuits, since they work with a set of specific values(2 for binary systems), so

it becomes crucial to know what values of the voltages can it sustain for each value. This range is also referred to as Noise Immunity. There are two such values of Noise margins for a binary system:
NML(Noise Margin for Low) = V_{IL} - V_{OL}
NMH(Noise Margin for HIGH) = V_{OH} – V_{IH}

1.3.1 Maximizing the Noise Margins :

When designing static C-MOS circuits, it is advisable to balance the driving strengths of the transistors by making the P-MOS section wider than the N-MOS section, if one wants to maximize the noise margins and obtain symmetrical characteristics.
We derive the sizes of PMOS and NMOS transistors such that the switching threshold of a CMOS inverter, implemented in our generic 0.25 μm CMOS process, is located in the middle between the supply rails. We use the process parameters presented in Example 3.7, and assume a supply voltage of 2.5 V. The minimum size device has a width/length ratio of 1.5.

$$\frac{(W/L)_p}{(W/L)_n} = \frac{115 \times 10^{-6}}{30 \times 10^{-6}} \times \frac{0.63}{1.0} \times \frac{(1.25 - 0.43 - 0.63/2)}{(1.25 - 0.4 - 1.0/2)} = 3.5$$

1. V_M is relatively insensitive to variations in the device ratio. This means that small variations of the ratio (eg., making it 3 or 2.5) do not disturb the transfer characteristic that much. It is therefore an accepted practice in industrial designs to set the width of the P-MOS transistor to values smaller than those required for exact symmetry.
2. The effect of changing the W_p/W_n ratio is to shift the transient region of the VTC. Increasing the width of the P-MOS or the N-MOS moves V_M towards V_{DD} or GND respectively. This property can be very useful, as asymmetrical transfer characteristics are actually desirable in some designs. This is demonstrated by the example of Figure. The incoming signal Vin has a very noisy zero value. Passing this signal through a symmetrical inverter would lead to erroneous values (Figure a). This can be addressed by raising the threshold of the inverter, which results in a correct response (Figure b). Further in the text, we will see other circuit instances where inverters with asymmetrical switching thresholds are desirable. Changing the switching threshold by a considerable amount is however not easy, especially when the ratio of supply voltage to transistor threshold is relatively small (2.5/0.4 = 6 for our particular example). To move the threshold to 1.5 V requires a transistor ratio of 11, and further increases are prohibitively expensive.

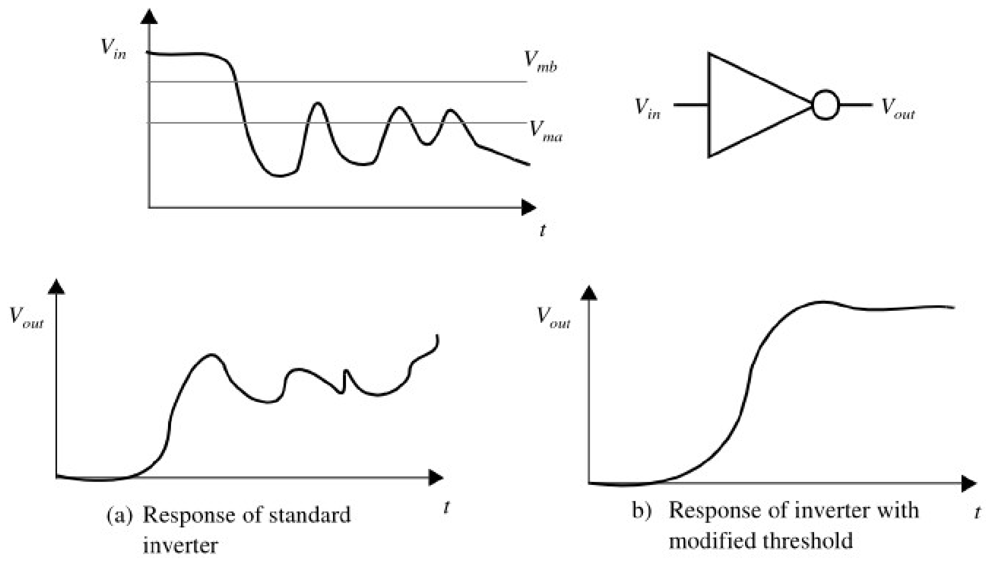
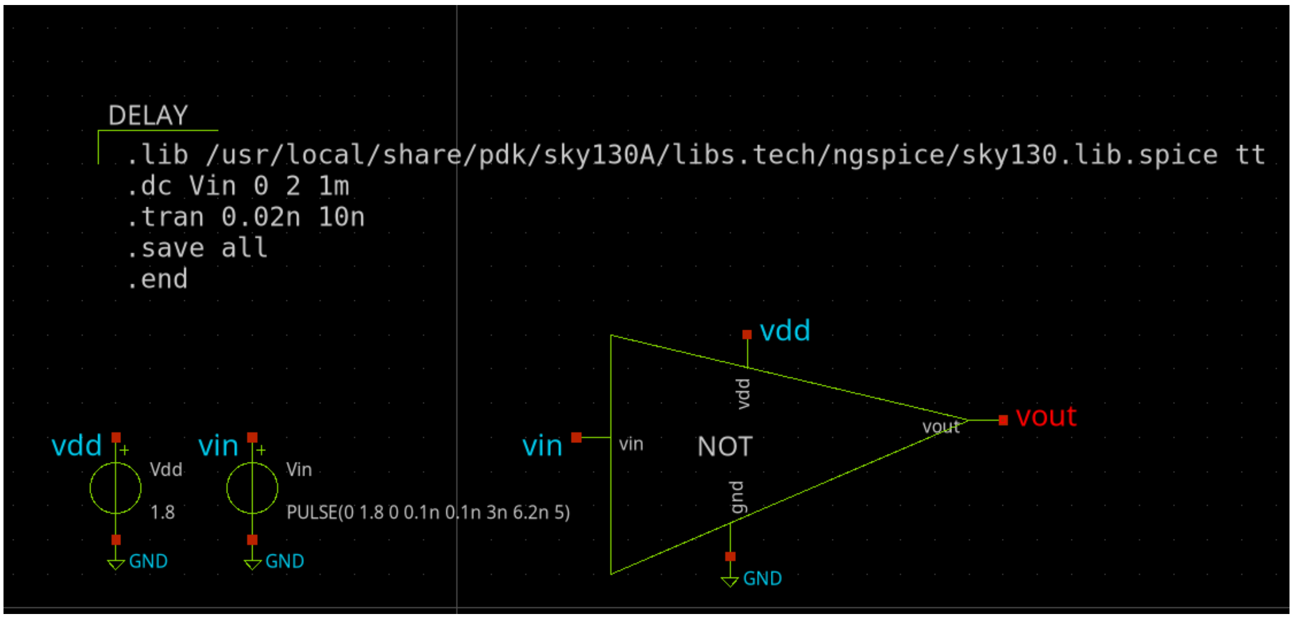

V_{in}
V_{mb}
V_{ma}
t
V_{in}
V_{out}
V_{out}
t
(a) Response of standard inverter
V_{out}
t
b) Response of inverter with modified threshold

Figure Changing the inverter threshold can improve the circuit reliability.

1.4. DELAY ANALYSIS OF C-MOS INVERTER :

First define the input as PULSE and let's set the spice command tell that take the input as the pulse given so that is transient one so we write :



```
DELAY
.lib /usr/local/share/pdk/sky130A/libs.tech/ngspice/sky130.lib.spice tt
.dc Vin 0 2 1m
.tran 0.02n 10n
.save all
.end
```

setplot tran1

1.4.1. UNLOADED CAPACITANCE (UNLOADED DELAY) :

now to get the T_{pHL}

1. We need the time taken for the Vin to reach 90% of RISE. (vin50)
2. We need the time taken for the Vout to reach 90% of FALL.(vout50)
3. Then T_{pHL} = vout50 – vin50.

meas tran vin50 when vin=0.9 RISE=2 (we get vin50)
meas tran vout50 when vout=0.9 FALL=2 (we get vout50)
let tpHL=vout50-vin50
print tpHL (we get T_{pHL})

CASE 1 :
* when we have W=2(pfet) & W=1(nfet)
* ".lib /usr/local/share/pdk/sky130A/libs.tech/ngspice/sky130.lib.spice tt
  .dc Vin 0 2 1m
  .tran 0.02n 10n
  .save all
  .end"
* Vin = "PULSE(0 1.8 0 0.3n 0.3n 3n 6.6n 5)"

we get :
vin50 = 6.750000e-09
vout50 = 6.774882e-09
tpHL = 2.488200e-11 = 24 ps

CASE 2 :

```
* when we have W=2(pfet) & W=1(nfet)
* ".lib /usr/local/share/pdk/sky130A/libs.tech/ngspice/sky130.lib.spice tt
    .dc Vin 0 2 1m
    .tran 0.02n 10n
    .save all
    .end"
* Vin = "PULSE(0 1.8 0 0.1n 0.1n 3n 6.2n 5)"
```

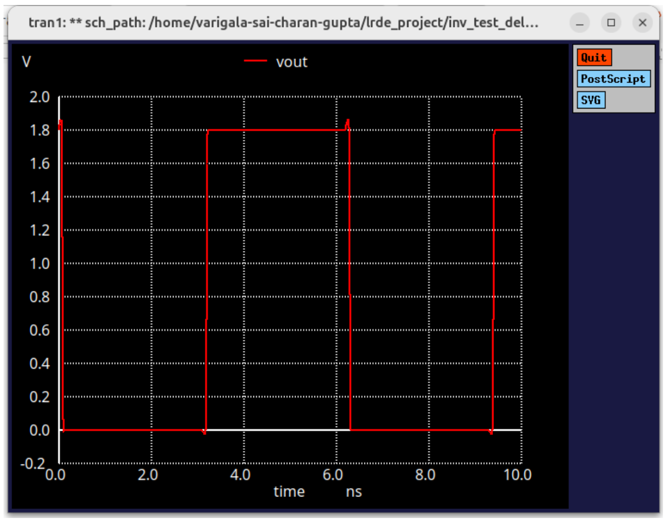
we get :
vin50 = 6.250000e-09
vout50 = 6.268350e-09
tpHL = 1.835000e-11 = 18 ps

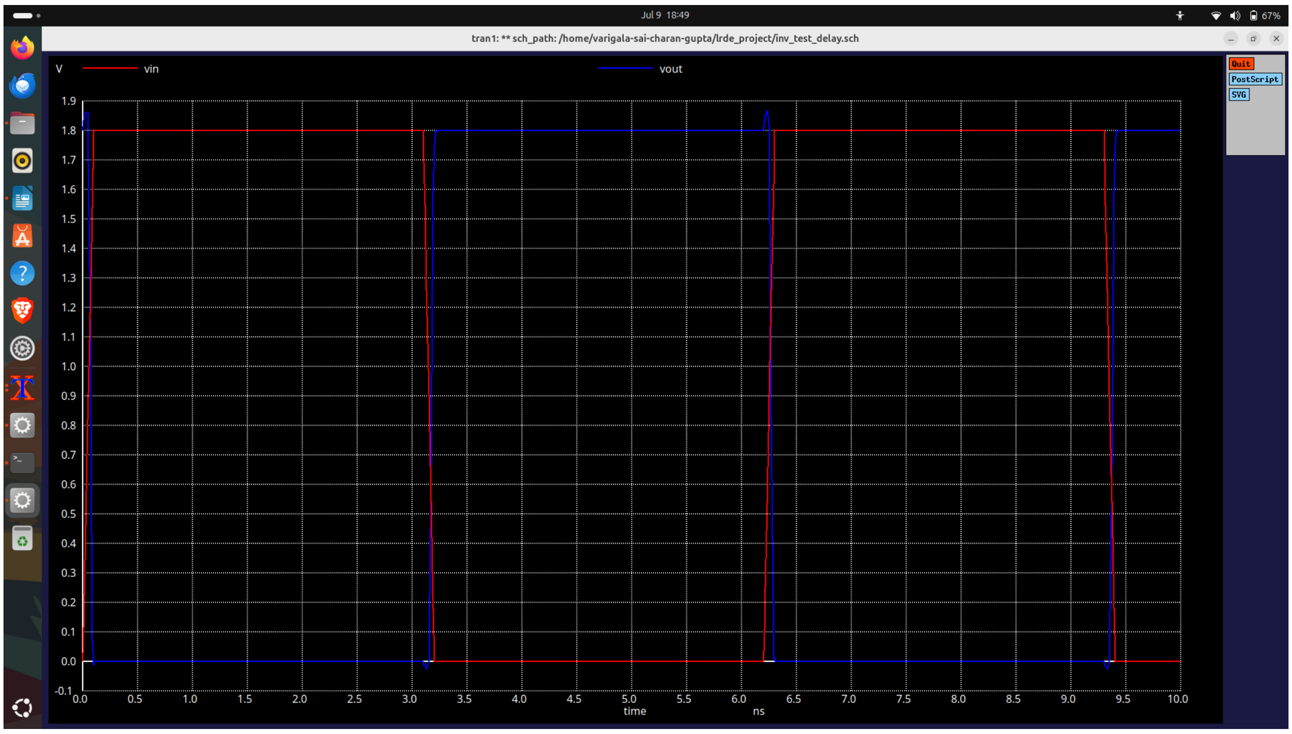
Now to calculate RISE & FALL times for the CASE 2 we write the code as follows :

```
meas tran t10 when vout=0.18 RISE=1   (we get t10)
meas tran t90 when vout=1.6 RISE=1    (we get t90)
let tr=t90-t10
print tr   (we get tr)
```

now according to the input given in CASE 2 we get the values as follows :
t10 = 3.155216e-09
t90 = 3.189992e-09
tr = 3.477600e-11

***as it is a isolated inverter it doesn’t have any load capacitance but due to internal capacitance we are getting this delay and this delay is known as **unloaded delay**.




tran1: ** sch_path: /home/varigala-sai-charan-gupta/lrde_project/inv_test_delay.sch
V
vin
vout
Quit
PostScript
SVG
time
ns

Now to get the FALL TIME we write :

meas tran t10 when vout=1.6 FALL=2 (we get t10)
meas tran t90 when vout=0.18 FALL=2 (we get t90)
let tf=t10-t90
print tf (we get t_f)

t10 = 6.285755e-09
t90 = 6.254331e-09
t_f = 3.142400e-11

1.4.2. LOADED CAPACITANCE (LOADED DELAY) :

CASE 1 :
* when we have W=4(pfet) & W=2(nfet)
* ".lib /usr/local/share/pdk/sky130A/libs.tech/ngspice/sky130.lib.spice tt
    .dc Vin 0 2 1m
    .tran 0.02n 40n
    .save all
    .end"
* Vin = "PULSE(0 1.8 0 0.1n 0.1n 10n 20n 10)"
* Capacitance = 0.125pf

now according to the input given in CASE 1 we get the values as follows :

For RISE TIME :

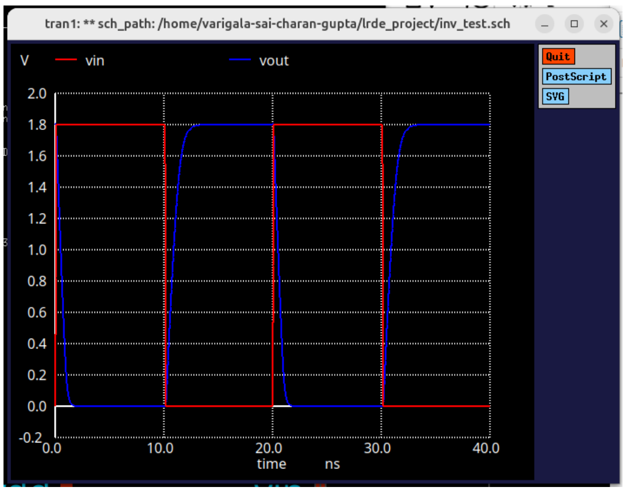
meas tran t10 when vout=0.18 RISE=1   (we get t10)
meas tran t90 when vout=1.6 RISE=1   (we get t90)
let tr=t90-t10
print tr   (we get tr)

t10 = 1.029360e-08
t90 = 1.159990e-08
tr = 1.306300e-09 = 1.3ns

For FALL TIME :

meas tran t10 when vout=0.18 FALL=2
(we get t10)
meas tran t90 when vout=1.6 FALL=2
(we get t90)
let tf=t10-t90
print tf   (we get tf)

t10 = 2.103020e-08
t90 = 2.017867e-08
tf = 8.515300e-10



CASE 2 :
* when we have W=2(pfet) & W=1(nfet)
* ".lib /usr/local/share/pdk/sky130A/libs.tech/ngspice/sky130.lib.spice tt
    .dc Vin 0 2 1m
    .tran 0.02n 40n
    .save all
    .end"
* Vin = "PULSE(0 1.8 0 0.1n 0.1n 10n 20n 10)"
* Capacitance = 0.125pf

now according to the input given in CASE 2 we get the values as follows :

For RISE TIME :

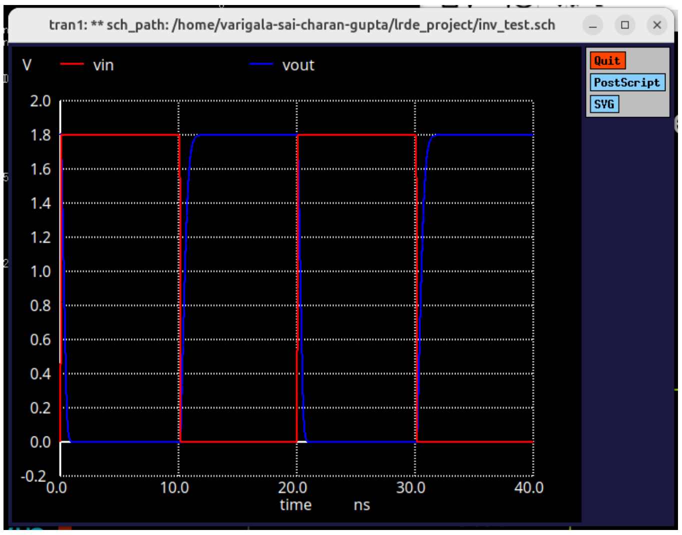
meas tran t10 when vout=0.18 RISE=1 (we get t10)
meas tran t90 when vout=1.6 RISE=1 (we get t90)
let tr=t90-t10
print tr (we get tr)

t10 = 1.023664e-08
t90 = 1.089515e-08
tr = 6.585100e-10

For FALL TIME :

meas tran t10 when vout=0.18 FALL=2 (we get t10)
meas tran t90 when vout=1.6 FALL=2 (we get t90)
let tf=t10-t90
print tf (we get tf)

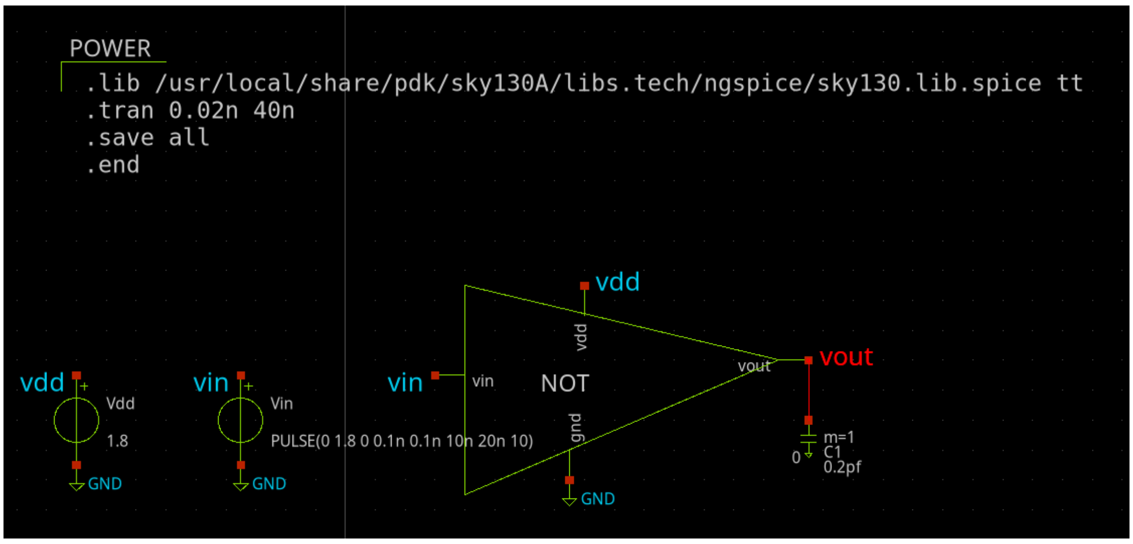
t10 = 2.055750e-08
t90 = 2.012817e-08
tf = 4.293300e-10



The propagation delay of a gate can be minimized in the following ways:

1. Reduce C_L. Remember that three major factors contribute to the load capacitance: the internal diffusion capacitance of the gate itself, the interconnect capacitance, and the fan-out. Careful layout helps to reduce the diffusion and interconnect capacitances. Good design practice requires keeping the drain diffusion areas as small as possible.
2. Increase the W/L ratio of the transistors. This is the most powerful and effective performance optimization tool in the hands of the designer. Proceed however with caution when applying this approach. Increasing the transistor size also raises the diffusion capacitance and hence C_L. In fact, once the intrinsic capacitance (i.e. the diffusion capacitance) starts to dominate the extrinsic load formed by wiring and fan-out, increasing the gate size does not longer help in reducing the delay, and only makes the gate larger in area. This effect is called “self-loading”. In addition, wide transistors have a larger gate capacitance, which increases the fan-out factor of the driving gate and adversely affects its speed.
3. Increase V_{DD}. The delay of a gate can be modulated by modifying the supply voltage. This flexibility allows the designer to trade-off energy dissipation for performance, as we will see in a later section. However, increasing the supply voltage above a certain level yields only very minimal improvement and hence should be avoided. Also, reliability concerns (oxide breakdown, hot-electron effects) enforce firm upper-bounds on the supply voltage in deep sub-micron processes.

1.5. POWER ANALYSIS :


POWER
```
.lib /usr/local/share/pdk/sky130A/libs.tech/ngspice/sky130.lib.spice tt
.tran 0.02n 40n
.save all
.end
```

Each time the capacitor C_L gets charged through the P-MOS transistor, its voltage rises from 0 to V_{DD}, and a certain amount of energy is drawn from the power supply. Part of this energy is dissipated in the P-MOS device, while the remainder is stored on the load capacitor. During the high-to-low transition, this capacitor is discharged, and the stored energy is dissipated in the N-MOS transistor.

A precise measure for this energy consumption can be derived. Let us first consider the low-to-high transition. We assume, initially, that the input waveform has zero rise and fall times, or, in other words, that the N-MOS and P-MOS devices are never on simultaneously. Therefore, the equivalent circuit of Figure a is valid. The values of the energy E_{VDD}, taken from the supply during the transition, as well as the energy EC, stored on the capacitor at the end of the transition, can be derived by integrating the instantaneous power over the period of interest. The corresponding waveforms of $v_{out}(t)$ and $i_{VDD}(t)$ are pictured in Figure b.

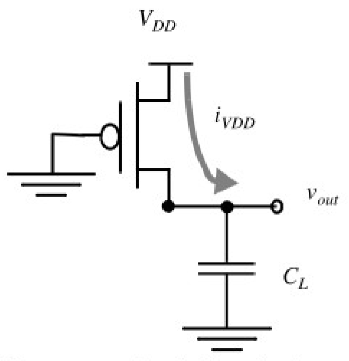


Figure a Equivalent circuit during the low-to-high transition.

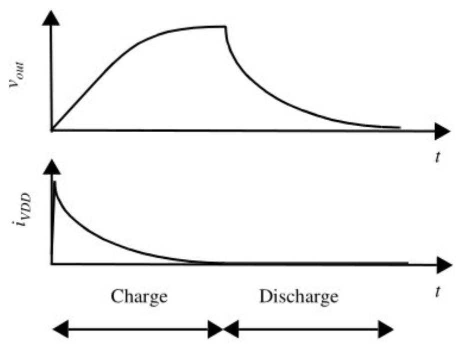


Figure b Output voltages and supply current during (dis)charge of C_L.

These results can also be derived by observing that during the low-to-high transition, C_L is loaded with a charge C_LV_{DD}. Providing this charge requires an energy from the supply equal to $C_LV_{DD}{}^2$ (= Q × V_{DD}). The energy stored on the capacitor equals $C_LV_{DD}{}^2/2$. This means that only half of the energy supplied by the power source is stored on C_L. The other half has been dissipated by the P-MOS transistor. Notice that this energy dissipation is independent of the size (and hence the resistance) of the P-MOS device! During the discharge phase, the charge is removed from the capacitor, and its energy is dissipated in the N-MOS device. Once again, there is no dependence on the size of the device. In summary,each switching cycle (consisting of an L→H and an H→L transition) takes a fixed amount of energy, equal to $C_LV_{DD}{}^2$. In order to compute the power consumption, we have to take into account how often the device is switched. If the gate is switched on and off $f_{0\to1}$ times per second, the power consumption equals

$$P_{dyn} = C_L V_{DD}{}^2 f_{0\to1}$$

$f_{0\to1}$ represents the frequency of energy-consuming transitions, this is 0 → 1 transitions for static C-MOS.

1.5.1. LOW ENERGY/POWER DESIGN TECHNIQUES :

With the increasing complexity of the digital integrated circuits, it is anticipated that the power problem will only worsen in future technologies. This is one of the reasons that lower supply voltages are becoming more and more attractive. Reducing V_{DD} has a quadratic effect on P_{dyn} .

When a lower bound on the supply voltage is set by external constraints (as often happens in real-world designs), or when the performance degradation due to lowering the supply voltage is intolerable, the only means of reducing the dissipation is by lowering the effective capacitance. This can be achieved by addressing both of its components: the physical capacitance and the switching activity.

A *reduction in the switching activity* can only be accomplished at the logic and architectural abstraction levels.

Lowering the physical capacitance is an overall worthwhile goal, which also helps to improve the performance of the circuit. As most of the capacitance in a combinational logic circuit is due to transistor capacitances (gate and diffusion), it makes sense to keep those contributions to a minimum when designing for low power.

This means that transistors should be kept to *minimal size* whenever possible or reasonable. This definitely affects the performance of the circuit, but the effect can be offset by using logic or architectural speed-up techniques. The only instances here transistors should be sized up is when the load capacitance is dominated by extrinsic capacitances (such as fan-out or wiring capacitance).

The concussions that can be drawn are :

- Device sizing, combined with supply voltage reduction, is a very effective approach in reducing the energy consumption of a logic network. This is especially true for networks with large effective fan-outs, where energy reductions with almost a factor of 10 can be observed. But the gain is also sizable for smaller values of F. The only exception is the F=1 case, where the minimum size device is also the most effective one.
- Oversizing the transistors beyond the optimal value comes at a hefty price in energy. This is unfortunately a common approach in many of today's designs.
- The optimal sizing factor for energy is smaller than the one for performance, especially for large values of F. For example, for a fan-out of 20, f_{opt}(energy) = 3.53, while f_{opt}(performance) = 4.47. Increasing the device sizes only leads to a minimal supply reduction once V_{DD} starts approaching V_{TE}, hence leading to very minimal energy gains.

MODULE 6 : C-MOS INVERTER LAYOUT DESIGN USING MAGIC VLSI TOOL

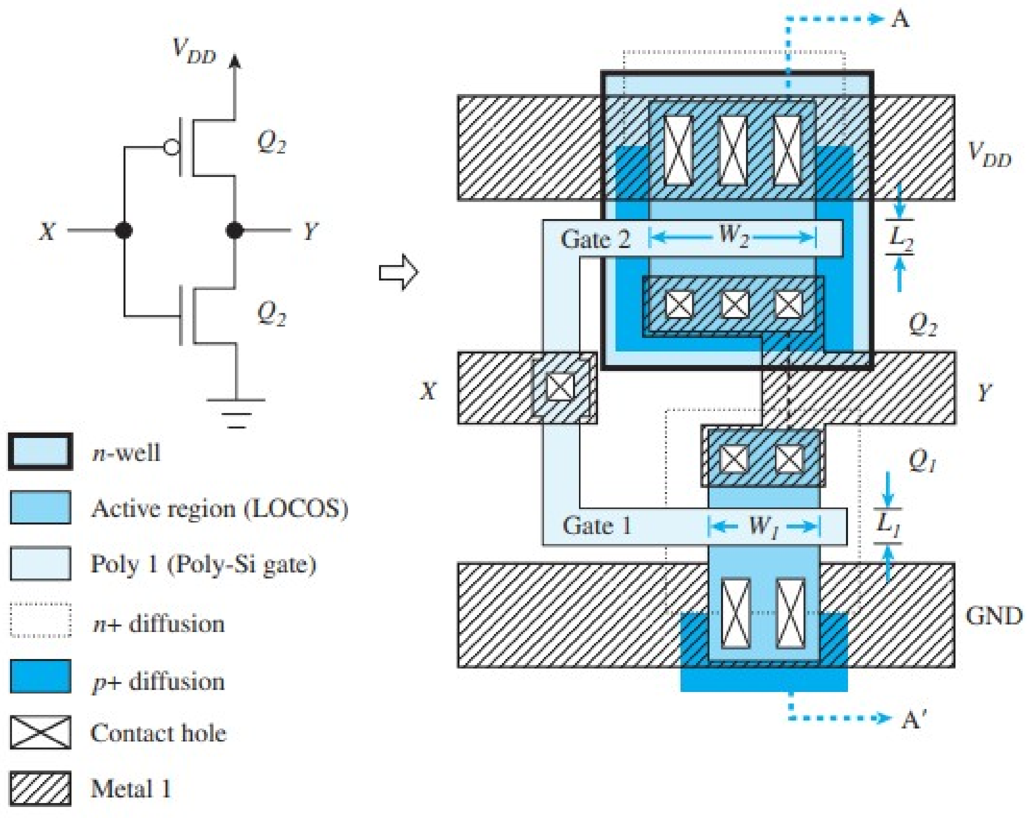


Figure A.1 A CMOS inverter schematic and its layout.

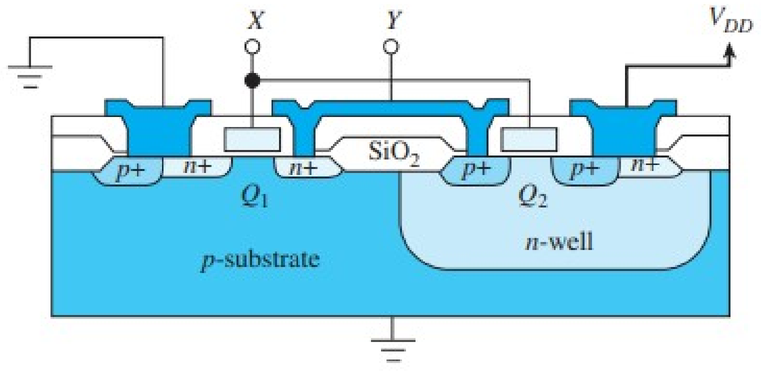
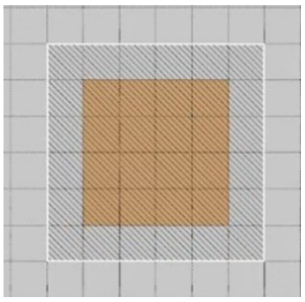


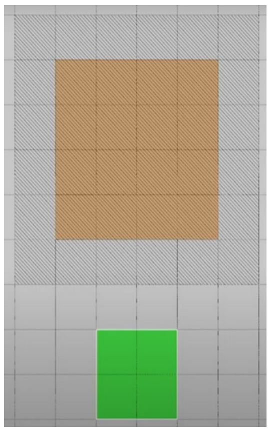
Figure A.2 Cross section along the plane AA′ of a CMOS inverter. Note that this particular layout is good for illustration purposes, but is not necessarily appropriate for latchup prevention.

As we want to design the layout of the C-MOS inverter as mentioned in the above figure, so we need to follow certain steps.

Steps to design the layout of the C-MOS inverter :

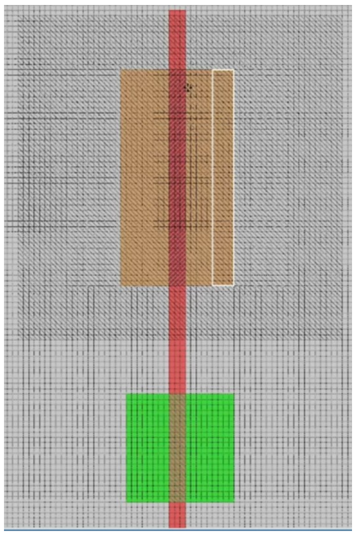


1. Set grid 0.5um
2. Snap to grid on
3. paint nwell (*by selecting the area on the grid area*)
4. paint pdiff (*paints the PMOS active area (p-diffusion) in this area.*)
5. paint ndiff (paints the NMOS active area (n-diffusion). A design rule is that n-diffusion must be at least 10 lambda away from p-diffusion. If you placed the n-diffusion area closer to the p-diffusion area, white dots will appear indicating design rule violation.)



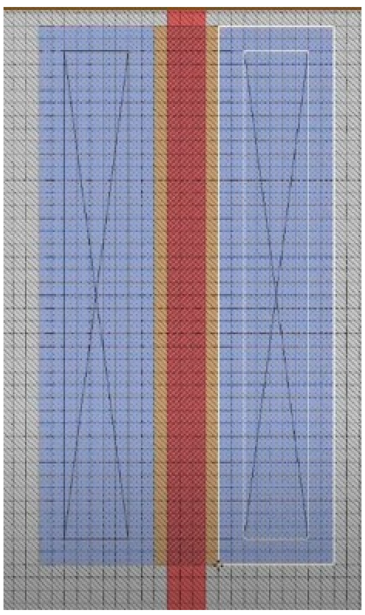
6. let's change the grid to 50nm*50nm

   grid 50nm 50nm
7. paint poly (now we add poly layer i.e. It is commonly used to form the gate electrodes of MOSFETs)
8. paint li ('li' stands for "local interconnect." The local interconnect layer is used to make short connections between devices within the same cell or small area on the integrated circuit. It is typically used for routing signals locally within a small region of the chip without the need to go up to the higher metal layers, which are used for longer distance routing.)

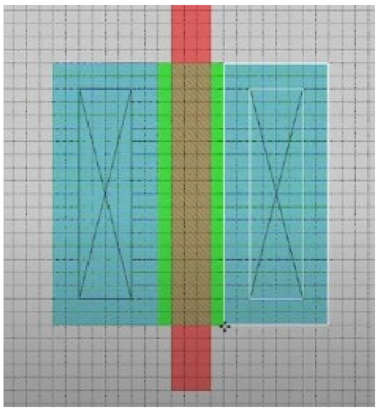


9. paint pdc (To make a contact between the Vdd! wire and pdiffusion area, form a box over at the overlap between the metal1 and pdiffusion areas and type the command:)
10. paint li
11. paint ndc (to make a contact between the GND wire and ndiffusion (source of the NMOS), make a box over the overlap between metal1 and n-diffusion layers.)
12. paint li

13. paint ntapc (It usually stands for an n-type tap contact, which is a contact to the n-well or n-type region. In CMOS technology, n-tap contacts are used to connect the n-well to a voltage reference, typically Vdd (positive power supply).)



14. paint ptap (The p-tap contact is used to connect the p-well or p-type substrate to a reference voltage, typically the ground (GND). This is essential for ensuring proper biasing and functioning of the p-well or p-type regions in CMOS technology.)
15. paint li
16. paint ptapc
17. paint m1 (‘m1’ stands for the first layer of metal, commonly referred to as Metal 1. Metal 1 is the lowest layer of the metal interconnect system used to create the wiring in an integrated circuit. It is typically used for local routing and connections between different components within the same cell or small region of the chip.)
18. label vdd



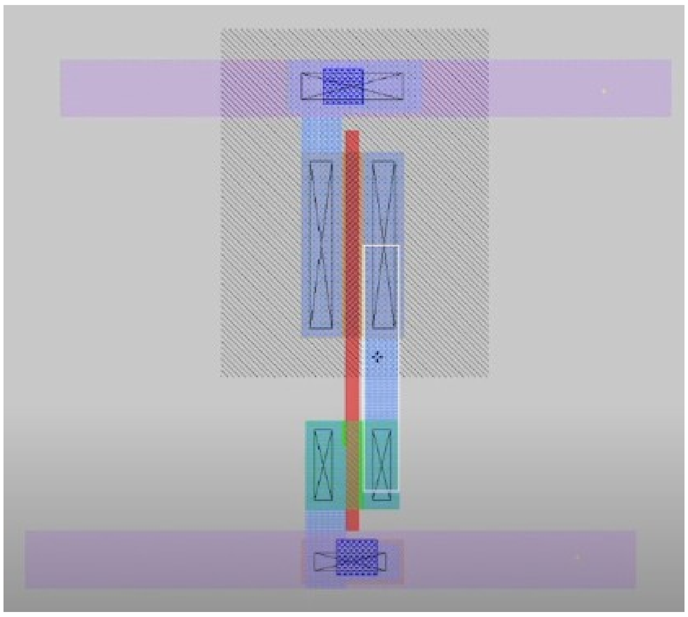
19. label vss
20. paint mcon (‘mcon’ stands for "metal contact." The metal contact layer is used to create connections between the lower layers, such as diffusion (active regions), polysilicon, and the first metal layer (Metal 1).

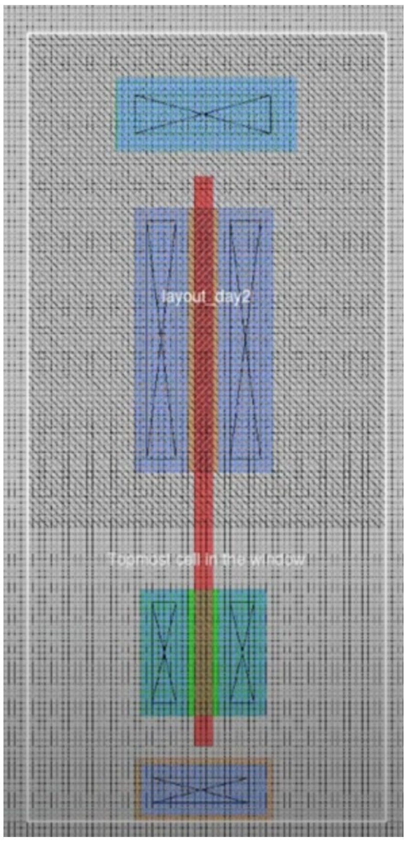
    Key roles of mcon in VLSI design include:

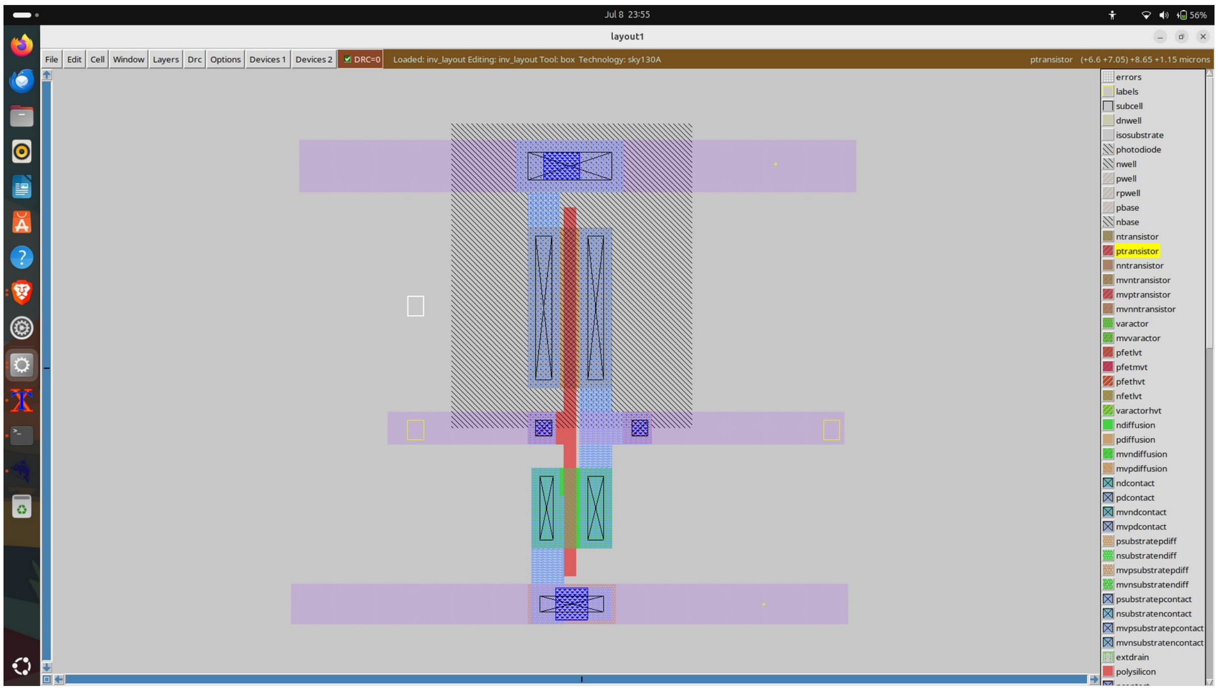
1. **Creating Contacts**: mcon is used to form the physical contact points between different layers, ensuring electrical connectivity. For example, it connects the Metal 1 layer to the underlying source/drain diffusion regions or to polysilicon gates.
2. **Reducing Resistance**: By providing a direct path for current to flow between layers, mcon helps reduce the contact resistance, improving the overall performance of the circuit.

3. **Enabling Interconnections**: It facilitates the creation of interconnections between transistors and other components on the chip.)

21. paint li
22. paint m1
23. paint li
24. paint polyc (i.e. it is polyconnect)





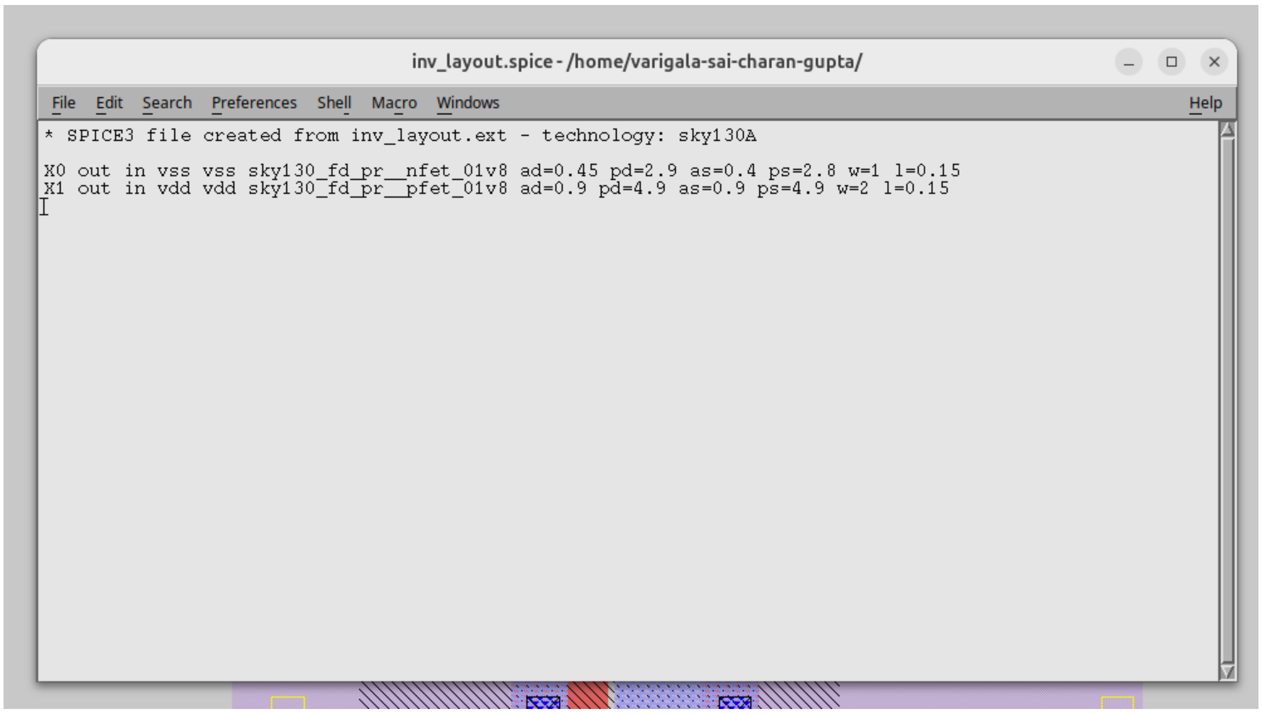


Loaded: inv_layout Editing: inv_layout Tool: box Technology: sky130A

ptransistor (+6.6 +7.05) +8.65 +1.15 microns

The above screenshot is the final layout of the C-MOS inverter circuit.

Now to get the netlist of the spice model to know what are the designed parameters we will first extract the spice models and then we get it as :


inv_layout.spice - /home/varigala-sai-charan-gupta/
File Edit Search Preferences Shell Macro Windows Help

```
* SPICE3 file created from inv_layout.ext - technology: sky130A

X0 out in vss vss sky130_fd_pr__nfet_01v8 ad=0.45 pd=2.9 as=0.4 ps=2.8 w=1 l=0.15
X1 out in vdd vdd sky130_fd_pr__pfet_01v8 ad=0.9 pd=4.9 as=0.9 ps=4.9 w=2 l=0.15
```

From this above screenshot we get an idea of what we have developed in MAGIC VLSI tool.

Now in general after this we need to perform the layout vs schematic and then we need to do the routing and we need to prepare the information for the packaging material, after this we will achieve the final product that is our desired NOT GATE IC.

CONCLUSION

This report has comprehensively covered key aspects of VLSI (Very Large Scale Integration) design, beginning with an introduction to VLSI and the various stages involved in the VLSI design process. By examining different types of VLSI designs, the distinctions between ASIC (Application-Specific Integrated Circuit) and FPGA (Field-Programmable Gate Array) were highlighted, along with an exploration of Configuration Logic Blocks (CLB's).

Further sections delved into the intricacies of simulation, synthesis, and implementation, with particular emphasis on clock and timing generation techniques, clock parameters, and a comparative study of Xilinx and Altera tools. The fabrication process was detailed, including IC fabrication steps, advancements in FINFET technology for beyond 20nm scaling, and the significance of packaging.

A specific focus was given to the design and analysis of a C-MOS inverter using the SKY130 PDK, encompassing pre-layout analysis, DC and noise analysis, delay and power analysis, and finally the layout design using the MAGIC VLSI tool.

Through these modules, the report has not only provided a theoretical understanding but also practical insights into VLSI design and implementation, equipping readers with a solid foundation and practical knowledge essential for advancing in the field of VLSI technology.

REFERENCES

1. CMOS DIGITAL INTEGRATED CIRCUITS “Analysis and Design” 2ND EDITION by SUNG-MO (STEVE) KANG [University of Illinois at Urbana- Champaign] & YUSUF LEBLEBIGI [Worcester Polytechnic Institute] Swiss Federal Institute of Technology-Lausanne.
2. R. S. Muller, T.I. Kamins, and M. Chan, Device Electronics for Integrated Circuits, 3rd ed., Hoboken, NJ, John Wiley & Sons, 2003.
3. J. D. Plummer, M.D. Deal, and P.B. Griffin, Silicon VLSI Technology, Upper Saddle River, NJ, Prentice Hall, 2000.
4. S. Wolf, Microchip Manufacturing, Lattice Press (www.latticepress.com), 2004.
5. MAGIC Tutorials 1 through 11, included with the MAGIC release 6.5 software.
6. WhyRD “VLSI PROJECT TUTORIALS”
7. Git hub Repository by Subham Rath under the guidance of Prof. Santunu Sarangi and Prof. Saroj Rout in collaboration with VSD
8. Digital integrated circuits “a design perspective” by Jan M. Rabaey prentice hall electronics and VLSI series charles G. Sooni, Series editor.