VLSI System (DI05000021)

Milav Dabgar

Lecturer, GP Palanpur

Table of Contents I

Table of Contents II

Table of Contents III

Table of Contents IV

Lecture 1.1: Energy band diagram and MOS system under external bias

External Bias: Modes of Operation

- Applying a gate voltage $V_G \neq 0$ forces the Fermi levels to shift: $E_{FM} - E_{FS} = -qV_G$.
- For a p-type substrate, three distinct electrostatic states emerge based on V_G:
- 1. Accumulation ($V_G < 0$)
- 2. Depletion ($V_G > 0$, but small)
- 3. Inversion ($V_G > V_T$, large positive bias)

Accumulation Mode ($V_G < 0$)

Diagram Placeholder

Energy band diagram showing upward band bending near the oxide interface. Display positive charges (holes) accumulating exactly at the oxide-semiconductor boundary.

- Applied V_G is negative. E_{FM} is raised relative to E_{FS}.
- Electric field points from the semiconductor toward the gate.
- Majority carriers (holes) are attracted to the Si-SiO2 interface.
- Energy bands bend upward at the surface ($\phi_s < 0$). E_V moves closer to E_{FS}.

Depletion Mode ($V_G > 0$, V_G ¡ V_T)

Diagram Placeholder

Energy band diagram showing downward band bending. Include a physical cross section below it showing the depletion region devoid of mobile carriers, populated only by negative fixed acceptor ions.

- Applied V_G is positive, but small. E_{FM} is lowered.
- Holes are repelled from the surface, leaving behind fixed, uncompensated, negatively charged acceptor ions (N_A^-).
- A depletion region of width W_d forms.
- Energy bands bend downward ($\phi_s > 0$). E_i moves closer to E_{FS}.

Depletion Region Width

- The depletion region width W_d increases with increasing surface potential ϕ_s.
- Using Poisson's equation, W_d can be derived as:
- $W_d = \sqrt{(2 \cdot \epsilon_s \cdot \phi_s)/(q \cdot N_A)}$
- The total depletion charge per unit area is $Q_d = -q \cdot N_A \cdot W_d$.

Onset of Inversion

- As V_G increases further, bands bend downward so much that the intrinsic level E_i crosses the Fermi level E_{FS} at the surface.
- When E_i falls below E_{FS}, the surface electron concentration exceeds the hole concentration ($n_s > p_s$).
- The p-type silicon surface has electrically 'inverted' to become n-type.
- Weak inversion occurs when $0 < \phi_s < 2\phi_F$.

Strong Inversion ($V_G \geq V_T$)

Diagram Placeholder

Energy band diagram showing severe downward bending, with E_C drawn very close to E_{FS} at the interface. Clearly label $\phi_s = 2\phi_F$.

- Strong inversion is formally defined when the surface electron concentration n_s equals the bulk hole concentration N_A.
- This occurs when the surface potential $\phi_s = 2\phi_F$, where ϕ_F is the bulk Fermi potential.
- A highly conductive n-type inversion layer (channel) is fully formed at the interface.
- Any further increase in V_G results primarily in an increase in inversion charge (Q_{inv}), while W_d maxes out at $W_{d(max)}$.

Threshold Voltage Definition

- The Threshold Voltage (V_T) is the specific gate voltage V_G required to achieve strong inversion ($\phi_s = 2\phi_F$).
- $V_T = V_{FB} + 2\phi_F - (Q_{d,max}/C_{ox})$
- V_{FB} accounts for non-ideal flat-band voltage (work function differences and oxide charges).
- $Q_{d,max}$ is the depletion charge at maximum width: $-\sqrt{2q\epsilon_s N_A(2\phi_F)}$.

Summary of Charge Distribution

- $Q_G + Q_s = 0$ (Charge neutrality of the MOS system)
- Accumulation: Q_s is positive (holes at surface).
- Depletion: Q_s is negative (fixed acceptor ions, Q_d).
- Inversion: $Q_s = Q_d + Q_{inv}$. The total negative charge is shared between the fixed depletion layer and the mobile inversion layer.

Key Takeaways

- The ideal MOS capacitor requires flat energy bands at zero bias.
- Gate voltage modulates surface potential, forcing the semiconductor into accumulation, depletion, or inversion.
- Strong inversion requires $\phi_s = 2\phi_F$, forming a mobile carrier channel essential for transistor operation.
- Threshold voltage V_T is heavily dependent on oxide capacitance, substrate doping, and material work functions.

Next Lecture Preview

- Next time, we transition from the 2-terminal MOS capacitor to the 4-terminal MOSFET.
- We will introduce the Source and Drain terminals.
- We will examine circuit symbols and the fundamental physical structure of NMOS and PMOS devices.
- We will define basic operational regions (Cutoff, Linear, Saturation).

Lecture 1.2: Symbols, Structure and operation of MOSFET

Symbols, Structure and Operation of MOSFET

- Unit 1: MOS TRANSISTOR
- Lecture 2: The Four-Terminal Device
- Focus: Device cross-sections, standard schematic symbols, and basic current conduction regimes.

Agenda

- The Four Terminals: Gate, Source, Drain, Body
- Enhancement vs. Depletion Mode Devices
- NMOS and PMOS Transistor Symbols
- Physical Structure of the NMOS Transistor
- The Concept of the Channel (L and W)
- Basic Operation: Cutoff Region
- Basic Operation: Linear / Triode Region
- Basic Operation: Saturation Region
- P-Channel MOSFET (PMOS) Specifics

The Four Terminals of a MOSFET

- Gate (G): The control terminal. A conductive electrode separated from the channel by the thin oxide layer.
- Source (S): Heavily doped region that acts as the source of charge carriers (electrons for NMOS, holes for PMOS).
- Drain (D): Heavily doped region that collects charge carriers from the channel.
- Body / Substrate / Bulk (B): The lightly doped semiconductor foundation. Often tied to the Source or the lowest/highest potential in the circuit.

Enhancement vs. Depletion Mode

- Enhancement-Mode (Normally OFF):
- - No conducting channel exists at $V_{GS} = 0$.
- - Requires an applied gate voltage ($V_{GS} > V_T$) to 'enhance' the channel and permit current flow.
- - Standard device in digital CMOS logic.
- Depletion-Mode (Normally ON):
- - A physical channel is implanted during fabrication.
- - Conducts current at $V_{GS} = 0$.
- - Requires an applied gate voltage to 'deplete' the channel and turn the device OFF.

NMOS Schematic Symbols

- The standard 4-terminal symbol shows an arrow on the Body terminal pointing INWARD (p-type body to n-type channel).
- Enhancement mode is represented by a broken/dashed line for the channel (indicating normally OFF).
- Depletion mode uses a solid continuous line for the channel.
- In digital design, a simplified 3-terminal symbol is often used, omitting the body terminal (assuming Source=Body connection).

Diagram: NMOS Schematic Symbols

PMOS Schematic Symbols

- The 4-terminal PMOS symbol shows an arrow on the Body terminal pointing OUTWARD (n-type body to p-type channel).
- Similar dashed (enhancement) vs. solid (depletion) channel conventions apply.
- In digital schematics, PMOS devices are frequently denoted by a 'bubble' (inversion circle) on the gate terminal.
- The bubble signifies active-low behavior (turns ON when Gate is Low/GND).

Diagram: PMOS Schematic Symbols

Physical Structure Geometry (L and W)

- Channel Length (L): The distance between the Source and Drain junctions. Defines the technology node (e.g., 180nm, 28nm).
- Channel Width (W): The transverse dimension of the gate. Determines the current driving capability.
- Effective Length (L_{eff}) is slightly less than drawn length (L_{drawn}) due to lateral diffusion (L_D) of source/drain implants.
- $L_{eff} = L_{drawn}$ - $2(L_D)$.

MOSFET Operation: Cutoff Region

- Condition: $V_{GS} < V_T$
- The surface is not inverted. No n-type channel exists to connect the n^+ Source and n^+ Drain.
- The Source-to-Drain path consists of two back-to-back pn junctions (Source-Body and Drain-Body).
- Drain Current $I_D = 0$ (ideally).
- The device acts as an open switch.

Diagram: MOSFET Operation: Cutoff Region

Applying Drain Voltage ($V_{DS} > 0$)

- Once $V_{GS} > V_T$, an inversion channel connects Source and Drain.
- Applying a positive V_{DS} creates a lateral electric field (ξ_y) along the channel.
- Electrons move from Source to Drain (current I_D flows from Drain to Source).
- The local channel potential $V(x)$ varies from 0 at the source to V_{DS} at the drain.

MOSFET Operation: Linear / Triode Region

- Conditions: $V_{GS} > V_T$ AND $V_{DS} < (V_{GS}$ - $V_T)$
- The effective gate overdrive $(V_{GS}$ - $V_T)$ is greater than V_{DS} everywhere along the channel.
- The channel remains continuous from source to drain.
- Current I_D increases roughly linearly with V_{DS}.
- The device acts as a voltage-controlled resistor (R_{on}).

Diagram: MOSFET Operation: Linear / Triode Region

Channel Pinch-Off

- The oxide electric field at any point x depends on V_{GS} - $V(x)$.
- At the drain end, the potential is V_{DS}. Thus, the effective gate voltage across the oxide is (V_{GS} - V_{DS}).
- As V_{DS} increases, the channel depth at the drain end decreases.
- When $V_{DS} = V_{GS}$ - V_T, the oxide voltage at the drain drops to exactly V_T.
- The inversion layer charge at the drain end reduces to zero. This is Channel Pinch-Off.

MOSFET Operation: Saturation Region

- Conditions: $V_{GS} > V_T$ AND $V_{DS} \geq (V_{GS}$ - $V_T)$
- The pinch-off point moves slightly toward the source, creating a depletion region near the drain.
- Electrons are swept across this depletion region by the high lateral electric field.
- Further increases in V_{DS} do not significantly increase I_D.
- The device acts as a voltage-controlled current source.

Diagram: MOSFET Operation: Saturation Region

Summary of I-V Regimes

- Define Overdrive Voltage: $V_{OV} = V_{GS}$ - V_T
- Cutoff: $V_{GS} < V_T$. Channel is OFF. $I_D = 0$.
- Linear: $V_{DS} < V_{OV}$. Channel is continuous. I_D is highly dependent on V_{DS}.
- Saturation: $V_{DS} \geq V_{OV}$. Channel is pinched off. I_D is independent of V_{DS} (ideal first-order model).

PMOS Transistor Operation

- Structure: n-type body, p^+ source and drain.
- Carriers: Current is conducted by holes, not electrons.
- Voltages: V_T is negative. Requires $V_{GS} < V_T$ to turn ON.
- Current Flow: I_D flows from Source to Drain.
- Operation remains symmetrical, but polarities are reversed compared to NMOS.

Key Takeaways

- The four terminals (Gate, Source, Drain, Body) interact to control the state of the surface channel.
- Channel pinch-off is a localized loss of inversion charge at the drain due to high V_{DS}.
- The transistor transitions from a voltage-controlled resistor (Linear) to a voltage-controlled current source (Saturation) upon pinch-off.
- NMOS and PMOS operate on symmetrical physics but with opposite voltage and current polarities.

Next Lecture Preview

- We have qualitatively described the channel and the I-V curves.
- Next lecture, we will introduce the Gradual Channel Approximation.
- We will formally derive the exact mathematical equations governing Drain Current (I_D) in both linear and saturation regions.
- We will analyze channel charge density mathematically.

Lecture 1.3: Channel formation and Gradual channel Approximation

Lecture 1.3: Channel formation and Gradual channel Approximation

Channel formation and Gradual channel Approximation

- Unit 1: MOS TRANSISTOR
- Lecture 3: Mathematical Foundations of Channel Conductivity
- Focus: Electrostatic assumptions, 1D modeling, and deriving inversion charge density.

Agenda

- Review of Inversion Charge Requirements
- The Dimensional Problem in MOSFET Electrostatics
- The Gradual Channel Approximation (GCA) Defined
- GCA Assumptions and Validity
- Defining Local Channel Potential, V(y)
- Formulating the Inversion Charge Density, $Q_I(y)$
- The Mobile Charge Profile along the Channel
- Drift Current Equation Setup

The Electrostatic Problem

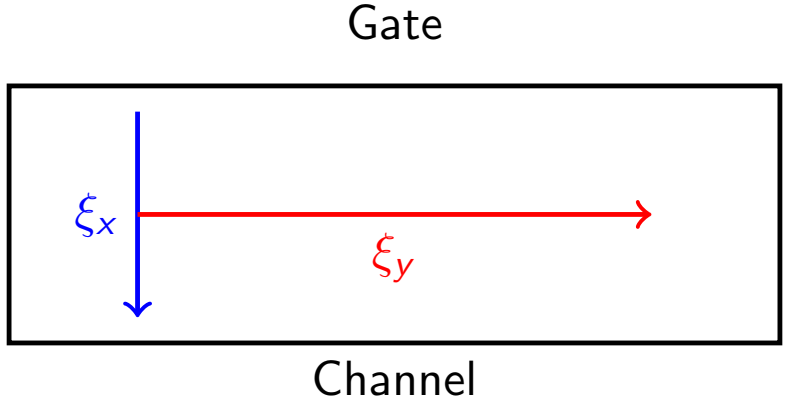
The Electrostatic Problem

- A MOSFET under bias represents a complex 2D (or 3D) Poisson boundary value problem.
- Vertical Electric Field (ξ_x): Created by V_{GS}. Responsible for depleting the bulk and inducing the inversion charge.
- Lateral Electric Field (ξ_y): Created by V_{DS}. Responsible for sweeping carriers from Source to Drain.
- Solving exact 2D Poisson equations analytically is impossible for general cases.

The Gradual Channel Approximation (GCA)

The Gradual Channel Approximation (GCA)

- Proposed by William Shockley.
- Core Assumption: The variation of the lateral electric field ($d\xi_y/dy$) is much smaller than the variation of the vertical electric field ($d\xi_x/dx$).
- Essentially: $\xi_x \gg \xi_y$.
- Allows the 2D problem to be decoupled into two independent 1D problems.

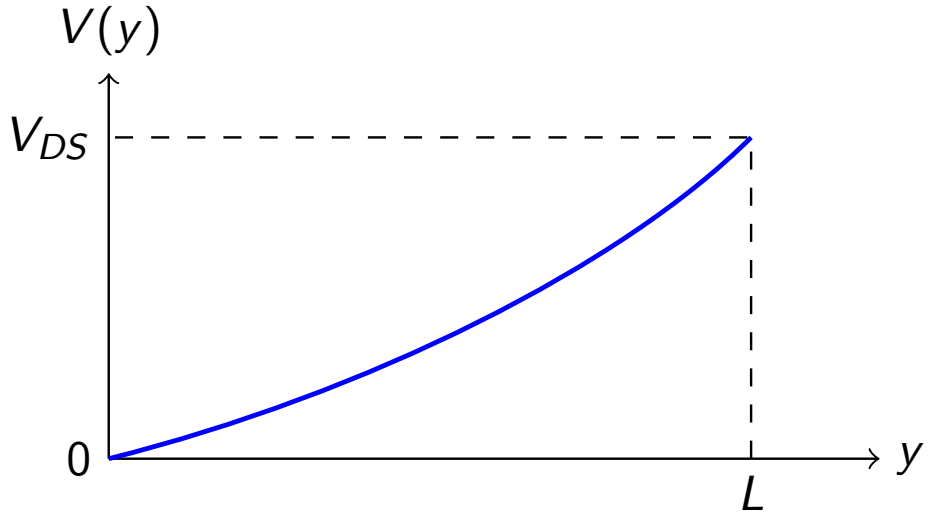


GCA Validity and Limits

- When is GCA valid?
  - Long channel devices (L $\gg$ oxide thickness).
  - When Source and Drain depletion regions are small compared to total channel length.
- When does GCA fail?
  - Deep sub-micron and nanoscale devices (Short Channel Effects).
  - When V_{DS} is very large relative to L, causing severe velocity saturation.

Local Channel Potential, V(y)

- Let y be the position along the channel from Source ($y = 0$) to Drain ($y = L$).
- $V(y)$ is the local potential of the channel at position y, measured relative to the Source.
- Boundary conditions:
  - At $y = 0$ (Source): $V(0) = 0$
  - At $y = L$ (Drain): $V(L) = V_{DS}$



Inversion Charge Density

Effective Gate Voltage at Position y

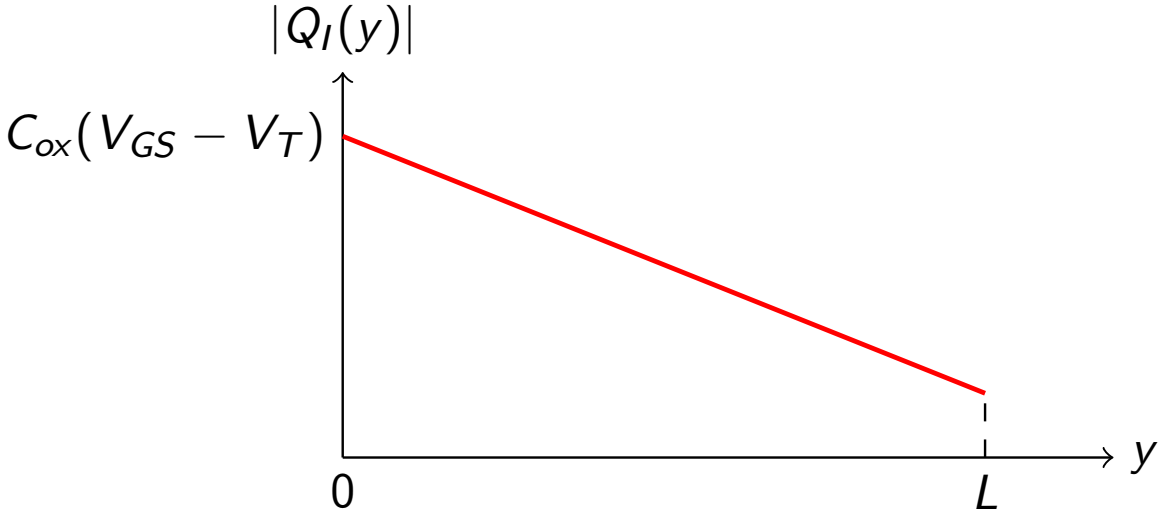
- The gate is held at a constant potential V_{GS} relative to the source.
- However, the channel beneath it has a varying potential $V(y)$.
- The effective voltage dropping across the oxide at position y is: $V_{ox}(y) = V_{GS} - V(y)$.
- This effective voltage is what induces the inversion charge locally.

Inversion Charge Density Equation

- Recall from MOS capacitor physics: Inversion charge Q_I is the total charge exceeding the threshold requirement.
- $Q_I(y) = -C_{ox} \cdot [V_{ox}(y) - V_T]$
- Substituting $V_{ox}(y)$:
- $Q_I(y) = -C_{ox} \cdot [V_{GS} - V(y) - V_T]$
- Units: Coulombs per square meter (C/m^2).

Analyzing the Charge Profile

- $Q_I(y) = -C_{ox}[(V_{GS} - V_T) - V(y)]$
- At the Source ($y = 0$): $V(0) = 0$. Charge is maximum: $Q_I(0) = -C_{ox}(V_{GS} - V_T)$.
- At the Drain ($y = L$): $V(L) = V_{DS}$. Charge is minimum: $Q_I(L) = -C_{ox}(V_{GS} - V_T - V_{DS})$.
- The charge density decreases linearly with increasing local potential $V(y)$.



Revisiting Pinch-Off Mathematically

- What happens when $V_{DS} = V_{GS} - V_T$?
- Evaluate Q_I at the drain ($y = L$):
- $Q_I(L) = -C_{ox}[(V_{GS} - V_T) - (V_{GS} - V_T)]$
- $Q_I(L) = 0$
- The math confirms the physical phenomenon: mobile charge density falls to zero at the drain.

Carrier Drift

Carrier Drift in the Channel

- Current in the channel is primarily drift current driven by the lateral electric field ξ_y.
- Electric field is the negative gradient of potential: $\xi_y = -dV(y)/dy$.
- Carrier drift velocity v_d is proportional to the field: $v_d = \mu_n \cdot \xi_y$
- Where μ_n is the surface electron mobility.

Setting up the Current Equation

- Current (I) is defined as the total charge crossing a plane per unit time.
- For a channel of width W, the current at any point y is:
- $I_D = -W \cdot Q_I(y) \cdot v_d(y)$
- Substitute $Q_I(y)$ and $v_d(y)$:
- $I_D = -W \cdot \{-C_{ox}[V_{GS} - V(y) - V_T]\} \cdot \{-\mu_n \cdot dV(y)/dy\}$

The Fundamental Differential Equation

- Simplifying the signs:
- $I_D = W \cdot \mu_n \cdot C_{ox} \cdot [V_{GS} - V_T - V(y)] \cdot (dV/dy)$
- By conservation of charge, DC current I_D must be constant everywhere along the channel (independent of y).
- This allows us to separate variables and integrate.

Summary

Summary of GCA Implications

- GCA decouples vertical (channel formation) and lateral (carrier transport) electrostatics.
- Allows us to define local charge Q_l as a simple function of local potential $V(y)$.
- Results in a single, integrable 1D differential equation for drain current.
- Valid only when lateral field gradients are small.

Key Takeaways

- The Gradual Channel Approximation relies on $\xi_x \gg \xi_y$ to simplify electrostatics.
- Local inversion charge $Q_I(y)$ tapers linearly with increasing local channel potential $V(y)$.
- The drift current is the product of channel width, local charge density, and drift velocity.
- The DC drain current I_D is constant at every cross-section of the channel.

Next Lecture Preview

- We have the differential equation set up.
- Next lecture, we will integrate this equation from Source to Drain.
- We will formally derive the exact current-voltage (I-V) characteristic equations for both Linear and Saturation regions.
- We will define the process transconductance parameter (k_n').

Lecture 1.4: MOSFET Current-Voltage characteristics

Unit 1: MOS TRANSISTOR

Lecture 4: Deriving the I-V Equations

Focus: Integration of the GCA equation, linear vs saturation models, and transconductance parameters.

Agenda

- Review of the GCA Differential Equation
- Separation of Variables and Integration Limits
- Derivation of the Linear Region Equation
- Defining Transconductance Parameters (k_n' and β)
- The Pinch-Off Condition Revisited
- Derivation of the Saturation Region Equation
- Channel Length Modulation (Early Effect)
- Plotting the Final I-V Characteristics

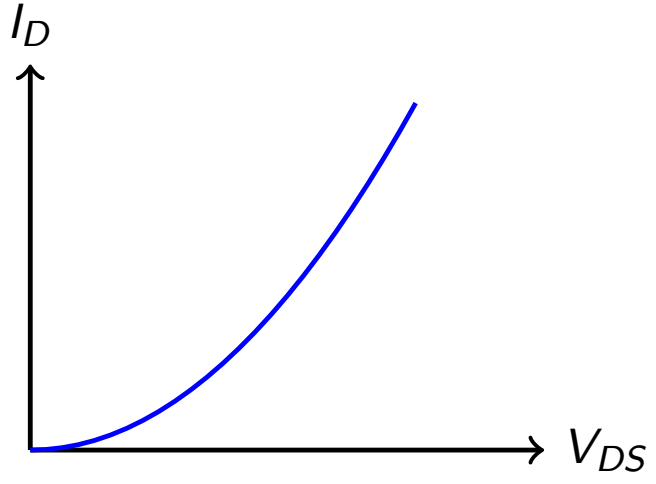
MOSFET Current-Voltage characteristics

The Starting Point

- Recall our differential equation from Lecture 3:
- $I_D \cdot dy = W \cdot \mu_n \cdot C_{ox} \cdot [V_{GS} - V_T - V(y)] \cdot dV$
- This equation represents the steady-state DC current at any arbitrary slice 'dy' of the channel.
- Since I_D is constant across the entire channel, we can integrate both sides over the channel geometry.

Integration Limits

- Spatial integration (Left Side):
  - From Source (y = 0) to Drain (y = L).
- Potential integration (Right Side):
  - At Source (y = 0), potential V = 0.
  - At Drain (y = L), potential $V = V_{DS}$.
- $\int_0^L I_D \, dy = \int_0^{V_{DS}} W \mu_n C_{ox} [V_{GS} - V_T - V] \, dV$



Performing the Integration

- Left side: $\int_0^L I_D \, dy = I_D \cdot L$
- Right side: $W \mu_n C_{ox} \int_0^{V_{DS}} (V_{GS} - V_T - V) \, dV$
- $= W \mu_n C_{ox} [(V_{GS} - V_T) V - \frac{V^2}{2}]_0^{V_{DS}}$
- $= W \mu_n C_{ox} [(V_{GS} - V_T) V_{DS} - \frac{V_{DS}^2}{2}]$

The Linear Region Equation

- Dividing by L gives the drain current equation for the Linear (Triode) region:
- $I_D = \mu_n C_{ox} \frac{W}{L}[(V_{GS} - V_T)V_{DS} - \frac{V_{DS}^2}{2}]$
- Valid strictly when $V_{GS} > V_T$ and $V_{DS} < (V_{GS} - V_T)$.
- Shows a parabolic relationship with V_{DS}.

Process and Device Parameters

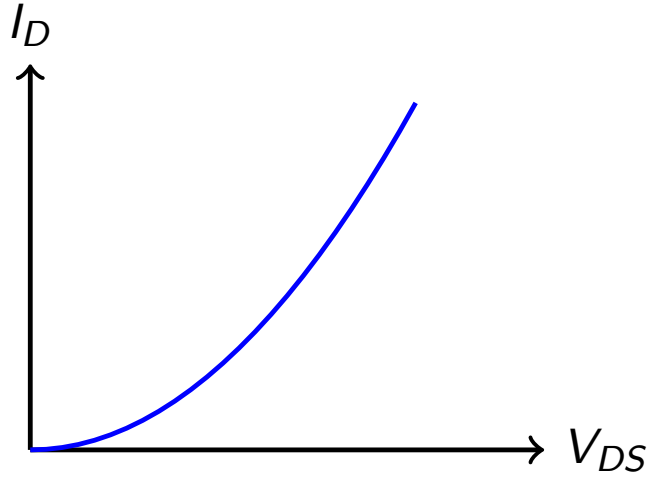
- To simplify equations, we define standard parameters:
- Process Transconductance Parameter (k'_n):
- $k'_n = \mu_n C_{ox}$ (Depends strictly on foundry technology)
- Device Transconductance Parameter (k_n or β):
- $k_n = k'_n \frac{W}{L} = \mu_n C_{ox} \frac{W}{L}$ (Depends on designer's layout)
- Linear Eq: $I_D = k_n[(V_{GS} - V_T)V_{DS} - \frac{V_{DS}^2}{2}]$

Deep Linear Region (Ohmic)

- When V_{DS} is very small ($V_{DS} \ll V_{GS} - V_T$):
- The $V_{DS}^2/2$ term becomes negligible.
- $I_D \approx k_n(V_{GS} - V_T)V_{DS}$
- The MOSFET behaves as a linear voltage-controlled resistor.
- $R_{ON} = \frac{V_{DS}}{I_D} = \frac{1}{k_n(V_{GS}-V_T)}$

Transition to Saturation

- As V_{DS} increases, the $V_{DS}^2/2$ term flattens the I-V curve.
- Pinch-off occurs exactly when $V_{DS} = V_{GS} - V_T = V_{OV}$.
- To find the saturation current, substitute $V_{DS} = (V_{GS} - V_T)$ into the linear equation.
- Beyond this voltage, the current becomes independent of V_{DS} (in the ideal model).



The Saturation Region Equation

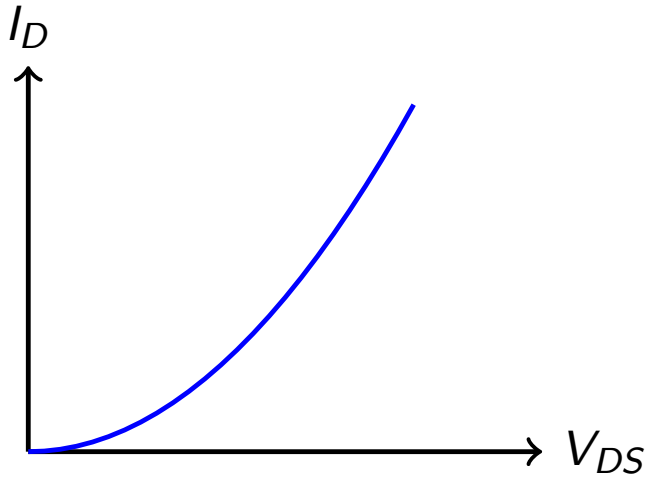
- Substitute $V_{DS} = (V_{GS} - V_T)$ into the linear equation:
- $I_{D,sat} = k_n[(V_{GS} - V_T)(V_{GS} - V_T) - \frac{(V_{GS}-V_T)^2}{2}]$
- $I_{D,sat} = k_n[(V_{GS} - V_T)^2 - \frac{1}{2}(V_{GS} - V_T)^2]$
- $I_{D,sat} = \frac{1}{2}\mu_n C_{ox}\frac{W}{L}(V_{GS} - V_T)^2$

The Saturation Region (Active Region)

- Equation: $I_D = \frac{1}{2} k_n (V_{GS} - V_T)^2$
- Valid strictly when $V_{GS} > V_T$ and $V_{DS} \geq (V_{GS} - V_T)$.
- The square-law dependence is the basis for most analog MOSFET amplifier design.
- I_D is controlled entirely by V_{GS}, making it an ideal transconductance amplifier.

Channel Length Modulation (CLM)

- Ideal model: I_D is constant in saturation.
- Reality: As V_{DS} increases beyond pinch-off, the depletion region at the drain expands.
- The pinch-off point moves towards the source, reducing the effective channel length ($L_{eff} < L$).
- Because $I_D \propto 1/L_{eff}$, current slightly increases with V_{DS}.
- This is the Early Effect in MOSFETs.



Modeling Channel Length Modulation

- We model CLM by adding a linear correction term to the saturation equation.
- $I_D = \frac{1}{2}\mu_n C_{ox}\frac{W}{L}(V_{GS} - V_T)^2(1 + \lambda V_{DS})$
- λ (Lambda) is the Channel Length Modulation parameter (Units: V^{-1}).
- λ is inversely proportional to channel length ($\lambda \propto 1/L$).
- Long devices have less CLM; short devices suffer heavily.

PMOS I-V Equations

- PMOS operates with holes (μ_p) and negative voltages.
- Define $V_{SG} = -V_{GS}$, $V_{SD} = -V_{DS}$, and $|V_T|$.
- Linear: $I_D = \mu_p C_{ox} \frac{W}{L}[(V_{SG} - |V_T|)V_{SD} - \frac{V_{SD}^2}{2}]$
- Saturation: $I_D = \frac{1}{2}\mu_p C_{ox} \frac{W}{L}(V_{SG} - |V_T|)^2(1 + \lambda V_{SD})$
- Note: hole mobility μ_p is typically 2x to 3x lower than electron mobility μ_n.

Key Takeaways

- Integration of the GCA yields the Linear and Saturation current equations.
- Linear region current depends on both V_{GS} and a quadratic V_{DS} term.
- Saturation region current depends ideally only on the square of $(V_{GS} - V_T)$.
- Channel Length Modulation causes a finite output resistance in saturation, modeled by λ.

Next Lecture Preview

- We have established the ideal equations for large devices.
- Next lecture, we will explore Device Scaling.
- Why do we want to make transistors smaller?
- We will examine Constant Voltage Scaling vs Full Scaling.
- We will discuss the negative side effects of scaling (Short Channel Effects).

Lecture 1.5: Needs of scaling and its Effects

Needs of scaling and its Effects

Needs of Scaling and its Effects

- Unit 1: MOS TRANSISTOR
- Lecture 5: MOSFET Scaling Theory
- Focus: Moore's Law, Constant-Voltage vs Full Scaling models, and physical limits.

Agenda

- The Motivation for Scaling (Moore's Law)
- The Scaling Factor ($S > 1$)
- Full Scaling (Constant-Field Scaling) Rules
- Impact of Full Scaling on Performance Metrics
- Constant-Voltage Scaling (CVS) Rules
- Impact of CVS on Performance Metrics
- Why CVS Fails: Reliability Limits
- Introduction to Short Channel Effects (SCE)
- Velocity Saturation and DIBL

The Motivation: Why Scale?

- 1. Integration Density: Smaller transistors mean more logic gates per unit area (cost reduction).
- 2. Speed / Performance: Shorter channel length (L) decreases carrier transit time ($\tau = L^2/\mu V_{DS}$).
- 3. Power Consumption: Smaller parasitic capacitances ($C_g \propto W \cdot L$) require less energy to charge and discharge.
- This tripartite benefit is the engine of Moore's Law.

The Scaling Factor (S)

- We define a dimensionless scaling factor, $S > 1$.
- Historically, $S \approx \sqrt{2} \approx 1.4$ per technology generation.
- This means every new node (e.g., 250nm to 180nm) reduces linear dimensions by $1/S$ (approx 0.7x).
- As a result, area ($L \times W$) scales by $1/S^2$ (0.5x), doubling transistor density.

Full Scaling (Constant-Field Scaling)

- Proposed by Robert Dennard (Dennard Scaling).
- Rule: Scale ALL physical dimensions AND ALL applied voltages by $1/S$.
- Dimensions: $L \to L/S$, $W \to W/S$, $t_{ox} \to t_{ox}/S$
- Voltages: $V_{DD} \to V_{DD}/S$, $V_T \to V_T/S$
- Result: Internal electric fields ($\xi = V/L$) remain strictly CONSTANT.

Parameter	**Original**	**Scaled by $1/S$**
Voltage (V)	V	V/S
Length (L)	L	L/S
Electric Field (ξ)	V/L	$\frac{V/S}{L/S} = V/L$

Effects of Full Scaling

- Let's derive the impact of Full Scaling on key metrics:
- Oxide Capacitance ($C_{ox} = \epsilon_{ox}/t_{ox}$): Scales by S.
- Gate Capacitance ($C_g = C_{ox} \cdot W \cdot L$): Scales by $(S) \cdot (1/S) \cdot (1/S) = 1/S$.
- Drain Current ($I_D \propto C_{ox} \frac{W}{L} V^2$): Scales by $(S) \cdot 1 \cdot (1/S^2) = 1/S$.
- Power Dissipation ($P = I \cdot V$): Scales by $(1/S) \cdot (1/S) = 1/S^2$.

Full Scaling: Power Density

- Power per transistor scales by $1/S^2$.
- Transistor density (number of transistors per unit area) scales by S^2.
- Power Density (Power per unit area of silicon):
- $(1/S^2) \cdot S^2 = 1$.
- Under ideal Dennard scaling, power density remains CONSTANT.

Why Full Scaling Stalled

- Full scaling requires scaling V_T down continuously.
- However, subthreshold leakage current (I_{off}) depends exponentially on $-V_T$.
- If V_T becomes too small ($< 0.3V$), the transistor cannot turn OFF.
- Leakage power explodes, destroying the constant power density paradigm.

Constant-Voltage Scaling (CVS)

- Historically used to maintain compatibility with 5V TTL logic standards.
- Rule: Scale dimensions (L, W, t_{ox}) by $1/S$. Keep Voltages (V_{DD}, V_T) CONSTANT.
- Electric Fields $(\xi = V/L)$: Scale by $V_{constant}/(L/S) = S$.
- Internal electric fields increase drastically with every generation.

Metric	**Full Scaling**	**CVS**
Dimensions (L, W)	$1/S$	$1/S$
Voltages (V)	$1/S$	1
Electric Field (ξ)	1	S

Effects of Constant-Voltage Scaling

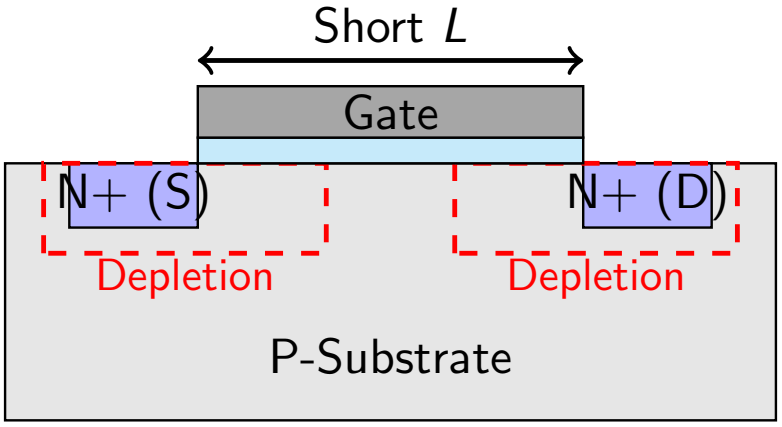
- Gate Capacitance (C_g): Scales by $1/S$.
- Drain Current (I_D): Scales by $(S) \cdot 1 \cdot (1) = S$. (Current actually increases!).
- Power Dissipation ($P = I \cdot V$): Scales by $(S) \cdot (1) = S$.
- Power Density: Scales by $(S) \cdot (S^2) = S^3$.

Reliability Limits of CVS

- The drastically increased electric fields ($\xi \times S$) cause physical damage:
- 1. Oxide Breakdown: High vertical fields punch through the thin SiO2.
- 2. Hot Carrier Injection (HCI): High lateral fields give electrons enough kinetic energy to jump the barrier into the oxide, becoming permanently trapped and shifting V_T.

Introduction to Short Channel Effects (SCE)

- As L approaches the depletion width dimensions, the Gradual Channel Approximation (GCA) fails completely.
- The Source and Drain depletion regions now constitute a significant portion of the channel.
- The Drain voltage (V_{DS}) begins to severely interfere with the Gate's control over the channel.



SCE: Velocity Saturation

- Recall $v_d = \mu_n \cdot \xi_y$. Velocity is assumed proportional to field.
- At high lateral fields (typical in short channels), velocity cannot increase infinitely due to optical phonon scattering.
- Velocity hits a hard limit: $v_{sat} \approx 10^7$ cm/s.
- Current equation changes from quadratic (V_{GS}^2) to linear (V_{GS}).
- $I_{D,sat} = W \cdot C_{ox} \cdot v_{sat}(V_{GS} - V_T)$

SCE: DIBL and V_T Roll-off

- Drain-Induced Barrier Lowering (DIBL):
  - High V_{DS} lowers the potential barrier at the source.
  - The drain essentially 'helps' the gate turn the device on.
  - V_T becomes a function of V_{DS} (V_T decreases as V_{DS} increases).
- V_T Roll-off (Short Channel Effect):
  - As L decreases, the S/D depletion charge supports some of the inversion charge.
  - Less gate voltage is needed. V_T drops drastically at very short lengths.

Key Takeaways

- Scaling historically provided simultaneous benefits in density, speed, and power.
- Full (Dennard) scaling maintains constant electric fields but is limited by non-scalable subthreshold leakage.
- Constant-Voltage scaling causes catastrophic increases in power density and reliability issues (HCI, Oxide Breakdown).
- At nanoscale lengths, velocity saturation and 2D electrostatics (DIBL) invalidate the classical square-law models.

Next Lecture Preview

- We have seen the failure of the planar MOSFET at small scales.
- Next lecture, we will look at modern structural solutions.
- We will introduce the Dual Gate MOSFET.
- We will explore how adding a second gate reasserts electrostatic control over the channel and suppresses Short Channel Effects.

Lecture 1.6: Dual Gate MOSFET

Dual Gate MOSFETs: Beyond the Planar Limit

- Evolution of the MOS Structure
- The Concept of Dual Gate (DG) MOSFETs
- Overcoming Short Channel Effects (SCE)
- Advantages vs. Fabrication Complexities

Agenda

- Motivation: Limits of Planar Scaling
- Concept and Physical Structure of DG-MOSFET
- Electrostatic Control and SCE Mitigation
- Advantages of Dual Gate Architecture
- Modes of Operation: Symmetric vs. Asymmetric
- Limitations and Fabrication Challenges

The Root Problem: Loss of Gate Control

- In deep submicron planar MOSFETs, the drain electric field penetrates the channel region.
- This proximity degrades the gate's ability to modulate the channel charge (loss of electrostatics).
- Results in severe Short Channel Effects (SCE):
  - Threshold Voltage (V_{th}) Roll-off
  - Drain-Induced Barrier Lowering (DIBL)
  - High subthreshold leakage current (I_{off})
- Solution requires shielding the channel from the drain's influence.

Concept of the Dual Gate MOSFET

- Introduces a second gate electrode on the opposite side of the channel (usually bottom).
- Channel is sandwiched between Top Gate and Bottom Gate.
- Both gates strongly couple to the channel potential.
- Effectively doubles the gate-channel capacitance (C_{ox}).
- Significantly suppresses field penetration from the drain.

Structural Differences from Single Gate

Planar Bulk MOSFET:

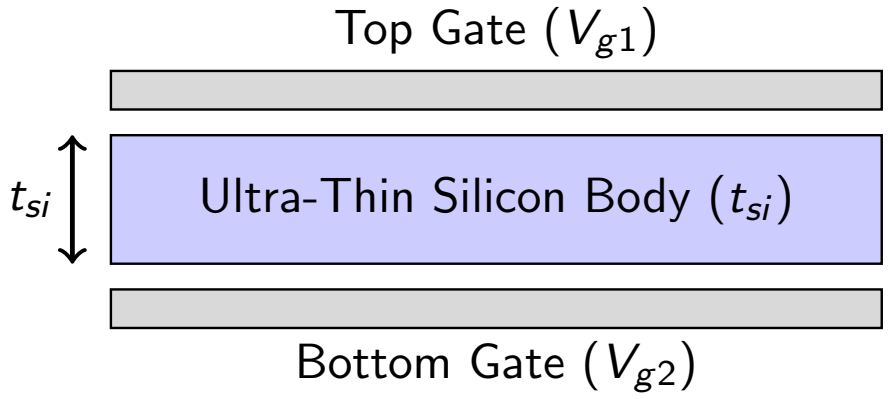
- One gate controls a thick silicon substrate.
- Depletion region extends deep into the bulk.
- Substrate terminal used for body-bias.

Dual Gate MOSFET:

- Two gates controlling an Ultra-Thin Body (UTB) channel.
- Fully Depleted (FD) operation; no neutral bulk region.
- Volume inversion possible due to dual electrostatic control.

Physics of Electrostatic Control

- Natural Length (λ) metric determines SCE immunity.
- For Planar: $\lambda \propto \sqrt{(\epsilon_{si}/\epsilon_{ox}) \cdot t_{ox} \cdot X_{dep}}$
- For Dual Gate: $\lambda \propto \sqrt{(\epsilon_{si}/2\epsilon_{ox}) \cdot t_{ox} \cdot t_{si}}$
- DG reduces λ, allowing shorter L_g without SCE.
- Requirement for good control: $L_g > 3 \times \lambda$



Volume Inversion in Dual Gate Devices

- Occurs when the silicon film (t_{si}) is sufficiently thin and both gates are biased simultaneously.
- Inversion layers from the top and bottom interfaces merge.
- Charge carriers flow through the entire volume of the silicon film, not just at the surface.
- Reduces surface scattering, thereby increasing effective carrier mobility (μ_{eff}).
- Increases total on-current (I_{on}).

Symmetric vs. Asymmetric Dual Gate

Symmetric Dual Gate:

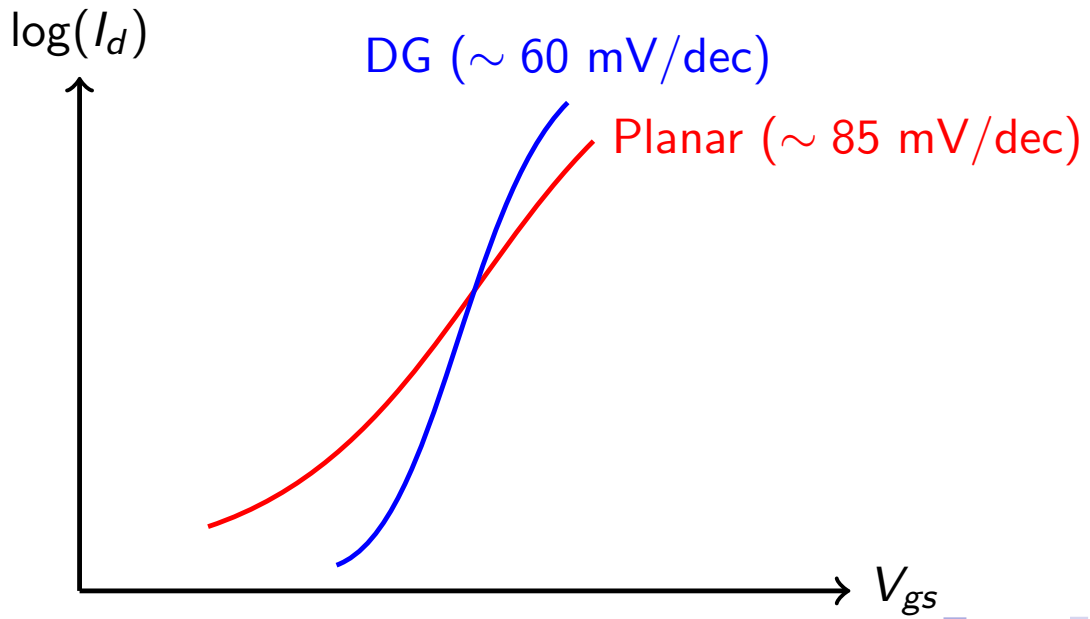
- Same work function for top and bottom gates.
- Both gates tied to the same input signal.
- Maximizes drive current and subthreshold slope.

Asymmetric Dual Gate:

- Different work functions (e.g., n+ poly and p+ poly) or different applied voltages.
- One gate drives the signal, the other shifts Threshold Voltage (V_{th}).
- Allows dynamic V_{th} tuning for low-power applications.

Advantages: Subthreshold Swing (SS)

- Ideal Subthreshold Swing (SS) at room temperature is 60 mV/decade.
- Planar MOSFETs rarely achieve this due to bulk capacitance (C_{dep}) dividing the gate voltage.
- DG-MOSFETs lack a neutral bulk; C_{dep} approaches 0.
- Result: $SS = (kT/q)\ln(10) \cdot (1 + C_{dep}/C_{ox}) \approx 60$ mV/dec.
- Provides sharper turn-off and drastically lower off-state leakage.



Advantages: DIBL and V_{th} Roll-off Suppression

- Dual gates screen the channel from the drain electric field.
- Drain Induced Barrier Lowering (DIBL) is severely minimized.
- Threshold voltage remains stable even as channel length (L_g) is scaled down.
- Eliminates the need for high channel doping.
- Lower doping reduces impurity scattering and threshold voltage variations.

Limitations and Fabrication Challenges

- Self-Alignment: Aligning the top and bottom gates perfectly is extremely difficult.
- Misalignment causes parasitic overlap capacitance (C_{gd}, C_{gs}) or underlap resistance.
- Creating an ultra-thin, uniform silicon body ($t_{si} < 10\text{nm}$) is challenging.
- Thickness variations in t_{si} directly cause V_{th} variations.
- Contacting the bottom gate requires complex 3D integration or etching.

Transition towards 3D Architectures

- Because planar bottom-gate fabrication is hard, the industry rotated the device 90 degrees.
- This rotation leads to the FinFET architecture.
- In a FinFET, the 'dual gates' are on the left and right sides of a vertical silicon fin.
- Gate self-alignment is naturally achieved by depositing the gate over the fin.
- Dual Gate theory perfectly describes FinFET electrostatics.

Summary of Dual Gate Characteristics

- Structure: Ultra-thin silicon body controlled by two gates.
- Electrostatics: Excellent control, minimizes Natural Length (λ).
- Performance: Ideal 60 mV/dec subthreshold swing, high mobility via volume inversion.
- Drawbacks: Extremely difficult planar fabrication and alignment.
- Legacy: The foundational theory for modern FinFET and GAA devices.

Lecture 1.7: Silicon on Insulator (SOI)

Advantages of SOI Summary

- **Reduced Junction Capacitance:** Faster switching speeds, lower dynamic power.
- **Absolute Latch-up Immunity:** No parasitic bipolar paths.
- **Lower Subthreshold Leakage:** Sharper subthreshold swing (in FD-SOI).
- **Radiation Hardness:** Reduced sensitive volume minimizes soft errors (SEU) from alpha particles/cosmic rays.

Limitations of SOI Summary

- **Wafer Cost:** Manufacturing SOI wafers (e.g., via SIMOX or Smart Cut) is expensive.
- **Floating Body Effect (PD-SOI):** Causes threshold voltage variations and the Kink Effect.
- **Self-Heating:** Poor thermal conductivity of the BOX requires careful thermal layout.
- **Complex Modeling:** Circuit models for PD-SOI must account for history effects of the floating body.

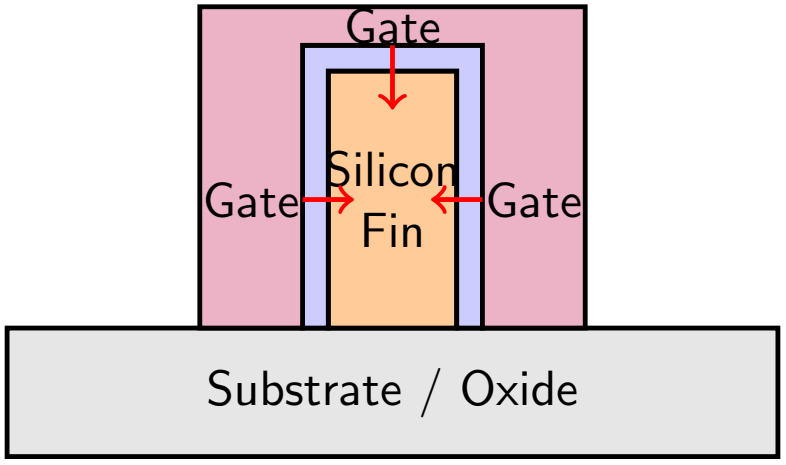
Lecture 1.8: High-K metal gate MOSFET and FIN-FET

Introduction to the FinFET

- A non-planar, 3D transistor architecture.
- Derived from the Dual-Gate concept.
- The silicon channel is raised into a thin, vertical 'Fin'.
- The gate electrode wraps around three sides of the fin (Top, Left, Right).
- First mass-produced by Intel at the 22nm node (Tri-Gate).

FinFET Electrostatics and SCE Immunity

- Gate controls the channel from three directions simultaneously.
- The fin body is ultra-thin (Fully Depleted).
- No path for the drain electric field to bypass the gate.
- Massive suppression of Drain Induced Barrier Lowering (DIBL).
- Near-ideal Subthreshold Swing (~ 60 mV/dec).



FinFET Channel Width Quantization

- In planar MOSFETs, channel width (W) is a continuous variable.
- In FinFETs, Effective Width (W_{eff}) per fin $= 2 \times H_{fin} + W_{fin}$.
- To increase drive current, designers must use multiple discrete fins in parallel.
- Width becomes quantized: $W_{total} = N \times W_{eff}$ (where $N = 1, 2, 3...$).
- This discrete sizing fundamentally changes standard cell layout and design rules.

FinFET Advantages vs. Planar

- Lower Subthreshold Leakage: Sharper turn-off allows lower threshold voltages.
- Higher Drive Current: Higher effective width per unit footprint area.
- Lower Operating Voltage (VDD): Due to improved electrostatics.
- Lower Doping Requirements: Fully depleted fin reduces random dopant fluctuation (RDF), improving V_{th} matching.

Limitations and Manufacturing Challenges of FinFETs

- High Aspect Ratio Etching: Etching tall, perfectly vertical, thin silicon fins is difficult.
- Surface Roughness: Etch damage on the vertical fin sidewalls degrades carrier mobility.
- Parasitic Resistance: The thin fin extensions leading to the source/drain contacts have high series resistance.
- Self-Heating: The 3D fin structure is surrounded by oxide, trapping thermal energy similar to SOI.
- Capacitance: Complex 3D fringe capacitances between gate and fin.

Summary

- High-K dielectrics solved gate tunneling leakage by increasing physical thickness while keeping EOT small.
- Metal gates eliminated poly-depletion and Fermi-pinning issues.
- FinFETs solved short-channel effects by wrapping the gate in 3D around an ultra-thin silicon fin.
- FinFET design introduces quantized width and complex 3D parasitic extraction challenges.

Lecture 1.9: Gate All Around (GAA) FET

Gate All Around (GAA) FETs

- Beyond FinFETs: The 3nm Node and Below
- Evolution of Electrostatic Control
- Nanosheets and Nanowires
- Ultimate SCE Immunity
- Manufacturing Challenges of GAA

Agenda

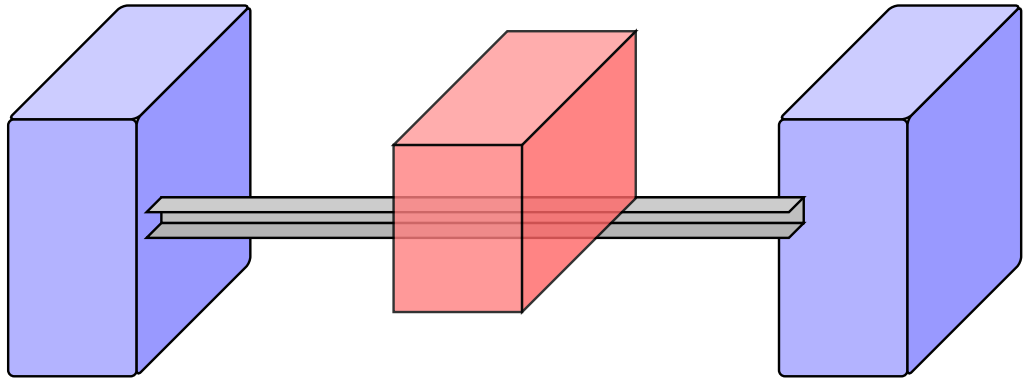
- The End of the FinFET Era
- Concept of Gate All Around (GAA)
- Nanowire vs. Nanosheet Architectures
- Electrostatic and Performance Advantages
- Regaining Continuous Width Sizing
- Key Fabrication Challenges: Epi, Release Etch, Inner Spacer

The Limits of FinFET Scaling

- In a FinFET, the gate wraps 3 sides, but the bottom of the fin remains connected to the substrate.
- As gate length (L_g) shrinks below 15nm, the drain electric field leaks through the bottom of the fin.
- To stop this, fins must be made extremely thin ($W_{fin} < 5nm$) and very tall.
- Fins thinner than 5nm suffer from severe surface roughness scattering, destroying mobility.
- Tall fins are mechanically unstable and prone to collapse during etching.

Concept of Gate All Around (GAA)

- Detach the channel from the substrate completely.
- Suspend the channel like a bridge between the Source and Drain.
- Wrap the High-K dielectric and Metal Gate 360 degrees around the channel.
- Provides perfect electrostatic control from all directions (Top, Bottom, Left, Right).
- Utterly blocks sub-surface leakage paths.



Nanowire vs. Nanosheet

Nanowire (NW-FET):

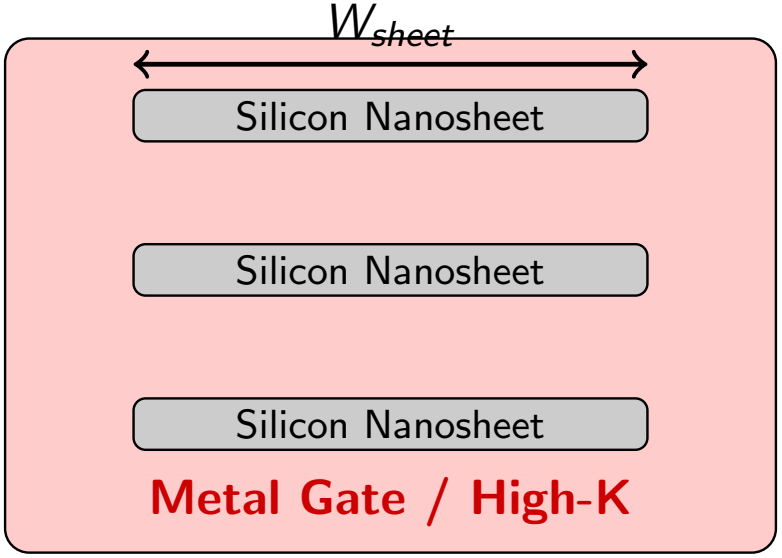
- Channel is a very thin cylinder or square of silicon.
- Excellent electrostatics, but very small cross-sectional area.
- Extremely low drive current (I_{on}) per wire.

Nanosheet (NS-FET) / MBCFET:

- Channel is flattened into a wide, thin ribbon.
- Maintains excellent vertical gate control.
- Wide sheet provides much higher drive current footprint.
- Adopted by major foundries (TSMC, Samsung, Intel) for 3nm/2nm.

Stacking for Density

- To maximize drive current per unit area, nanosheets are stacked vertically.
- Typically 3 to 4 silicon nanosheets are stacked on top of each other.
- The gate electrode weaves between the sheets.
- Total Effective Width $(W_{eff}) = N_{sheets} \times 2 \times (W_{sheet} + H_{sheet})$.
- Massive increase in current density compared to a single FinFET.

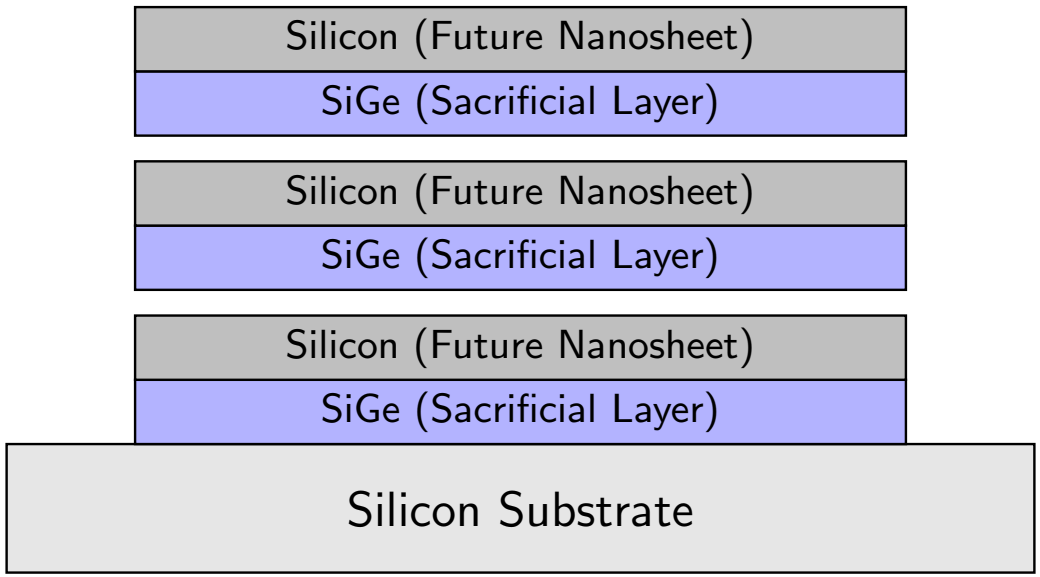


Regaining Continuous Width Sizing

- FinFET problem: Width is quantized (you must add discrete fins: 1, 2, 3...).
- GAA Nanosheet solution: The width of the nanosheet (W_{sheet}) can be drawn continuously in lithography.
- Designers can tailor the exact sheet width for specific cells.
- Wide sheets for High-Performance (HP) cells.
- Narrow sheets for High-Density, Low-Power (HD) cells.
- Brings back the layout flexibility of planar CMOS.

Fabrication Challenge 1: Epitaxial Stacking

- Building suspended sheets requires a sacrificial layer.
- Epitaxial growth of alternating Silicon (Si) and Silicon-Germanium (SiGe) layers.
- Si layers will become the final nanosheets.
- SiGe layers act as temporary mechanical placeholders.
- Must be grown with atomic-level uniformity.

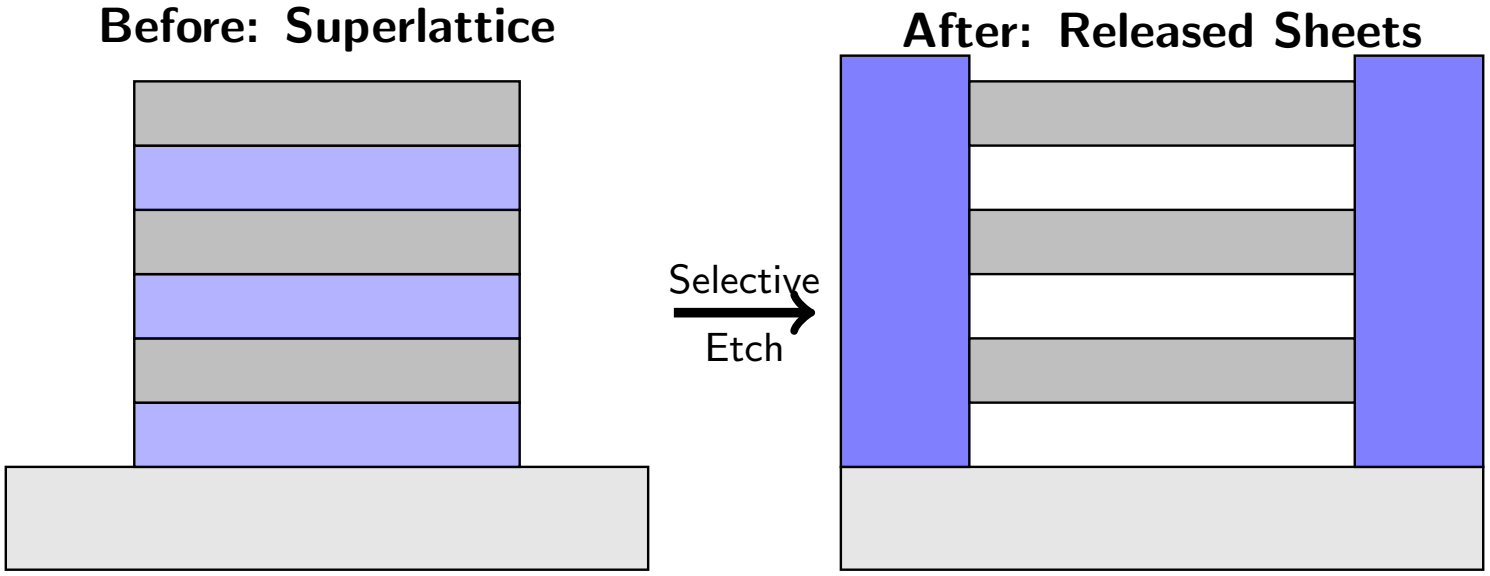


Fabrication Challenge 2: The Inner Spacer

- Before removing the SiGe, its edges must be etched back and filled with a dielectric (Inner Spacer).
- Purpose: To isolate the final metal gate from the Source/Drain epitaxy.
- Prevents massive parasitic Gate-to-Source/Drain capacitance (C_{gs}, C_{gd}).
- Requires extreme selectivity: etching SiGe laterally without damaging the adjacent Silicon.
- One of the most complex modules in the 3nm node.

Fabrication Challenge 3: The Release Etch

- The 'Channel Release' or 'Wire Release' step.
- A highly selective isotropic chemical etch removes all the SiGe sacrificial layers.
- Leaves the pure Silicon nanosheets suspended in mid-air (supported by the Source/Drain pillars).
- Etch selectivity between SiGe and Si must be $> 100:1$ to avoid thinning the delicate Si sheets.



Fabrication Challenge 4: Gate Stack Fill

- The tiny gaps between the suspended sheets (often $< 10nm$) must be filled.
- First, Atomic Layer Deposition (ALD) of interfacial oxide.
- Second, ALD of the High-K dielectric (HfO2).
- Third, ALD of Work Function Metals (TiN, TaN) to set V_{th}.
- If the gap pinches off before filling completely, a void is formed, destroying the transistor.

GAA Advantages Summary

- Unparalleled Electrostatics: Subthreshold swing approaches the theoretical limit.
- V_{th} Stability: Near-zero DIBL allows scaling to the lowest possible VDD.
- High Drive Current: Vertically stacked nanosheets provide massive W_{eff} in a tiny footprint.
- Design Flexibility: Continuous nanosheet width allows precise power/performance tuning.

The Future: Beyond GAA

- Backside Power Delivery Network (BSPDN): Routing power lines underneath the wafer to save routing area.
- CFET (Complementary FET): Stacking an NMOS nanosheet directly on top of a PMOS nanosheet.
- CFET folds the CMOS inverter into a single vertical footprint, effectively halving standard cell area.
- The continuous push to maintain Moore's Law into the Angstrom Era.

Lecture 2.1: Concept and working of Ideal Inverter

Ideal Inverter Concepts

Unit 2: MOS Inverters

- Introduction to Digital Logic Gates
- The Inverter (NOT Gate) as the Fundamental Building Block
- Concept of the Ideal Inverter
- Voltage Transfer Characteristics (VTC)
- Static and Dynamic Behavior

Agenda

- What is an Inverter?
- Logic Levels vs. Voltage Levels
- Properties of an Ideal Digital Inverter
- The Ideal Voltage Transfer Characteristic (VTC)
- Ideal Inverter Switching Threshold
- Power Dissipation in Ideal vs. Real Circuits

The Role of the Inverter

- Logical Function: Implements the Boolean NOT operation.
- Input A $\rightarrow$ Output Y $= \sim$A.
- Electrical Function: Acts as a voltage restorer and buffer.
- Must map analog voltages to strict digital logic levels.
- Logic '1' is represented by high voltage (V_{OH}, typically VDD).
- Logic '0' is represented by low voltage (V_{OL}, typically GND).

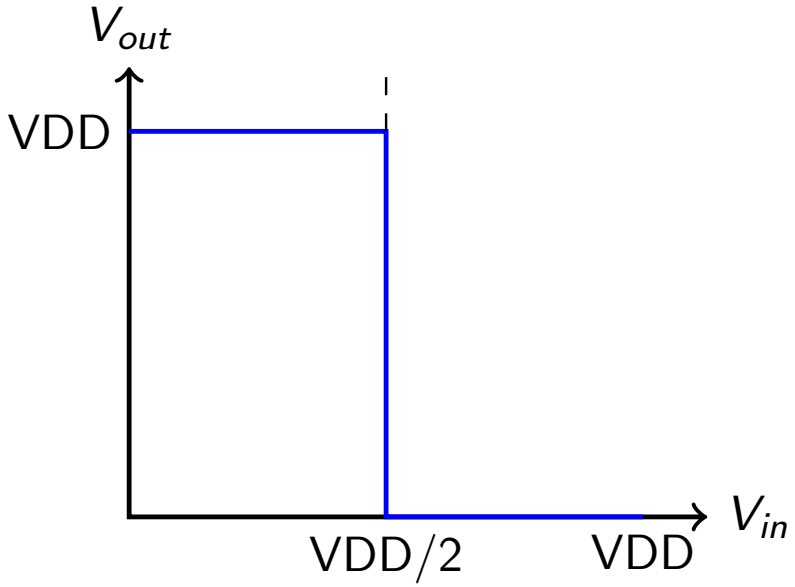
Defining the Ideal Inverter

An ideal inverter possesses mathematically perfect characteristics:

1. Infinite Voltage Gain ($\Delta V_{out}/\Delta V_{in} = -\infty$) at the switching threshold.
2. Sharp, instantaneous step transition between Logic 1 and Logic 0.
3. Zero static power dissipation (draws no current when holding a steady state).
4. Infinite Input Impedance (draws zero current from the driving source).
5. Zero Output Impedance (can drive infinite load capacitance instantaneously).

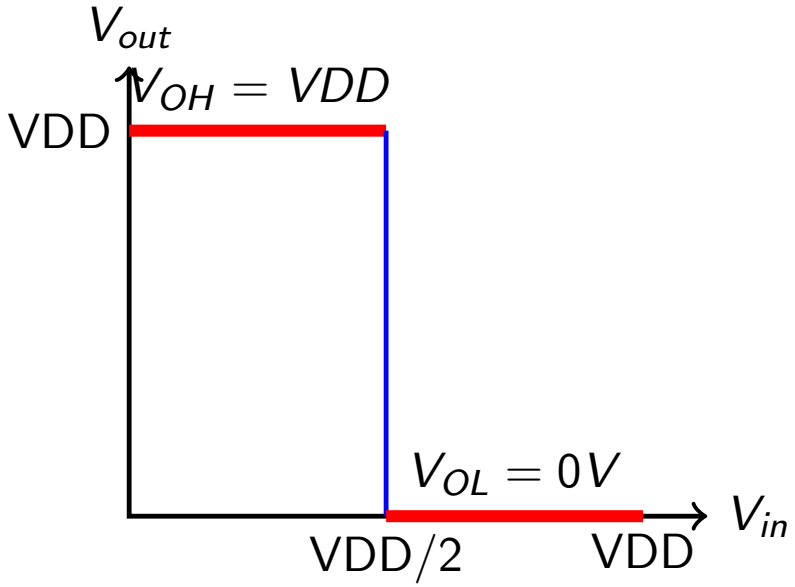
The Voltage Transfer Characteristic (VTC)

- The VTC is a plot of Output Voltage (V_{out}) versus Input Voltage (V_{in}).
- It characterizes the static, steady-state behavior of the inverter.
- X-axis: V_{in} sweeps from 0V to VDD.
- Y-axis: V_{out} responds accordingly.
- For an ideal inverter, the VTC is a perfect rectangular step function.



Ideal Logic Levels

- Output High Voltage (V_{OH}): The maximum output voltage. Ideal = VDD.
- Output Low Voltage (V_{OL}): The minimum output voltage. Ideal = 0V (GND).
- The ideal inverter swings rail-to-rail.
- Ensures maximum signal swing and unambiguous logic states for the next stage.

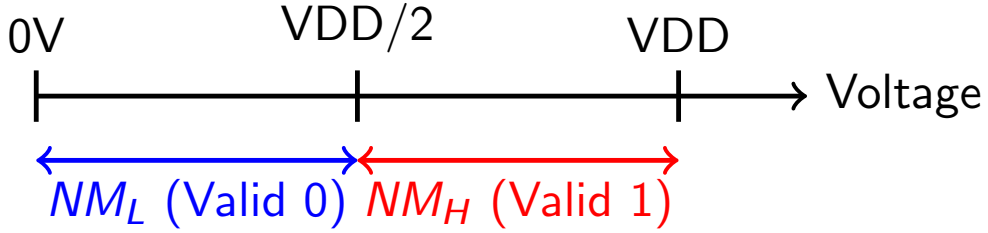


The Switching Threshold (V_{th})

- Also known as the Inverter Threshold Voltage or Inversion Point (V_M).
- The point where $V_{in} = V_{out}$.
- For an ideal inverter, this transition occurs exactly at VDD / 2.
- Provides perfectly symmetric noise margins.
- At $V_{in} = (VDD/2) - \epsilon$, $V_{out} = VDD$.
- At $V_{in} = (VDD/2) + \epsilon$, $V_{out} = 0V$.

Ideal Noise Margins

- Noise Margin High (NM_H): How much noise a '1' can take before being misread.
- Noise Margin Low (NM_L): How much noise a '0' can take before being misread.
- Ideal Inverter values:
- $NM_H = VDD - (VDD/2) = VDD/2$
- $NM_L = (VDD/2) - 0 = VDD/2$
- Maximum possible robustness against voltage spikes.

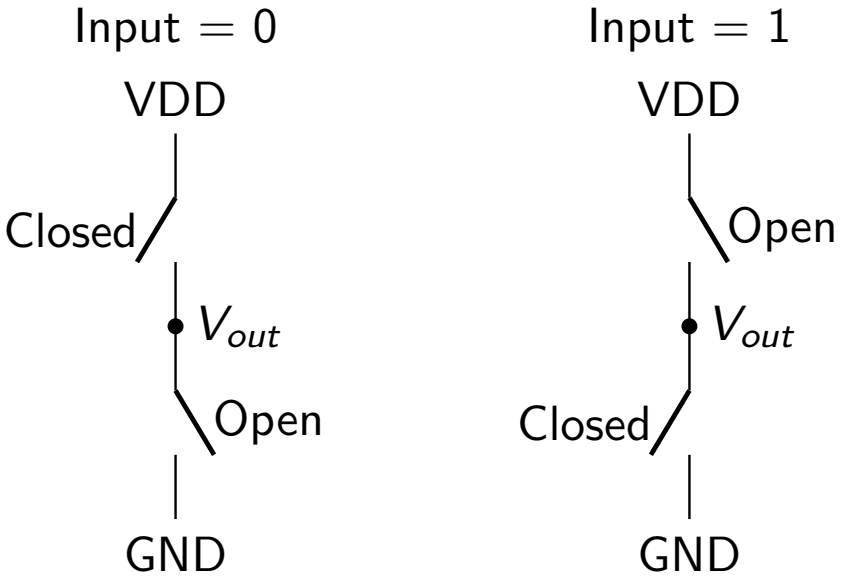


Ideal Transition Region and Gain

- Voltage Gain (g) $= \Delta V_{out}/\Delta V_{in}$ (the slope of the VTC).
- In the ideal inverter, the transition from V_{OH} to V_{OL} occurs instantly at V_M.
- The width of the transition region is zero.
- Therefore, the slope is a vertical line.
- Gain $= -\infty$.
- Real inverters have a finite transition region where both pull-up and pull-down networks are partially on.

Ideal Static Power Dissipation

- Static Power is consumed when the logic state is NOT changing.
- Ideal Inverter acts as a perfect switch: one path is always completely open (infinite resistance).
- When $V_{in} = 0$, Pull-down is open, $I_{static} = 0$.
- When $V_{in} = VDD$, Pull-up is open, $I_{static} = 0$.
- $P_{static} = VDD \times I_{static} = 0$ Watts.



Ideal Dynamic Performance

- Dynamic performance relates to switching speed (Propagation Delay).
- Time required to charge/discharge the load capacitor (C_{load}).
- Ideal Inverter has Zero Output Impedance ($R_{out} = 0$).
- Time Constant (τ) = $R_{out} \times C_{load} = 0$.
- Propagation Delay (t_p) = 0 seconds.
- Ideal inverter can operate at infinite clock frequencies.

Reality Check: Why Inverters are NOT Ideal

- Finite Gain: MOSFETs operate in saturation, not as perfect switches; VTC has a curved transition.
- Non-Zero R_{out}: MOSFETs have finite channel resistance (R_{on}), leading to RC delays.
- Non-Zero I_{static}: Subthreshold leakage and gate tunneling currents consume static power.
- Asymmetric Switching: PMOS and NMOS have different mobilities, often shifting V_M away from VDD/2.

Summary of Ideal Inverter Metrics

Parameter	**Ideal Value**
Output Swing	$V_{OH} = VDD$, $V_{OL} = 0V$ (Rail-to-Rail)
Switching Threshold	$V_M = VDD/2$
Noise Margins	$NM_H = NM_L = VDD/2$
Voltage Gain	$g = -\infty$
Static Power	$P_{static} = 0$
Propagation Delay	$t_p = 0$

Lecture 2.2: Noise Margin and Noise Immunity

Noise Margin and Noise Immunity

- Course Code: DI05000021
- Course Title: VLSI System
- Unit 2: MOS Inverters
- Lecture 11: Noise Margin and Noise Immunity

Agenda

- Introduction to Noise in Digital Circuits
- Voltage Transfer Characteristic (VTC) Recap
- Critical Voltage Points (VIL, VIH, VOL, VOH)
- Definition and Calculation of Noise Margins
- Noise Immunity Concept
- Impact of Cascaded Stages on Noise
- Summary and Key Takeaways

What is Noise in Digital Circuits?

- Noise: Unwanted variations in voltage or current in a circuit node.
- Sources of Noise:
  - Internal: Crosstalk, ground bounce, supply voltage variations, thermal noise.
  - External: Electromagnetic Interference (EMI), external radiation.
- In a digital system, logical '1' and '0' are represented by voltage ranges, not exact single values.
- If noise exceeds a certain threshold, the circuit may interpret a '0' as a '1' or vice versa, leading to logical errors.

The Voltage Transfer Characteristic (VTC)

- The VTC plots the Output Voltage (V_{out}) versus the Input Voltage (V_{in}).
- It visually describes how a logic gate maps an input signal to an output signal.
- For an inverter, the VTC starts at $V_{out} = V_{OH}$ when $V_{in} = 0$, and transitions to $V_{out} = V_{OL}$ as V_{in} increases.
- The transition region determines the gain of the inverter and its sensitivity to input variations.

Defining Critical Logic Levels

- To define logic ranges, we identify four critical points on the VTC where the small-signal gain is exactly -1 ($\frac{dV_{out}}{dV_{in}} = -1$).
- V_{IL}: Maximum input voltage recognized as a Logic '0'.
- V_{IH}: Minimum input voltage recognized as a Logic '1'.
- V_{OL}: Nominal output voltage for a Logic '0' state.
- V_{OH}: Nominal output voltage for a Logic '1' state.

Understanding Unity Gain Points

- The condition $\frac{dV_{out}}{dV_{in}} = -1$ separates the valid logic regions from the transition region.
- In the valid logic regions ($V_{in} < V_{IL}$ and $V_{in} > V_{IH}$), the gain magnitude is less than 1.
- Attenuation: Noise at the input is attenuated at the output, preventing noise amplification in cascaded gates.
- In the transition region ($V_{IL} < V_{in} < V_{IH}$), gain magnitude is greater than 1, meaning noise would be amplified.

Logic Levels and Valid Regions

- Logic '0' Input Range: $0 \leq V_{in} \leq V_{IL}$
- Logic '1' Input Range: $V_{IH} \leq V_{in} \leq V_{DD}$
- Undefined/Transition Region: $V_{IL} < V_{in} < V_{IH}$
- If a signal falls in the undefined region, the logic gate cannot guarantee whether it will be interpreted as a '0' or a '1'.

Definition of Noise Margin

- Noise Margin specifies the maximum amplitude of extraneous noise voltage that can be added to an input without compromising the output logic level.
- Low-Level Noise Margin (NM_L): The margin for a logic '0'.
- High-Level Noise Margin (NM_H): The margin for a logic '1'.
- Larger noise margins indicate a more robust circuit.

Calculating Low-Level Noise Margin (NM_L)

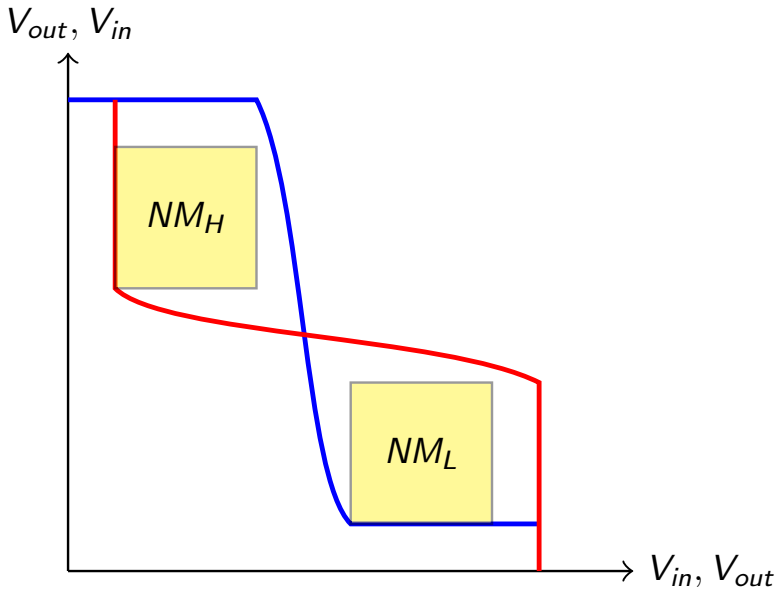
- Consider Gate 1 driving Gate 2 with a Logic '0'.
- Gate 1 outputs V_{OL}.
- Gate 2 requires an input voltage $\leq V_{IL}$ to properly recognize it as a '0'.
- The noise voltage can increase the signal line voltage.
- Maximum allowed positive noise: $NM_L = V_{IL} - V_{OL}$

Calculating High-Level Noise Margin (NM_H)

- Consider Gate 1 driving Gate 2 with a Logic '1'.
- Gate 1 outputs V_{OH}.
- Gate 2 requires an input voltage $\geq V_{IH}$ to properly recognize it as a '1'.
- The noise voltage can decrease the signal line voltage.
- Maximum allowed negative noise: $NM_H = V_{OH} - V_{IH}$

Graphical Interpretation on VTC

- By superimposing the normal VTC and a mirrored VTC (representing the cascaded gate), we create a 'butterfly curve'.
- The largest square that fits between the normal VTC and mirrored VTC in the logic '0' region represents NM_L.
- The largest square in the logic '1' region represents NM_H.
- This graphical method is often used in SRAM cell stability analysis (Static Noise Margin).



Noise Immunity vs. Noise Margin

- Noise Margin is a purely static DC metric, defined by steady-state voltage levels.
- Noise Immunity is a dynamic concept.
- A circuit might survive a very large noise spike if it is very short in duration (transient noise).
- Factors affecting dynamic noise immunity:
  - Gate capacitance (filters out high-frequency noise spikes)
  - Gate switching speed and delay

Ideal Noise Margins

- In a purely ideal inverter with V_{DD} supply:
- $V_{OL} = 0$, $V_{OH} = V_{DD}$
- The ideal VTC is a vertical step at $V_{in} = V_{DD}/2$.
- Thus, $V_{IL} = V_{DD}/2$ and $V_{IH} = V_{DD}/2$.
- $NM_L = V_{DD}/2 - 0 = V_{DD}/2$
- $NM_H = V_{DD} - V_{DD}/2 = V_{DD}/2$
- Maximum possible noise margins are equal to half the supply voltage.

Summary

- Noise Margins (NM_L and NM_H) quantify the maximum DC noise a gate can tolerate.
- Defined using critical points $V_{IL}, V_{IH}, V_{OL}, V_{OH}$ derived from the unity-gain points on the VTC.
- $NM_L = V_{IL} - V_{OL}$
- $NM_H = V_{OH} - V_{IH}$
- Symmetrical VTC curves generally provide the best overall noise margins.

Lecture 2.3: Voltage Transfer Characteristic of Ideal Inverter

Voltage Transfer Characteristic of Ideal Inverter

Agenda

- Recap: What is a VTC?
- Properties of an Ideal Inverter
- The Ideal VTC Curve
- Switching Threshold (V_{th})
- Transition Width in Ideal vs. Real Inverters
- Implications for Noise Margins
- Summary of Ideal Characteristics

Recap: What is a VTC?

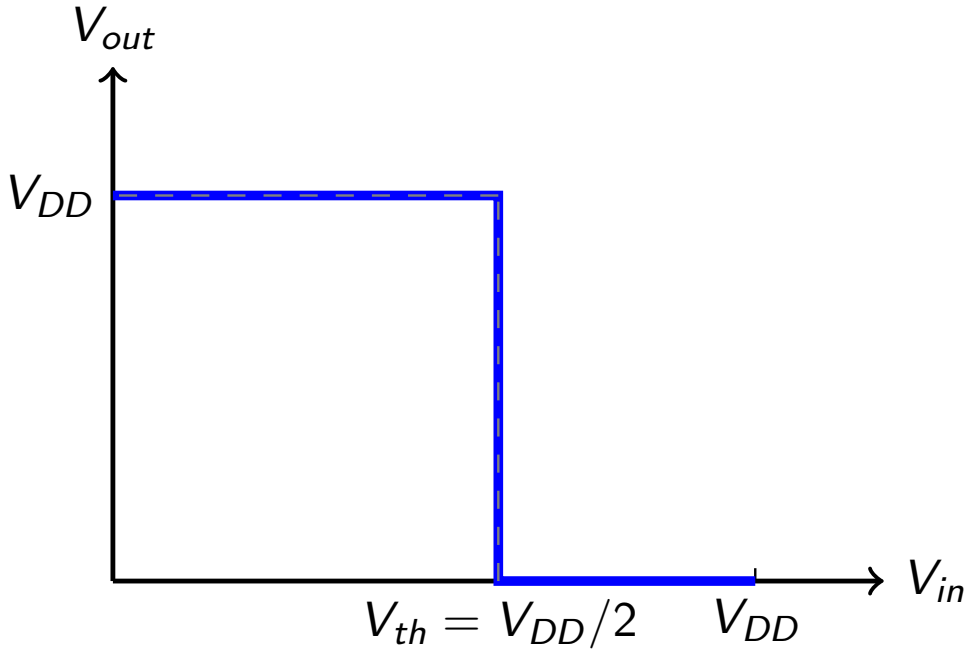
- Voltage Transfer Characteristic (VTC) is a static DC plot.
- It maps the steady-state output voltage (V_{out}) as a function of the input voltage (V_{in}).
- It determines fundamental digital gate properties: Logic levels, transition regions, and noise margins.
- A VTC ignores transient effects (capacitance, time delays).

Properties of an Ideal Inverter

- An ideal inverter perfectly digitizes analog voltage levels.
- Property 1: Infinite input impedance (draws zero current from the driver).
- Property 2: Zero output impedance (can drive any load without voltage drop).
- Property 3: Full output voltage swing ($V_{OL} = 0V$, $V_{OH} = V_{DD}$).
- Property 4: Instantaneous switching between logic states.

The Ideal VTC Curve

- The VTC of an ideal inverter is a perfect step function.
- For $V_{in} < V_{th}$, the output is strictly V_{DD}.
- For $V_{in} > V_{th}$, the output is strictly $0V$.
- At exactly $V_{in} = V_{th}$, the output transitions instantaneously from V_{DD} to 0.



Switching Threshold (V_{th})

- The switching threshold, V_{th} (also called logic threshold), is the point where the gate switches state.
- For a highly symmetric and ideal gate, $V_{th} = V_{DD}/2$.
- Why $V_{DD}/2$? Because it provides perfectly symmetrical noise margins.
- The ideal inverter spends zero time and zero voltage range in the 'undefined' state.

Transition Width in an Ideal Inverter

- Transition Width (TW) is defined as the voltage range between V_{IL} and V_{IH}.
- $TW = V_{IH} - V_{IL}$.
- In our ideal inverter, the switching is instantaneous at V_{th}.
- Therefore, $V_{IL} = V_{th}$ and $V_{IH} = V_{th}$.
- Result: The ideal Transition Width is identically ZERO ($TW = 0$).

Noise Margins of the Ideal Inverter

- Recalling the formulas:
- $NM_L = V_{IL} - V_{OL}$
- $NM_H = V_{OH} - V_{IH}$
- Plugging in ideal values ($V_{OL} = 0$, $V_{OH} = V_{DD}$, $V_{IL} = V_{IH} = V_{DD}/2$):
- $NM_L = V_{DD}/2 - 0 = V_{DD}/2$
- $NM_H = V_{DD} - V_{DD}/2 = V_{DD}/2$

Ideal vs. Real Inverter VTC

- Real inverters have finite gain in the transition region, meaning a non-zero Transition Width.
- Real inverters may not swing completely to $0V$ or V_{DD} (depending on the load topology).
- Real inverters have finite input/output impedances.
- The ideal VTC serves as the design target when sizing transistors in a CMOS inverter.

Implications of Finite Transition Width

- A non-zero TW means the gate has an 'undefined' logic region.
- If a signal gets stuck in the TW region, the output may sit at an intermediate voltage.
- Intermediate voltages cause static power dissipation in CMOS gates, as both NMOS and PMOS can be partially ON.
- Thus, minimizing TW is crucial for both noise immunity and power consumption.

Gain of the Ideal Inverter

- Small-signal voltage gain $g = \frac{dV_{out}}{dV_{in}}$
- For an ideal inverter, in the flat regions (Logic '0' and '1'), $g = 0$.
- This implies perfect attenuation of noise.
- At $V_{in} = V_{th}$, the gain approaches negative infinity ($g \to -\infty$).
- Infinite gain implies infinite sensitivity, forcing the immediate state change.

Design Goals Based on Ideal Inverter

- Goal 1: Maximize output swing ($V_{OH} \rightarrow V_{DD}$, $V_{OL} \rightarrow 0$).
- Goal 2: Center the switching threshold ($V_{th} \approx V_{DD}/2$).
- Goal 3: Maximize the magnitude of the gain in the transition region to narrow the TW.
- Goal 4: Optimize layout for minimal parasitic capacitance (to approach ideal switching speed).

Summary

- The ideal inverter provides the theoretical maximum performance limit.
- Its VTC is a perfect step function with a zero-width transition region.
- Ideal noise margins are perfectly symmetric and equal to $V_{DD}/2$.
- Real inverter designs aim to approximate this ideal step function by sizing transistors properly.

Lecture 2.4: Working and VTC of Resistive Load Inverter

Working and VTC of Resistive Load Inverter

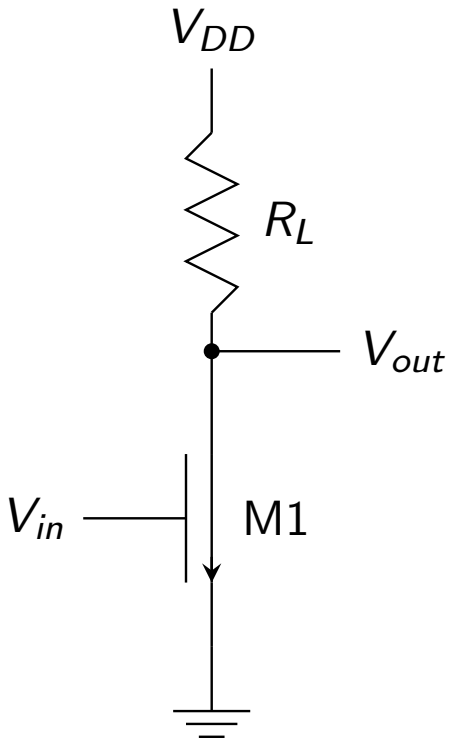
- Course Code: DI05000021
- Course Title: VLSI System
- Unit 2: MOS Inverters
- Lecture 13: Resistive Load Inverter

Agenda

- Circuit Topology of Resistive Load Inverter
- Working Principle: Input LOW (Logic 0)
- Working Principle: Input HIGH (Logic 1)
- Voltage Transfer Characteristic (VTC)
- Analysis of VOH and VOL
- The Load Resistance Trade-off: Power vs. Speed vs. Area
- Why Resistive Load is Obsolete in CMOS

Circuit Topology

- Driver: An enhancement-mode nMOS transistor.
- Load: A passive linear resistor (R_L).
- Input (V_{in}) is applied to the gate of the nMOS transistor.
- Output (V_{out}) is taken from the drain of the nMOS transistor.
- The source of the nMOS is grounded, and the top of R_L is tied to V_{DD}.



Operation: Input LOW (Logic '0')

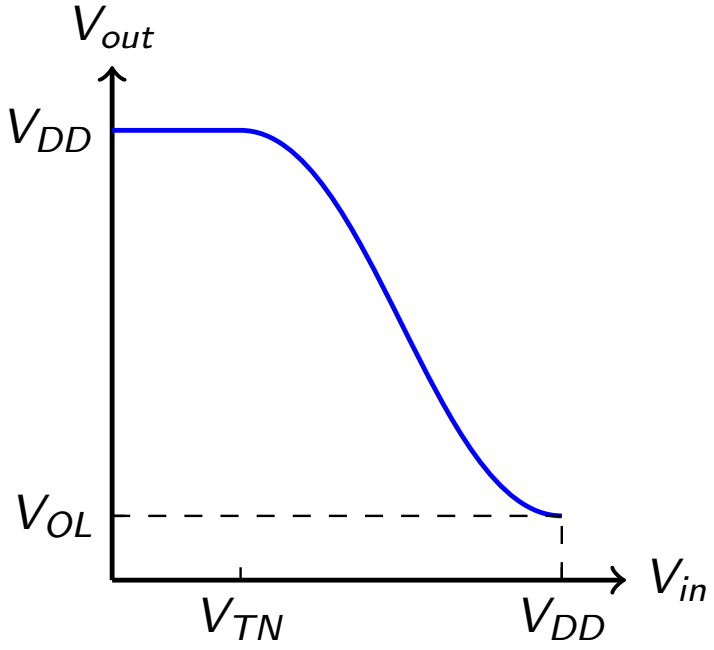
- Condition: $V_{in} < V_{TN}$ (where V_{TN} is the threshold voltage of the nMOS).
- The nMOS driver transistor is strictly OFF (Cutoff region).
- No current flows through the driver ($I_D = 0$).
- Because $I_D = 0$, there is no voltage drop across the load resistor R_L ($V_R = I_D \times R_L = 0$).
- Output voltage is pulled completely to the supply rail: $V_{out} = V_{DD}$.

Operation: Input HIGH (Logic '1')

- Condition: $V_{in} = V_{DD}$.
- The nMOS driver turns ON. Since V_{in} is high and V_{out} drops, the nMOS operates in the linear region.
- A steady-state DC current flows from V_{DD} through R_L and the nMOS to ground.
- The output voltage V_{out} is determined by the voltage divider formed by R_L and the ON-resistance of the nMOS (R_{on}).
- Result: $V_{out} = V_{OL} > 0V$.

Voltage Transfer Characteristic (VTC)

- The VTC plot deviates significantly from the ideal inverter.
- At $V_{in} = 0$, $V_{out} = V_{DD}$ (V_{OH}).
- As V_{in} increases past V_{TN}, the nMOS turns on in saturation, and V_{out} begins to fall.
- As V_{out} continues to fall, the nMOS enters the linear region.
- At $V_{in} = V_{DD}$, $V_{out} = V_{OL}$. Notice that V_{OL} is NOT zero.



Analysis of Output Low Voltage (V_{OL})

- To achieve a small V_{OL} (which is desired for a good logic '0'), the voltage drop across the nMOS must be very small.
- This requires $R_L \gg R_{on}$ of the nMOS.
- Since R_{on} is inversely proportional to the W/L ratio of the nMOS, we must either make R_L very large, or make the nMOS very wide.
- Equation governing V_{OL} involves equating the linear region current of nMOS with the current through the resistor.

Approximation of V_{OL}

$$V_{OL} \approx V_{DD}\left(\frac{R_{on}}{R_{on}+R_L}\right)$$

The Trade-off: Resistance and Area

- Problem 1: Creating a large resistor in silicon takes up a massive amount of chip area.
- A typical R_L needed might be $\approx 100k\Omega$.
- In a standard CMOS process, a $100k\Omega$ polysilicon resistor requires vastly more area than an entire transistor.
- This makes resistive load inverters highly impractical for dense VLSI integration.

The Trade-off: Resistance and Speed

- Problem 2: The pull-up time constant.
- When switching from low to high, the load capacitor C_{load} must be charged through R_L.
- Time constant $\tau_{pull-up} = R_L \times C_{load}$.
- If we make R_L very large to improve V_{OL} and reduce static power, the charging time τ becomes very large.
- Result: The inverter becomes extremely slow at pulling the output high.

The Trade-off: Static Power Dissipation

- Problem 3: Static power consumption.
- When the output is LOW, the nMOS is ON, and a steady current $I_{DC} = (V_{DD} - V_{OL})/R_L$ flows constantly from V_{DD} to ground.
- Static Power $P_{stat} = I_{DC} \times V_{DD}$.
- If we reduce R_L to make the circuit faster, the static power dissipation increases drastically.
- Millions of such gates would melt the chip.

Asymmetric Propagation Delay

- **Pull-down: Fast.** The nMOS can be made wide to provide a large discharge current for C_{load}.
- **Pull-up: Slow.** Dictated strictly by the passive RC time constant ($R_L \times C_{load}$).
- This fundamental asymmetry ($t_{pHL} \ll t_{pLH}$) makes timing analysis complex and limits the maximum clock frequency.

Summary

- The resistive load inverter uses an nMOS driver and a passive resistor load.
- It achieves an ideal $V_{OH} = V_{DD}$, but suffers from a non-ideal $V_{OL} > 0$.
- It presents severe, conflicting trade-offs between Area (large R_L), Speed (RC time constant), and Static Power.
- These limitations make it fundamentally unsuitable for modern VLSI design.

Lecture 2.5: Working and VTC of Depletion load Inverter - Part 1

Working and VTC of Depletion Load Inverter - Part 1

- Course Code: DI05000021
- Course Title: VLSI System
- Unit 2: MOS Inverters
- Lecture 14: Depletion Load Inverter (Part 1)

Agenda

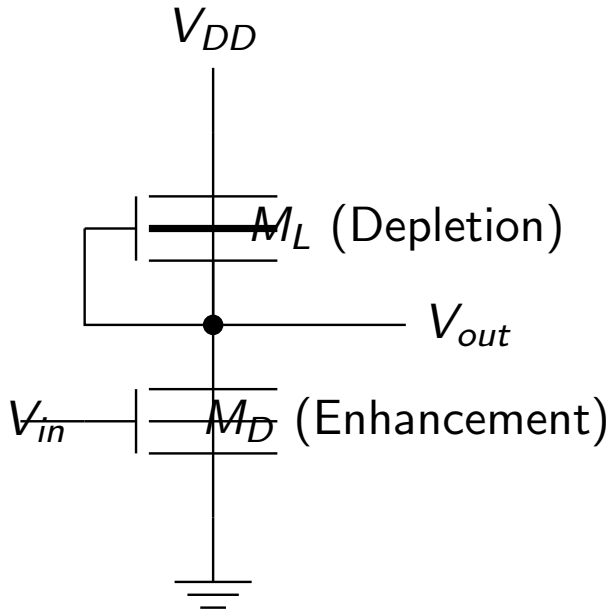
- The Area Problem of Resistive Loads
- Introducing the Active Load
- Circuit Topology of the Depletion-Load Inverter
- Characteristics of a Depletion-Mode nMOS
- Configuring the Load Transistor
- Operating Regions: Input LOW
- Operating Regions: Input HIGH
- Comparison with Resistive Load

The Need for an Active Load

- Recall: A passive resistor R_L takes up too much silicon area.
- Solution: Replace the passive resistor with a transistor.
- A transistor used in place of a resistor is called an 'Active Load'.
- An active load is much more compact than a passive resistor.
- It also offers a non-linear I-V characteristic that can improve switching speed.

Circuit Topology

- Driver: An enhancement-mode nMOS transistor (M_D).
- Load: A depletion-mode nMOS transistor (M_L).
- The gate of the load transistor M_L is tied directly to its own source.
- The source of M_L and drain of M_D are tied together to form the output node V_{out}.
- The drain of M_L is tied to V_{DD}.



What is a Depletion-Mode nMOS?

- Unlike an enhancement-mode device, a depletion-mode nMOS has a physically implanted channel.
- It is normally ON when $V_{GS} = 0V$.
- Its threshold voltage is negative: $V_{T,dep} < 0$.
- To turn it OFF, you must apply a negative gate-to-source voltage ($V_{GS} < V_{T,dep}$).
- In our inverter configuration, since gate is tied to source, $V_{GS,load} = 0V$ always.

The Load Transistor as a Current Source

- Since $V_{GS,L} = 0V$, and $0 > V_{T,dep}$, the load transistor is always conducting.
- It acts as a non-linear resistor or a constant current source, depending on its V_{DS}.
- Load $V_{DS,L} = V_{DD} - V_{out}$.
- If $V_{DS,L}$ is large (V_{out} is low), M_L is in saturation, acting as a constant current source.
- If $V_{DS,L}$ is small (V_{out} is high), M_L is in the linear region.

Operation: Input LOW (Logic '0')

- Condition: $V_{in} < V_{T,enh}$ (driver threshold).
- The driver transistor M_D is completely OFF.
- No current flows through the circuit ($I_D = 0$).
- The load transistor M_L must pull the output up.
- Since $I_D = 0$, M_L is in the linear region with zero voltage drop ($V_{DS,L} = 0$).
- Output voltage is fully restored to the rail: $V_{OH} = V_{DD}$.

Operation: Input HIGH (Logic '1')

- Condition: $V_{in} = V_{DD}$.
- The driver transistor M_D is heavily ON (Linear region).
- The load transistor M_L is also ON (Saturation region).
- A static DC current flows from V_{DD} to ground.
- The output voltage V_{out} drops to a low value (V_{OL}), but not perfectly zero.
- V_{OL} depends on the ratio of the aspect ratios (W/L) of M_D and M_L.

Sizing for V_{OL}

- To achieve a small V_{OL}, the driver must be much 'stronger' than the load.
- In MOSFETs, strength is determined by the Aspect Ratio ($Z = L/W$ or $\beta \propto W/L$).
- We define the inverter ratio $k_R = \frac{(W/L)_{driver}}{(W/L)_{load}}$.
- A typical ratio is $k_R \approx 4$.
- This is called a 'ratioed logic' family, because logic levels depend on transistor sizing.

Advantages over Resistive Load

- Area: A depletion load takes vastly less silicon area than a passive resistor.
- Speed: The constant current source behavior of the saturated load during pull-up charges C_{load} much faster than an RC exponential curve.
- Sharper VTC: The non-linear load line results in a steeper transition region in the VTC compared to the resistive load.

Remaining Disadvantages

- Static Power: Just like the resistive load, a DC current flows when the output is LOW. Static power dissipation is still a major issue.
- Asymmetric Delay: Pull-down is still faster than pull-up.
- Ratioed Logic Constraints: Requires careful sizing; minimal size transistors cannot be used everywhere.
- Manufacturing Complexity: Requires an extra ion implantation step to create the depletion channel, increasing fabrication cost.

Summary

- The depletion-load inverter uses a depletion-mode nMOS as an active load with $V_{GS} = 0$.
- It significantly reduces layout area and improves pull-up switching speed compared to passive resistors.
- It achieves an ideal $V_{OH} = V_{DD}$, but $V_{OL} > 0$ depends on careful transistor sizing (ratioed logic).
- It still suffers from continuous static power dissipation when output is LOW.

Lecture 2.6: Working and VTC of Depletion load Inverter - Part 2

Working and VTC of Depletion Load Inverter - Part 2

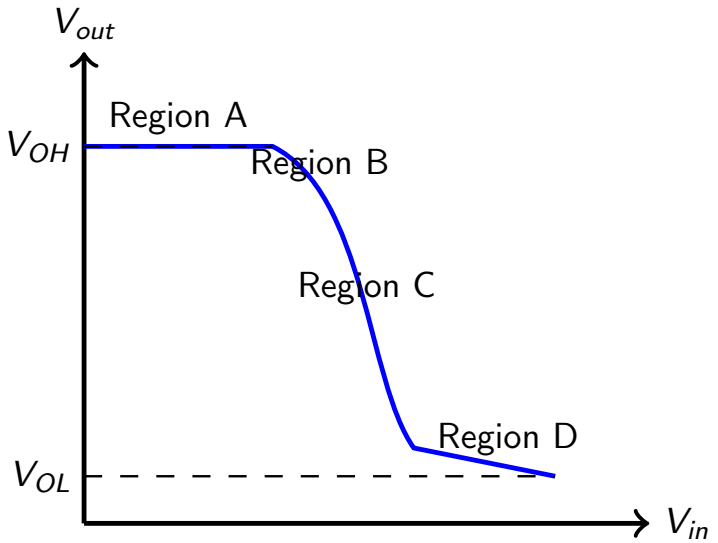
- Course Code: DI05000021
- Course Title: VLSI System
- Unit 2: MOS Inverters
- Lecture 15: Depletion Load Inverter (Part 2)

Agenda

- Recap: The Depletion-Load Topology
- The Depletion-Load VTC Curve
- Operating Regions across the VTC
- The Significance of the Z-Ratio (Z_{pu}/Z_{pd})
- Impact of Sizing on V_{OL} and Noise Margins
- Static Power Dissipation Analysis
- Dynamic Power Dissipation
- Transition to CMOS (Next Unit Preview)

The Depletion-Load VTC Curve

- The VTC shows a sharper transition than the resistive load inverter.
- Region A: V_{in} is low. Output is flat at $V_{OH} = V_{DD}$.
- Region B: Transition begins. Driver turns ON (Saturation), Load is in Linear region.
- Region C: Steepest transition. Both Driver and Load are in Saturation.
- Region D: V_{in} is high. Output flattens out at V_{OL}. Driver is in Linear, Load in Saturation.



Region C: The High Gain Region

- In Region C, V_{in} is near the switching threshold.
- Driver (M_D) is in Saturation because $V_{out} > V_{in} - V_{T,enh}$.
- Load (M_L) is in Saturation because $V_{DD} - V_{out} > -V_{T,dep}$.
- When both devices act as current sources, the circuit exhibits very high voltage gain.
- This high gain narrows the transition width, approaching the ideal VTC shape.

The Concept of Z-Ratio (Z_{pu}/Z_{pd})

- Transistor Aspect Ratio is defined as $Z = L/W$ (Length / Width).
- Note: Z is the inverse of the commonly used W/L ratio.
- Z_{pu}: Aspect ratio of the Pull-Up device (Load, M_L).
- Z_{pd}: Aspect ratio of the Pull-Down device (Driver, M_D).
- The Inverter Ratio $k_R = \frac{(W/L)_{driver}}{(W/L)_{load}} = \frac{Z_{pu}}{Z_{pd}}$.

Impact of Sizing on V_{OL}

- In Region D (Input HIGH), $V_{out} = V_{OL}$.
- Equating currents: Load (Saturation) = Driver (Linear).
- V_{OL} is inversely proportional to k_R (Z_{pu}/Z_{pd}).
- To lower V_{OL}, we must increase the ratio k_R.
- Typically, $k_R \approx 4$ yields an acceptable V_{OL} for safe logic '0' recognition.

Trade-offs in Inverter Sizing

- Why not make k_R infinitely large?
- Option 1: Make M_D very wide (Low Z_{pd}).
  - Consequence: Increases input gate capacitance, slowing down the previous stage.
- Option 2: Make M_L very long (High Z_{pu}).
  - Consequence: Decreases load current capability, making pull-up transition very slow.

Static Power Dissipation

- Static power is consumed when the circuit is NOT switching.
- When $V_{in} = 0V$ (Logic 0): $I_{DC} = 0$, therefore $P_{stat} = 0$.
- When $V_{in} = V_{DD}$ (Logic 1): $I_{DC} > 0$ because both M_L and M_D are ON.
- Average Static Power $P_{avg} = \frac{1}{2} \times I_{DC(on)} \times V_{DD}$ (assuming a 50% duty cycle).

Dynamic Power Dissipation

- Dynamic power is consumed during the transition between states.
- It takes energy to charge and discharge the load capacitance C_{load}.
- Energy drawn from supply to charge C_{load} is $C_{load} \times V_{DD}^2$.
- Half is stored in the capacitor, half is dissipated in the load transistor.
- During discharge, the stored energy is dissipated in the driver transistor.

Total Power Formula

- Total Power = Static Power + Dynamic Power
- $P_{total} = P_{static} + P_{dynamic}$
- $P_{total} = I_{avg} V_{DD} + f \times C_{load} \times V_{DD}^2$
- (where f is the switching frequency)
- In depletion-load logic, Static Power is usually the dominant factor, severely limiting the maximum number of gates per chip.

The Solution to Static Power

- To eliminate static power, we need a load that turns OFF when the driver turns ON.
- A depletion load cannot do this (it is always ON).
- Solution: Use a complementary device that acts as a switch controlled by the same input.
- This leads directly to the invention of CMOS (Complementary MOS), using a pMOS pull-up network.

Summary

- The VTC of a depletion-load inverter has a sharp transition region due to both devices saturating simultaneously.
- It is a ratioed logic circuit; proper logic levels require careful control of Z_{pu}/Z_{pd}.
- It suffers from asymmetric switching speeds and high static power dissipation.
- These limitations paved the way for CMOS technology.

Lecture 2.7: Working and VTC of CMOS inverter

Lecture 2.7: Working and VTC of CMOS inverter

Working and VTC of CMOS Inverter

- Working and VTC of CMOS inverter (without derivation)

Agenda

- Introduction to CMOS Logic
- CMOS Inverter Circuit Structure
- Steady-State DC Analysis
- Voltage Transfer Characteristic (VTC)
- The 5 Regions of Operation
- Noise Margins and Switching Threshold

Introduction to CMOS Inverter

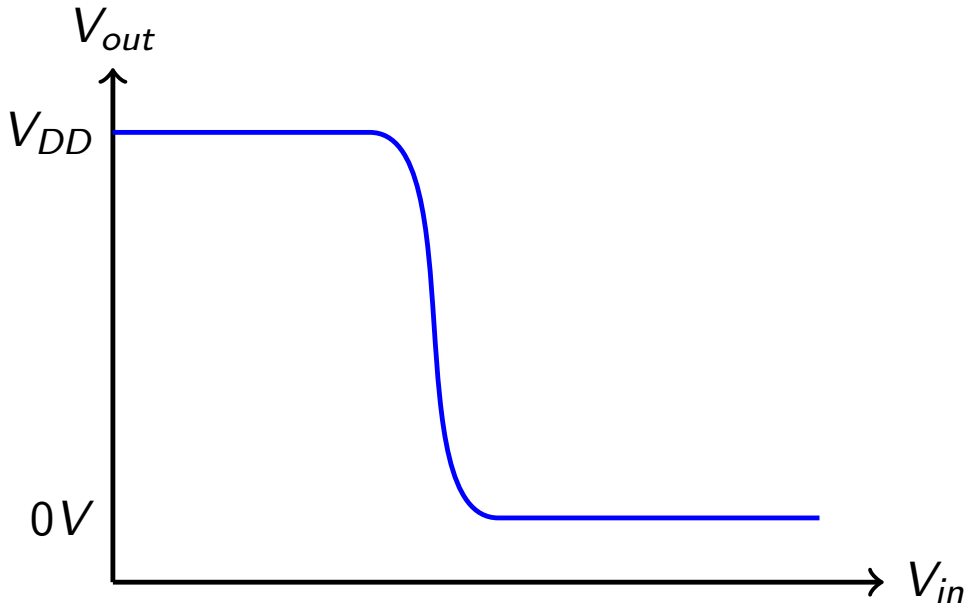
- Complementary MOS (CMOS) pairs an NMOS pull-down network with a PMOS pull-up network.
- PMOS is connected between VDD and the output node.
- NMOS is connected between the output node and Ground.
- Gates of both transistors are tied together to form the input node.

Zero Static Power Dissipation

- When Vin = 0V: NMOS is OFF, PMOS is ON. Output is pulled exactly to VDD.
- When Vin = VDD: NMOS is ON, PMOS is OFF. Output is pulled exactly to GND.
- In steady-state, one transistor is always OFF.
- No direct DC path exists between VDD and GND, yielding zero static power dissipation.

Voltage Transfer Characteristic (VTC) Overview

- VTC plots Vout versus Vin.
- Exhibits a sharp transition region near the logic threshold.
- Provides full rail-to-rail swing (VOH = VDD, VOL = 0V).
- The steep slope in the transition region yields high noise margins.



The 5 Regions of the VTC

- Region A: 0 $\leq$ Vin ¡ Vtn. (NMOS OFF, PMOS Linear)
- Region B: Vtn $\leq$ Vin ¡ Vinv. (NMOS Saturation, PMOS Linear)
- Region C: Vin = Vinv. (Both NMOS and PMOS in Saturation)
- Region D: Vinv ¡ Vin $\leq$ VDD - $|Vtp|$. (NMOS Linear, PMOS Saturation)
- Region E: Vin ¿ VDD - $|Vtp|$. (NMOS Linear, PMOS OFF)

Region A: Solid Logic HIGH

- Input Condition: $0 \leq$ Vin ¡ Vtn
- NMOS State: Vgs_n ¡ Vtn, so NMOS is Cut-off.
- PMOS State: Vsg_p ¿ $|Vtp|$, so PMOS is heavily ON.
- Vout = VDD. No current flows (Id = 0).

Region B: Transition Begins

- Input Condition: Vtn $\leq$ Vin ¡ Vinv
- NMOS turns ON and enters Saturation (Vds_n ¿ Vgs_n - Vtn).
- PMOS remains in Linear region (Vsd_p ¡ Vsg_p - $|Vtp|$).
- Vout begins to drop slowly as steady-state current starts to flow.

Region C: The Switching Threshold

- Input Condition: Vin = Vinv (Switching Threshold).
- Both NMOS and PMOS are in Saturation.
- Short-circuit current (Id) reaches its absolute maximum.
- Vout drops precipitously; the VTC slope is theoretically infinite in an ideal device.

Region D: Transition Ends

- Input Condition: Vinv ¡ Vin $\leq$ VDD - $|Vtp|$
- NMOS enters Linear region (Vds_n ¡ Vgs_n - Vtn).
- PMOS enters Saturation.
- Vout falls rapidly towards 0V.

Region E: Solid Logic LOW

- Input Condition: Vin ¿ VDD - $|Vtp|$
- PMOS State: Vsg_p ¡ $|Vtp|$, so PMOS is Cut-off.
- NMOS State: Strongly ON in Linear region.
- Vout = 0V. Id = 0.

Symmetric Inverter Design

- Goal: Vinv = VDD/2 for optimal noise margins.
- Requires matching the transconductance parameters: kn = kp.
- Since mobility of electrons (μ_n) is roughly 2-3x hole mobility (μ_p)...
- We must make the PMOS width (Wp) 2-3x larger than NMOS width (Wn).

Noise Margins in CMOS

- NML = VIL - VOL = VIL - 0 = VIL
- NMH = VOH - VIH = VDD - VIH
- For a symmetric inverter, NML = NMH $\approx$ VDD/2 - threshold.
- CMOS offers significantly higher noise margins compared to depletion load.

Summary of CMOS Advantages

- Full rail-to-rail output swing.
- Zero static power dissipation.
- Symmetrical VTC possible through transistor sizing.
- High noise margins.
- Extremely high input impedance.

End of Lecture

Next: Complex Logic Gates in NMOS

Lecture 3.1: Two Input NAND and NOR Gate with depletion NMOS load

Agenda

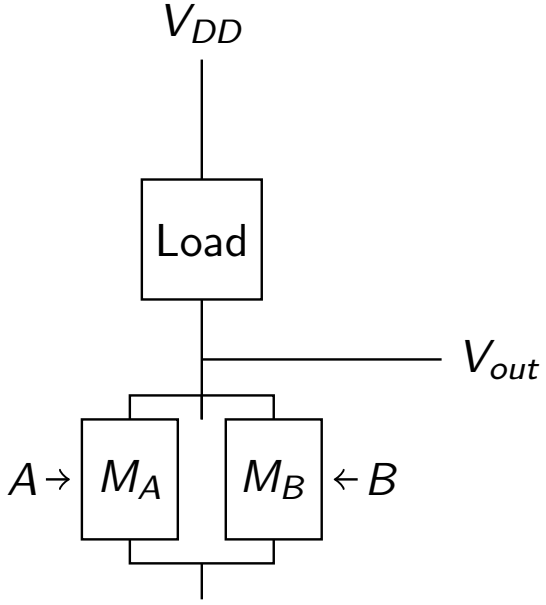
- Review of Depletion Load Logic
- The 2-Input NMOS NOR Gate Circuit
- NOR Gate: Operation and Sizing
- The 2-Input NMOS NAND Gate Circuit
- NAND Gate: Operation and Sizing
- Comparison: Area and Performance

Depletion Load Review

- A depletion-mode NMOS (with $V_t < 0$) acts as the pull-up load.
- Gate and Source are shorted ($V_{gs} = 0$), so it is always ON.
- Provides a constant-current-like source charging the output node.
- Logic gates are formed by replacing the single driver NMOS with a pull-down network.

Two-Input NMOS NOR Gate

- Pull-down network consists of two enhancement NMOS transistors in **PARALLEL**.
- Inputs A and B drive the gates of these transistors.
- Output is taken at the drain common node.



NOR Gate Logic Verification

- **A=0, B=0:** Both drivers OFF. Output pulled to V_{DD} by depletion load ($V_{OH} = V_{DD}$).
- **A=1, B=0:** Driver A is ON. Pulls output low to V_{OL}.
- **A=0, B=1:** Driver B is ON. Pulls output low to V_{OL}.
- **A=1, B=1:** Both drivers ON. Output pulled even lower.

A	**B**	**Output**
0	0	1 (V_{DD})
0	1	0 (V_{OL})
1	0	0 (V_{OL})
1	1	0 ($< V_{OL}$)

Sizing the NMOS NOR Gate

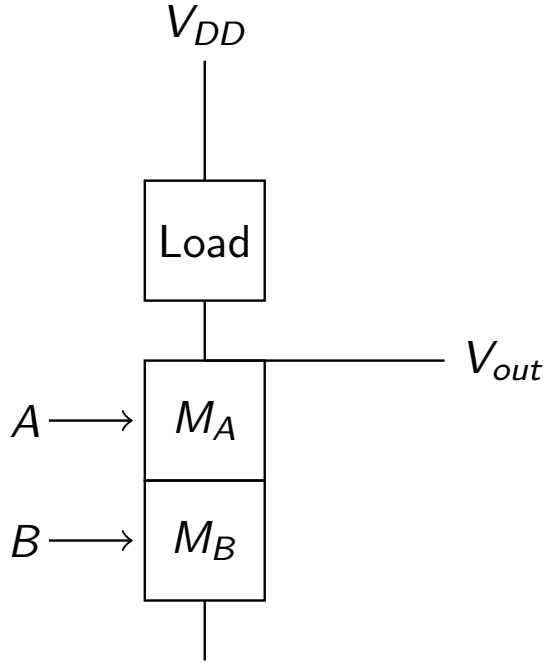
- The worst-case V_{OL} occurs when only **ONE** driver is ON.
- Therefore, each parallel driver must be sized identically to an equivalent inverter driver.
- $(W/L)_{driver} = (W/L)_{inverter}$
- Total area is relatively small: 1 load + 2 standard drivers.

Key Takeaway

NOR requires parallel drivers which are easy to size and do not severely penalize area.

Two-Input NMOS NAND Gate

- Pull-down network consists of two enhancement NMOS transistors in **SERIES**.
- Inputs A and B drive the gates of the top and bottom transistors.
- Both must be ON to provide a path to ground.



NAND Gate Logic Verification

- **A=0, B=0:** Both drivers OFF. Output V_{DD}.
- **A=0, B=1 or A=1, B=0:** One driver OFF, breaking the path. Output V_{DD}.
- **A=1, B=1:** Both drivers ON. Output is pulled down to V_{OL}.

A	**B**	**Output**
0	0	1 (V_{DD})
0	1	1 (V_{DD})
1	0	1 (V_{DD})
1	1	0 (V_{OL})

Sizing the NMOS NAND Gate

- Series transistors stack their ON-resistances (R_{on}).
- To achieve the same equivalent resistance (and same V_{OL}) as a standard inverter, drivers must be widened.
- For 2 inputs: $(W/L)_{driver} = 2 \times (W/L)_{inverter}$
- This significantly increases the silicon area.

Resistance Rule

Two identical transistors in series have double the resistance. To halve the resistance back to the original level, their width must be doubled.

Body Effect in Series NMOS

- The top NMOS in the series chain has its source elevated above ground ($V_S > 0$).
- $V_{sb} > 0$ leads to the **Body Effect**, increasing the threshold voltage (V_{tn}).
- This requires even wider sizing to maintain adequate drive strength and a low V_{OL}.
- Body effect severely degrades NMOS NAND performance.

NAND vs NOR in NMOS Logic

Feature	NOR Gate	NAND Gate
Driver Config	Parallel	Series
Driver Size	$1 \times (W/L)_{inv}$	$N \times (W/L)_{inv}$
Body Effect	None (sources grounded)	Present on upper transistors
Preference	**Highly Preferred**	Less optimal for high fan-in

Design Ratio (k_R) Considerations

- The inverter ratio $k_R = \frac{(W/L)_{driver}}{(W/L)_{load}}$.
- For NOR: $k_{R\_NOR} = k_{R\_inv}$.
- For NAND: k_{R_NAND} requires individual drivers to have a ratio of $N \times k_{R\_inv}$ (where N is the number of inputs).

Cascading Logic Gates

- Output of an NMOS logic gate can directly drive the inputs of another.
- Ensure that V_{OL} of the driving gate is well below the V_{tn} of the driven gate.
- Static power is consumed whenever the output is LOW due to a direct path from V_{DD} to ground.

Summary

- NMOS NOR gates use parallel drivers; sized identically to an inverter.
- NMOS NAND gates use series drivers; must be widened to compensate for stacked resistance.
- NOR gates are preferred in NMOS logic due to better area efficiency and absence of body effect.

Next: Combinational MOS logic circuits with Depletion nMOS Loads

Lecture 3.2: Combinational MOS logic circuits with Depletion nMOS Loads

Summary

- Complex logic is implemented using series (AND) and parallel (OR) NMOS networks.
- Proper sizing requires identifying the worst-case series pull-down path.
- Deep series stacks must be avoided to minimize area, body effect, and delay.

End of Lecture

Next: Two input NAND and NOR Gate using CMOS logic

Lecture 3.3: Two input NAND and NOR Gate using CMOS logic

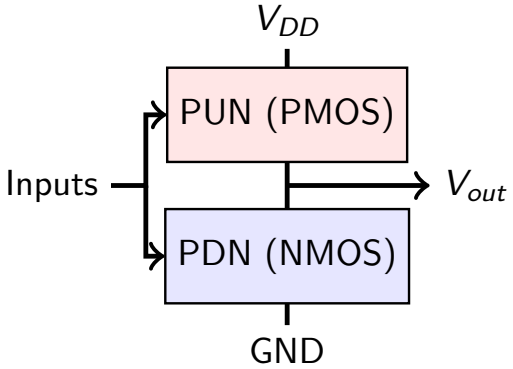
Agenda

- CMOS Logic Gate Philosophy
- Duality between Pull-Up and Pull-Down Networks
- The CMOS NAND Gate
- Sizing the CMOS NAND Gate
- The CMOS NOR Gate
- Sizing the CMOS NOR Gate
- NAND vs NOR in CMOS

Lecture 3.3: Two input NAND and NOR Gate using CMOS logic

The CMOS Logic Philosophy

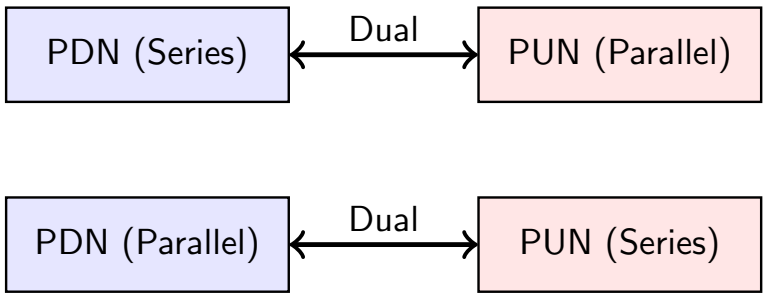
- A CMOS gate consists of two networks: Pull-Up Network (PUN) and Pull-Down Network (PDN).
- PUN consists solely of PMOS transistors, connected to VDD.
- PDN consists solely of NMOS transistors, connected to GND.
- The networks are MUTUALLY EXCLUSIVE: When PUN is ON, PDN is OFF, and vice versa.



Lecture 3.3: Two input NAND and NOR Gate using CMOS logic

Duality of Networks

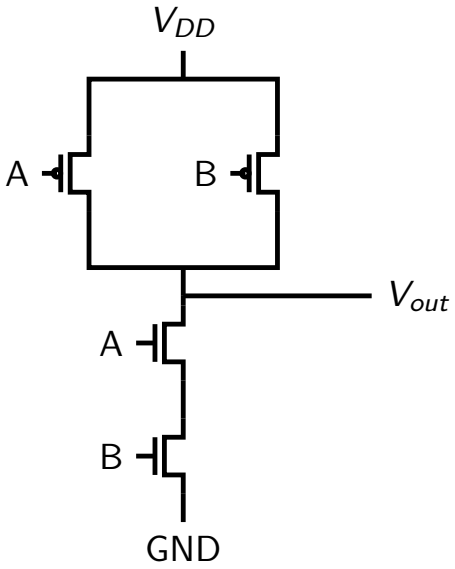
- If the PDN uses series connections (AND), the PUN must use parallel connections.
- If the PDN uses parallel connections (OR), the PUN must use series connections.
- This structural duality guarantees the mutually exclusive ON/OFF behavior.



Lecture 3.3: Two input NAND and NOR Gate using CMOS logic

CMOS NAND Gate Structure

- Logic: Output is LOW only when both A AND B are HIGH.
- PDN (NMOS): Two transistors in SERIES.
- PUN (PMOS): Two transistors in PARALLEL.



CMOS NAND Gate Verification

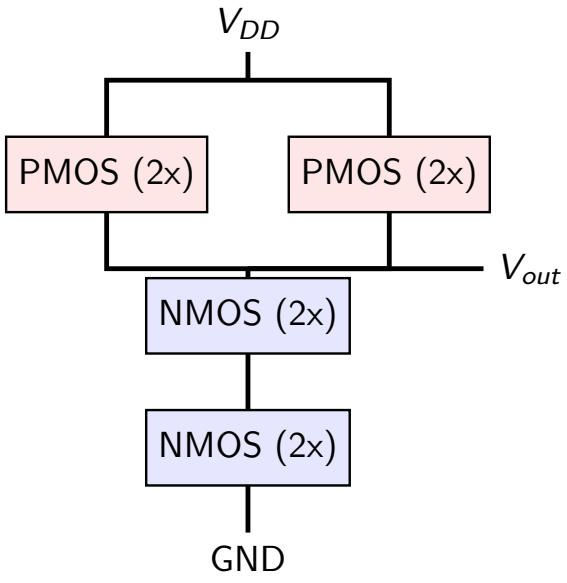
- A=0, B=0: Both NMOS OFF. Both PMOS ON. Vout = VDD.
- A=0, B=1: One NMOS OFF, breaking PDN. One PMOS ON, connecting Vout to VDD.
- A=1, B=1: Both NMOS ON, connecting Vout to GND. Both PMOS OFF, breaking PUN.

A	B	NMOS A	NMOS B	PMOS A	PMOS B	V_{out}
0	0	OFF	OFF	ON	ON	V_{DD}
0	1	OFF	ON	ON	OFF	V_{DD}
1	0	ON	OFF	OFF	ON	V_{DD}
1	1	ON	ON	OFF	OFF	GND

Lecture 3.3: Two input NAND and NOR Gate using CMOS logic

Sizing the CMOS NAND Gate

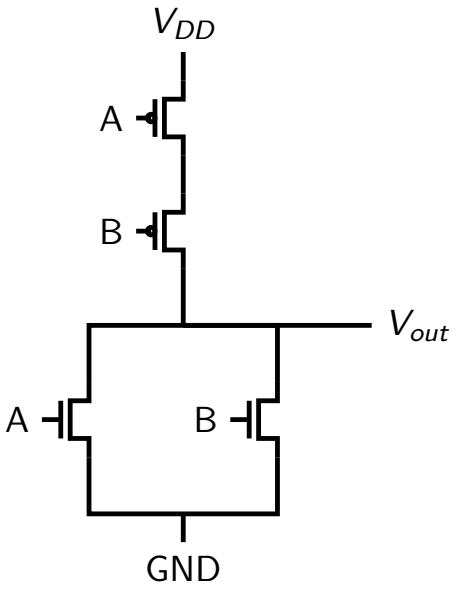
- Assume a symmetric inverter requires Wn = 1x, Wp = 2x.
- PDN (Series NMOS): Worst-case has 2 transistors in series. Wn must be $2 \times 1x = 2x$.
- PUN (Parallel PMOS): Worst-case has 1 transistor ON. Wp remains 2x.
- Result: Wn = 2x, Wp = 2x.



Lecture 3.3: Two input NAND and NOR Gate using CMOS logic

CMOS NOR Gate Structure

- Logic: Output is LOW if either A OR B is HIGH.
- PDN (NMOS): Two transistors in PARALLEL.
- PUN (PMOS): Two transistors in SERIES.



CMOS NOR Gate Verification

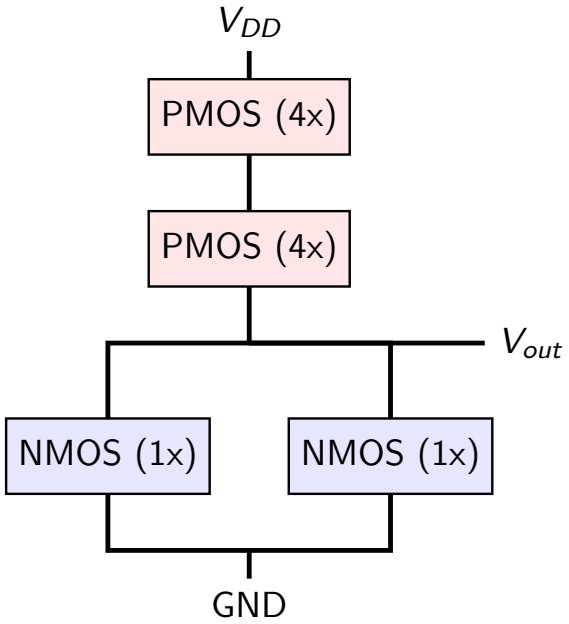
- A=0, B=0: Both NMOS OFF. Both PMOS ON. Vout = VDD.
- A=1, B=0: NMOS 'A' is ON. PMOS 'A' is OFF (breaking PUN). Vout = GND.
- A=1, B=1: Both NMOS ON. Both PMOS OFF. Vout = GND.

A	B	NMOS A	NMOS B	PMOS A	PMOS B	V_{out}
0	0	OFF	OFF	ON	ON	V_{DD}
0	1	OFF	ON	ON	OFF	GND
1	0	ON	OFF	OFF	ON	GND
1	1	ON	ON	OFF	OFF	GND

Lecture 3.3: Two input NAND and NOR Gate using CMOS logic

Sizing the CMOS NOR Gate

- Base inverter: Wn = 1x, Wp = 2x.
- PDN (Parallel NMOS): Worst-case has 1 ON. Wn remains 1x.
- PUN (Series PMOS): Worst-case has 2 transistors in series. Wp must be $2 \times 2x = 4x$.
- Result: Wn = 1x, Wp = 4x.



Lecture 3.3: Two input NAND and NOR Gate using CMOS logic

NAND vs NOR in CMOS

NAND Gate

- NMOS in series, PMOS in parallel.
- Max area factor: $2x$ (NMOS) + $2x$ (PMOS) $= 4$ units per input.

NOR Gate

- NMOS in parallel, PMOS in series.
- Max area factor: $1x$ (NMOS) + $4x$ (PMOS) $= 5$ units per input.

Why CMOS NAND is Preferred

- Smaller total silicon area.
- Lower input capacitance (due to smaller PMOS gate areas).
- Faster switching speeds because of reduced parasitic capacitances.
- Therefore, standard cell libraries are heavily NAND-based.

Multi-Input Considerations

- As fan-in increases to 3 or 4:
  - NAND: 3-input NAND requires Wp=2x, Wn=3x. (Manageable)
  - NOR: 3-input NOR requires Wn=1x, Wp=6x. (Huge area, very slow)
- Gates with ¿4 inputs are rarely implemented as single CMOS stages.

Lecture 3.3: Two input NAND and NOR Gate using CMOS logic

Summary

- CMOS logic uses mutually exclusive, dual PUN and PDN networks.
- NAND uses series NMOS / parallel PMOS.
- NOR uses parallel NMOS / series PMOS.
- **In CMOS, NAND is the preferred universal gate due to area and speed advantages.**

End of Lecture

Next: AOI and OAI complex Logic circuits using CMOS

Lecture 3.4: AOI and OAI complex Logic circuits using CMOS

Lecture 3.4: AOI and OAI complex Logic circuits using CMOS

Milav Dabgar

Lecturer, GP Palanpur

Table of Contents

Systematic Design of Complex CMOS

Systematic Design of Complex CMOS

- Step 1: Express the function in inverted form, e.g., Y = ~F(A,B,C...).
- Step 2: Use F to design the NMOS Pull-Down Network (PDN).
  - AND = Series, OR = Parallel.
- Step 3: Construct the PMOS Pull-Up Network (PUN) as the DUAL of the PDN.
  - Series PDN $\rightarrow$ Parallel PUN.
  - Parallel PDN $\rightarrow$ Series PUN.

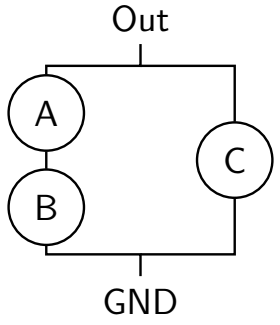
AOI Gates

What is an AOI Gate?

- AND-OR-INVERT (AOI).
- Implements sum-of-products logic directly, followed by an inversion.
- Example: AOI21 implements Y = $\sim$(A*B + C).
- The '21' denotes one 2-input AND term and one 1-input term, fed into an OR-INVERT.

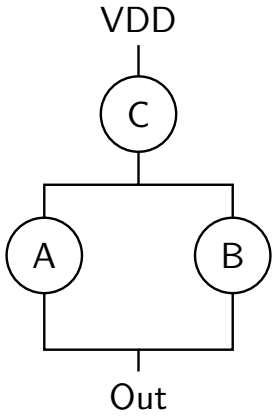
Designing the AOI21 Gate: PDN

- Function: Y = ∼(A*B + C)
- Inner function: F = A*B + C
- NMOS logic: A and B are ANDed → Series.
- The (A,B) branch is ORed with C → Parallel to C.



Designing the AOI21 Gate: PUN

- PUN must be the dual of the PDN.
- PDN has: (A series B) parallel with C.
- PUN must be: (A parallel B) series with C.
- Construct using PMOS transistors connected to VDD.



Sizing the AOI21 Gate

- Base inverter: Wn = 1x, Wp = 2x.
- PDN Worst Path: A-B series stack (2 devices) $\rightarrow$ Wn(A)=2x, Wn(B)=2x. Wn(C)=1x.
- PUN Worst Path: C-A or C-B series stack (2 devices) $\rightarrow$ Wp(A)=4x, Wp(B)=4x, Wp(C)=4x.

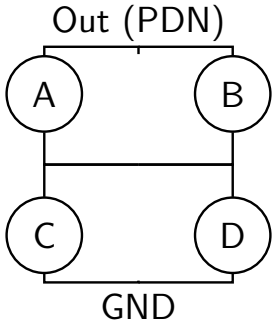
OAI Gates

What is an OAI Gate?

- OR-AND-INVERT (OAI).
- Implements product-of-sums logic directly, followed by an inversion.
- Example: OAI22 implements Y = $\sim$((A+B) * (C+D)).
- Highly efficient in CMOS due to structural characteristics.

Designing the OAI22 Gate

- Function: Y = $\sim$((A+B) * (C+D))
- PDN: (A parallel B) SERIES with (C parallel D).
- PUN (Dual): (A series B) PARALLEL with (C series D).



Sizing the OAI22 Gate

- PDN Worst Path: 2 series devices (e.g., A-C) $\rightarrow$ All NMOS = 2x.
- PUN Worst Path: 2 series devices (e.g., A-B or C-D) $\rightarrow$ All PMOS = 4x.
- Total area is well-balanced.

Why OAI is better than AOI in CMOS

- AOI gates tend to have more series PMOS devices if the logic has many AND terms.
- OAI gates push the series stacking into the NMOS network, and keep PMOS largely parallel.
- Since NMOS handles series resistance better (higher mobility), OAI gates are generally faster and smaller.

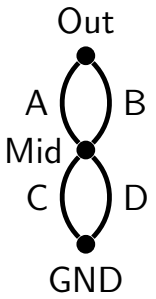
Euler Paths and Stick Diagrams

Layout Considerations: Stick Diagrams

- Direct translation of schematics to silicon requires efficient routing.
- Stick diagrams abstract the physical layers (Poly, Diffusion, Metal).
- Goal: Arrange transistors so the diffusion area (active region) is contiguous, eliminating metal interconnects between sources and drains.

The Euler Path Method

- Represent the PDN and PUN as mathematical graphs.
- Find a common Euler Path (a sequence that visits every edge/transistor exactly once) that is valid for BOTH networks.
- The sequence of the path dictates the physical ordering of the polysilicon gate inputs.



Summary

Summary

- Complex CMOS gates use dual PDN and PUN networks.
- AOI and OAI architectures allow multi-level logic in a single fast stage.
- OAI is often preferred because it avoids deep PMOS series stacks.
- Euler paths enable highly optimized, area-efficient physical layouts.

End of Lecture

Next: Simple XOR/XNOR function using CMOS logic

Lecture 3.5: Simple XOR/XNOR function using CMOS logic

Summary

- XOR and XNOR functions require complementary inputs in standard static CMOS.
- Static CMOS implementations yield 12 transistors total.
- Transmission gate implementations provide a functional area-saving alternative.
- Trade-offs exist between transistor count, signal swing, and noise margins.

Questions & Next Steps

- Any questions regarding XOR/XNOR topologies?
- Next Lecture: Complex logic circuits using CMOS logic.
- We will see how to implement arbitrary boolean expressions like $Y = \overline{A(B + C) + DE}$.

Lecture 3.6: Complex logic circuits using CMOS logic

Summary

- Complex CMOS gates efficiently implement arbitrary inverted Boolean functions.
- **PDN logic uses NMOS:** AND = series, OR = parallel.
- **PUN logic uses PMOS:** AND = parallel, OR = series (Duality).
- Euler paths are essential for area-efficient layout by maximizing diffusion sharing.
- Transistor sizing compensates for series resistance.

Questions & Next Steps

- Any questions on compound gate design or Euler paths?
- **Next Lecture:** NOR gate based SR latch using CMOS logic.
- We will transition from combinational logic to sequential memory elements.

Lecture 3.7: NOR gate based SR latch using CMOS logic

Summary

- The NOR-based SR latch provides 1-bit memory using positive feedback.
- Active High inputs: S=1 sets Q=1, R=1 resets Q=0.
- Hold state occurs at S=0, R=0.
- Forbidden state occurs at S=1, R=1, violating complementary outputs.
- CMOS implementation requires 8 transistors.

Questions & Next Steps

- Any questions regarding the NOR-based SR latch?
- Next Lecture: NAND gate based SR latch using CMOS logic.
- We will examine the active-low dual of the circuit discussed today.

Lecture 3.8: NAND gate based SR latch using CMOS logic

Summary

- The NAND-based SR latch uses active-low Set and Reset inputs.
- Applying a '0' forces the respective output to '1'.
- The (1,1) input condition holds the previous state.
- The (0,0) input condition is forbidden.
- CMOS design uses 8 transistors with cross-coupled parallel-PMOS/series-NMOS networks.

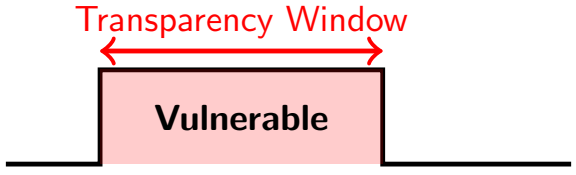
Questions & Next Steps

- Any questions regarding the NAND-based SR latch?
- Next Lecture: Clocked NOR based SR latch using CMOS logic.
- We will introduce a clock signal to synchronize the latching operation.

Lecture 3.9: Clocked NOR based SR latch using CMOS logic

Limitations of Level-Sensitive Latches

- The latch is transparent for the ENTIRE duration of the clock pulse.
- If inputs S and R change multiple times while CLK is HIGH, the output Q will follow.
- This can lead to timing violations in cascaded logic (race around condition).
- **Solution:** Edge-triggered Flip-Flops.



Summary

- A clock signal synchronizes the state transitions of the SR latch.
- The latch is opaque (Holds) when CLK=0, and transparent when CLK=1.
- Input AND gates prevent S and R from reaching the inner latch when CLK=0.
- Optimized CMOS design merges clock transistors directly into the latch to save silicon area.

Questions & Next Steps

- Any questions on synchronous memory or clocking logic?
- **Next Lecture:** Clocked NOR based JK latch using CMOS logic.
- We will resolve the forbidden state $(1, 1)$ issue of the SR latch.

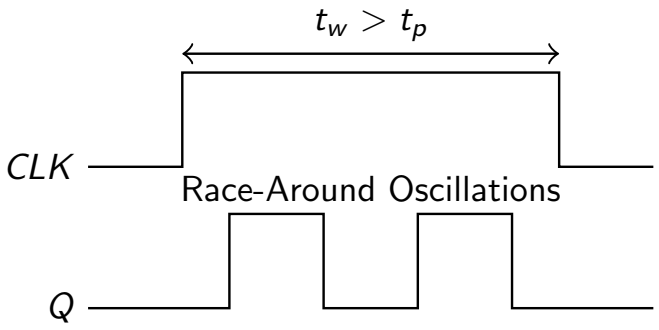
Lecture 3.10: Clocked NOR based JK latch using CMOS logic

Toggle State Analysis

- Assume $Q = 0, \overline{Q} = 1$. Apply $J = 1, K = 1, CLK = 1$.
- Set side logic evaluates: $(1 \cdot 1 \cdot 1) + 0 = 1$. $\overline{Q}$ pulls to 0.
- Reset side logic evaluates: $(1 \cdot 1 \cdot 0) + 0 = 0$. Q pulls to 1.
- State toggles perfectly from 0 to 1 without race conditions.

Limitations and Race-Around Condition

- If CLK remains high longer than the propagation delay of the latch (t_p), the outputs will toggle multiple times.
- This is known as the 'Race-Around Condition'.
- For stable operation, pulse width t_w must be $< t_p$.
- Practical solution: Master-Slave configuration or Edge-Triggered flip-flops.



Lecture Summary

- The JK latch solves the invalid state of the SR latch via output feedback.
- When $J = 1, K = 1$, the latch safely toggles its state.
- CMOS implementation uses two cross-coupled AOI complex gates (16 transistors).
- Level-sensitive JK latches suffer from race-around if the clock pulse is too wide.

Lecture 3.11: Clocked NOR based D latch using CMOS logic

Lecture Summary

- The D latch ensures S and R are never asserted simultaneously by using a single data input and an inverter.
- The CMOS realization relies on two cross-coupled AOI gates and one inverter, totaling 14 transistors.
- The latch is level-sensitive (transparent on high clock).
- Strict adherence to setup and hold times is required to avoid metastability.

Lecture 4.1: VLSI design flow (Y chart) and Design Properties

Lecture Agenda 1. Complexity of Modern VLSI Design 2. The Gajski-Kuhn Y-Chart 3. Three Domains of Representation 4. Levels of Abstraction 5. Key Design Properties: Hierarchy, Regularity, Modularity, Locality

The Complexity Challenge

- Modern VLSI chips contain billions of transistors (e.g., Apple M-series, Nvidia GPUs).
- Designing at the transistor level manually is impossible.
- We require structured methodologies, automation (EDA tools), and abstraction.
- A formal design flow ensures functionality, performance, and manufacturability.

The Gajski-Kuhn Y-Chart

- Introduced by Daniel Gajski and Robert Kuhn in 1983.
- Captures the essence of the VLSI design process.
- Consists of three axes representing different design domains.
- Concentric circles represent levels of abstraction (outer = high-level, inner = low-level).

The Three Domains of the Y-Chart

- Behavioral Domain: Specifies WHAT the system does. Focuses on function, algorithms, and timing.
- Structural Domain: Specifies HOW the system is connected. Focuses on components, gates, and netlists.
- Physical/Geometrical Domain: Specifies WHERE components are placed. Focuses on layouts, polygons, and masks.

Levels of Abstraction (Concentric Circles)

- 1. System Level: Processors, memories, buses.
- 2. Algorithmic Level: State machines, data paths.
- 3. Register Transfer Level (RTL): ALUs, registers, MUXes.
- 4. Logic Level: Logic gates (AND, OR, Flip-Flops).
- 5. Circuit Level: Transistors, resistors, capacitors.

Walking the Y-Chart: The Design Flow

- Top-down design starts at Behavioral System level.
- Synthesis: Moving from Behavioral to Structural (e.g., RTL code to gate netlist).
- Placement & Routing: Moving from Structural to Physical (netlist to layout).
- Extraction & Verification: Moving outwards from Physical back to Structural/Behavioral for checking.

Managing Complexity: The Four Pillars

- Even with automation, a flat design of billions of gates is computationally and humanly unmanageable.
- We employ four fundamental design properties:
- Hierarchy
- Regularity
- Modularity
- Locality

Design Property: Hierarchy

- Definition: Dividing a complex system into smaller, manageable sub-systems.
- A 'divide and conquer' approach.
- System -¿ Microprocessor -¿ ALU -¿ Adder -¿ Logic Gate -¿ Transistor.
- Reduces the sheer number of elements a designer or tool must handle at once.

Design Property: Regularity

- Definition: Maximizing the reuse of standard, identical blocks.
- Reduces the number of unique components that need to be designed and verified.
- Examples: Memory arrays (SRAM/DRAM), FPGA logic blocks, standard cell libraries.
- Regular layouts reduce DRC (Design Rule Check) errors and improve yield.

Design Property: Modularity

- Definition: Designing blocks with well-defined interfaces and predictable behavior.
- A modular block can be designed independently of the rest of the system.
- Requires strict adherence to interface protocols (e.g., AXI bus) and timing contracts.
- Allows parallel development by different teams.

Design Property: Locality

- Definition: Keeping connections and communications local whenever possible.
- Physical Locality: Placing communicating blocks close together on the silicon.
- Temporal Locality: Processing data locally to avoid long delays.
- Why? Long wires in VLSI have high resistance and capacitance (RC delay), causing slow signals and high power dissipation.

Synergy of the Design Properties

- Hierarchy breaks down the problem.
- Regularity simplifies the components.
- Modularity ensures the components plug together seamlessly.
- Locality ensures the final assembled physical chip meets speed and power goals.

Impact on EDA Tools

- Electronic Design Automation (EDA) heavily relies on these properties.
- Synthesis tools use standard cell libraries (Regularity).
- Place & Route tools utilize clustering algorithms to optimize wire length (Locality).
- Static Timing Analysis (STA) tools verify timing across boundaries (Modularity).

Lecture Summary

- The Gajski-Kuhn Y-chart models VLSI design across Behavioral, Structural, and Physical domains.
- Abstraction levels range from System down to Circuit.
- Hierarchy, Regularity, Modularity, and Locality are vital to managing multi-billion transistor designs.
- These principles enable structured top-down design and physical feasibility.

Lecture 4.2: FPGA Architecture and Characteristics

Lecture Agenda 1. What is an FPGA? 2. High-Level FPGA Architecture 3. Configurable Logic Blocks (CLBs) 4. Programmable Interconnects 5. Programming Technologies 6. Advantages vs. ASICs

What is an FPGA?

- FPGA: Field Programmable Gate Array.
- A semiconductor device containing programmable logic components and programmable interconnects.
- 'Field Programmable' means it is configured by the customer after manufacturing, not in the foundry.
- Used to implement any logical function that an ASIC could perform, but with flexibility.

High-Level Architecture

- FPGAs consist of three main components:
- 1. Configurable Logic Blocks (CLBs): The computing islands that perform logic.
- 2. Programmable Interconnects: The routing channels (wires and switches) connecting CLBs.
- 3. I/O Blocks (IOBs): Interfaces to the external pins of the chip.

Configurable Logic Blocks (CLBs)

- CLBs do not use fixed logic gates (AND/OR).
- They typically contain:
- Look-Up Tables (LUTs): Implement combinatorial logic via SRAM truth tables.
- Flip-Flops/Registers: For sequential logic and state storage.
- Multiplexers (MUX): To route signals within the CLB.

Programmable Interconnects

- Wiring is just as important as logic.
- Switch Matrices: Located at the intersection of routing channels.
- Programmable switches determine which wires connect to which.
- Takes up a large percentage of the chip's silicon area.

Input/Output Blocks (IOBs)

- Sit on the perimeter of the die.
- Programmable for different voltage standards (e.g., 3.3V, 1.8V, LVDS).
- Can be configured as input, output, or bidirectional.
- Often contain their own registers to improve setup/hold timing to external devices.

Programming Technology: SRAM

- Static RAM (SRAM) is the most common programming technology (e.g., Xilinx, Altera).
- Configuration is stored in volatile SRAM cells.
- Must be reconfigured every time power is applied (booted from an external Flash ROM).
- Advantage: Infinitely reprogrammable, uses standard CMOS logic process.

Programming Technology: Anti-fuse & Flash

- Anti-fuse: One-time programmable (OTP). Starts as an open circuit, programming 'blows' it to create a short.
- High radiation tolerance (used in space), non-volatile, cannot be changed.
- Flash: Uses floating-gate transistors.
- Non-volatile, reprogrammable (limited cycles), retains design on power down, highly secure.

Modern FPGA Additions (SoCs)

- Modern FPGAs are heterogeneous.
- They include hard IP blocks to save space and increase speed:
- DSP (Digital Signal Processing) slices for fast math.
- Block RAMs (BRAM) for dense memory.
- Hard ARM Cortex processor cores (e.g., Zynq SoC).

Advantages of FPGAs

- Fast Time-to-Market: No fabrication wait time; compile and download in minutes.
- Reconfigurability: Hardware can be updated remotely (bug fixes, new features).
- Parallelism: Can execute hundreds of operations simultaneously, unlike CPUs.
- Lower NRE Cost: No Non-Recurring Engineering (mask) costs compared to ASICs.

Limitations of FPGAs (vs. ASICs)

- Slower Clock Speeds: Due to routing delays in programmable switches.
- Higher Power Consumption: Programmable switches have high capacitance.
- Larger Silicon Area: An FPGA is much larger than an ASIC performing the same logic.
- High Unit Cost: At very high volumes (millions of units), ASICs are much cheaper per chip.

Typical Applications

- ASIC Prototyping: Verifying logic before spending millions on masks.
- Telecom & Networking: Routers and switches where protocols change rapidly.
- Aerospace & Defense: Anti-fuse FPGAs for secure, radiation-hardened systems.
- Data Center Acceleration: Offloading search, AI inference, and video transcoding.

Design Flow for FPGAs

- 1. RTL Design (Verilog/VHDL).
- 2. Synthesis (Translating RTL to LUTs).
- 3. Place & Route (Assigning logic to specific CLBs and routing wires).
- 4. Bitstream Generation.
- 5. Programming (Downloading to the board).

Lecture Summary

- FPGAs are field-programmable logic devices based on CLBs, Interconnects, and IOBs.
- Logic is typically implemented using SRAM-based Look-Up Tables (LUTs).
- They offer rapid time-to-market and reconfigurability but consume more power and area than ASICs.
- FPGAs bridge the gap between software flexibility and hardware performance.

Lecture 4.3: CMOS Fabrication using CMOS n-Well Process

Lecture Agenda 1. The Challenge of CMOS Fabrication 2. Step 1: Substrate Preparation and n-Well Formation 3. Step 2: Active Area Definition (Isolation) 4. Step 3: Gate Oxide and Polysilicon Deposition 5. Step 4: Source/Drain Implantation (n^+ and p^+) 6. Step 5: Contacts and Metallization

The Challenge of CMOS

- CMOS requires both NMOS and PMOS transistors on the same chip.
- NMOS needs a *p*-type substrate.
- PMOS needs an *n*-type substrate.
- Solution: Start with a *p*-type wafer (for NMOS), and create local *n*-type regions called 'n-wells' for the PMOS devices.

Step 1: Substrate and n-Well Formation

- Start with a lightly doped p-type silicon wafer.
- Grow a protective Oxide layer (SiO_2).
- Apply Photoresist, use Lithography to expose the n-well area.
- Perform Ion Implantation of n-type dopants (e.g., Phosphorus).
- Drive-in diffusion to deepen the n-well.

Step 2: Active Area Definition (Isolation)

- We must electrically isolate adjacent transistors to prevent parasitic paths.
- Traditional method: LOCOS (Local Oxidation of Silicon) using a Si_3N_4 mask to grow thick Field Oxide (FOX).
- Modern method: STI (Shallow Trench Isolation).
- Trench is etched and filled with dielectric material (SiO_2).

Step 3: Gate Oxide Formation

- The most critical step in fabrication.
- A very thin, high-quality layer of silicon dioxide (SiO_2) is grown over the active areas.
- This is the dielectric that separates the gate from the channel.
- Thickness is often less than 2 nanometers in modern processes.

Step 4: Polysilicon Gate Deposition

- Polycrystalline silicon (Polysilicon) is deposited over the entire wafer using Chemical Vapor Deposition (CVD).
- Heavily doped to make it highly conductive.
- Lithography is used to pattern the polysilicon.
- Etching removes the poly everywhere except where the gates will be.

Self-Aligned Gate Process

- The polysilicon gate acts as a mask for the next step.
- When implanting the source and drain, the gate blocks the ions from entering the channel region.
- This ensures the source and drain edges align perfectly with the gate edges.
- Eliminates severe parasitic capacitance issues.

Step 5: n^+ Source/Drain Implantation

- Protect the PMOS (n-well) regions using photoresist.
- Perform heavy Ion Implantation of *n*-type dopants (Arsenic or Phosphorus).
- This forms the heavily doped n^+ Source and Drain for the NMOS transistors in the *p*-substrate.
- Also forms the n^+ body contact for the n-well.

Step 6: p^+ Source/Drain Implantation

- Remove previous photoresist.
- Protect the NMOS (p-substrate) regions using new photoresist.
- Perform heavy Ion Implantation of p-type dopants (Boron).
- This forms the p^+ Source and Drain for the PMOS transistors in the n-well.
- Also forms the p^+ body contact for the p-substrate.

Step 7: Contacts Definition

- Deposit a thick layer of insulating oxide (e.g., BPSG - Borophosphosilicate glass) over the entire wafer.
- Use lithography and etching to cut vertical holes ('contacts' or 'vias') down to the Source, Drain, and Gate regions.
- These holes allow metal to reach the silicon.

Step 8: Metallization

- Sputter or electroplate a layer of Metal (Aluminum or Copper) over the wafer.
- The metal fills the contact holes, making electrical connection with the silicon.
- Pattern the metal using lithography/etching to create the local interconnect wires.
- Modern chips repeat this step up to 10-15 times to create a multi-layer routing grid.

Passivation and Final Testing

- Once all metal layers are finished, a final protective passivation layer (Silicon Nitride) is applied.
- Openings are etched only over the large bonding pads at the edge of the chip.
- Wafer testing is performed using probe cards to identify working dies before dicing.

Summary of Lithography Cycles

- Each physical feature requires a masking step (Lithography).
- Typical basic CMOS process requires masks for:
- 1. n-Well
- 2. Active Area (STI)
- 3. Polysilicon Gate
- 4. n^+ Implant
- 5. p^+ Implant
- 6. Contacts
- 7. Metal 1
- 8. Vias and subsequent metals...

Lecture Summary

- CMOS integration requires both *n*-type and *p*-type substrates; the n-well process solves this.
- Key steps include: n-well diffusion, isolation (STI), gate oxidation, and poly deposition.
- The polysilicon gate enables a self-aligned process for S/D implantation.
- Multiple layers of metallization connect the millions of transistors into a functional system.

Lecture 4.4: Diffusion and Ion implantation

Agenda Concept of Semiconductor Doping

Solid-State Diffusion Basics Fick's Laws of Diffusion

Diffusion Profiles Ion Implantation Principle

Channeling and Annealing Comparison: Diffusion vs.

Ion Implantation

Concept of Semiconductor Doping

- Intrinsic silicon is a poor conductor at room temperature.
- Doping introduces specific impurity atoms (Group III or Group V) into the silicon lattice.
- Creates n-type (electron rich) or p-type (hole rich) regions.
- Key requirements: precise control of dopant concentration and junction depth.

Solid-State Diffusion

- Process where dopant atoms move from a region of high concentration to low concentration.
- Requires high temperatures (typically 800°C to 1200°C).
- Dopant source can be solid, liquid, or gas.
- Atoms move through the silicon lattice via vacancy or interstitial mechanisms.

Fick's First Law of Diffusion

- Relates the diffusive flux to the concentration gradient.
- J = -D * (∂C / ∂x)
- Where J is the flux of atoms, D is the diffusion coefficient, and C is the concentration.
- Negative sign indicates flow is from high to low concentration.

Fick's Second Law of Diffusion

- Describes how the concentration changes with time.
- $\partial C / \partial t = D * (\partial^2 C / \partial x^2)$
- Derived from Fick's first law and the continuity equation.
- Allows calculation of the dopant profile C(x,t) after a specific diffusion time.

Diffusion Profiles: Constant Source vs. Limited Source

- Constant Source (Predeposition):
- Surface concentration (Cs) is maintained constant.
- Profile follows a Complementary Error Function (erfc).
- Limited Source (Drive-in):
- Fixed amount of dopant is introduced, then driven deeper.
- Profile follows a Gaussian distribution.

Limitations of Diffusion

- Isotropic nature leads to lateral diffusion under the mask.
- Difficult to independently control surface concentration and junction depth.
- High temperatures cause unwanted movement of previously formed junctions.
- Maximum solid solubility limits the peak concentration.

Ion Implantation Basics

- Physical process where dopant ions are accelerated to high energies and directed at the wafer.
- Room temperature process (mostly).
- Energy ranges from 1 keV to 1 MeV.
- Dose is controlled by measuring the ion beam current.

Ion Implantation Profile

- Peak concentration is located below the silicon surface.
- Profile is approximately Gaussian around the projected range (Rp).
- Rp is determined by the ion mass and acceleration energy.
- Straggle (ΔRp) describes the statistical spread of the stopping depths.

Ion Channeling Effect

- Silicon is a single crystal with open spaces (channels) in certain crystal orientations.
- Ions traveling down these channels experience few collisions and penetrate much deeper.
- Creates an unwanted 'tail' in the doping profile.
- Mitigation: Tilt the wafer (e.g., 7 degrees) or deposit a thin scattering oxide layer.

Post-Implant Annealing

- High-energy ions severely damage the silicon crystal structure (amorphization).
- Implanted dopants are mostly not in electrically active lattice sites.
- Annealing (high temperature step) is required to repair lattice damage.
- Also activates dopants by moving them to substitutional sites.

Diffusion vs. Ion Implantation

- Diffusion:
- High temperature, isotropic profile.
- Poor independent control of depth and dose.
- No lattice damage.
- Ion Implantation:
- Low temperature process, highly directional (anisotropic).
- Excellent independent control of depth (energy) and dose (beam current).
- Causes lattice damage requiring annealing.

Key Takeaways

- Diffusion relies on thermal energy to drive dopants down a concentration gradient.
- Fick's laws govern solid-state diffusion profiles (erfc and Gaussian).
- Ion implantation provides precise, independent control over dopant dose and junction depth.
- Annealing is mandatory after implantation to repair crystal damage and activate dopants.

Next Lecture Preview Oxidation mechanisms in silicon. Dry vs. Wet oxidation. Photolithography process steps. Resolution limits in lithography.

Lecture 4.5: Oxidation and Photolithography

Agenda Role of Silicon Dioxide (SiO2) Thermal

Oxidation Process Dry vs. Wet Oxidation

Deal-Grove Oxidation Model Principles of

Photolithography Photoresists: Positive and

Negative Lithography Sequence Resolution Limits

Role of Silicon Dioxide (SiO2)

- Excellent electrical insulator (high resistivity).
- Serves as the gate dielectric in MOSFETs.
- Acts as a mask against diffusion and ion implantation.
- Used for surface passivation (protecting the silicon surface from contamination).
- Provides isolation between different devices (Field Oxide).

Thermal Oxidation Process

- Growth of SiO2 by exposing silicon to oxygen or water vapor at high temperatures.
- Typically performed in horizontal or vertical quartz tube furnaces.
- Temperature range: 800°C to 1200°C.
- Oxidation consumes the underlying silicon substrate.

Dry vs. Wet Oxidation

- Dry Oxidation (O_2 ambient):
- $Si + O_2 \rightarrow SiO_2$
- Slow growth rate.
- Produces dense, high-quality oxide (used for Gate Oxide).
- Wet Oxidation (H_2O ambient):
- $Si + 2H_2O \rightarrow SiO_2 + 2H_2$
- Fast growth rate (due to higher solubility of H_2O in SiO_2).
- Lower density oxide (used for Field Oxide/Masking).

Deal-Grove Oxidation Model

- Standard model predicting oxide thickness (x) over time (t).
- $x^2 + A*x = B(t + \tau)$
- Reaction-controlled regime (thin oxide, linear growth): $x \approx (B/A)*t$
- Diffusion-controlled regime (thick oxide, parabolic growth): $x^2 \approx B*t$

Principles of Photolithography

- Process of transferring geometric shapes on a mask to the surface of a silicon wafer.
- Uses ultraviolet (UV) light to expose a light-sensitive polymer (photoresist).
- Acts as a temporary stencil for subsequent etching or implantation steps.
- The most critical and expensive step in IC fabrication.

Photoresists: Positive vs. Negative

- Positive Photoresist:
- UV exposure breaks polymer chains.
- Exposed areas become soluble in the developer.
- Pattern on wafer matches the mask.
- Negative Photoresist:
- UV exposure cross-links polymer chains.
- Exposed areas become insoluble.
- Pattern on wafer is the inverse of the mask.

Lithography Sequence: Steps 1-3

- 1. Surface Preparation: Clean wafer, apply adhesion promoter (HMDS).
- 2. Spin Coating: Dispense liquid photoresist and spin at high speed (e.g., 3000 RPM) for uniform thickness.
- 3. Soft Bake: Low temperature bake (100°C) to drive off solvents and solidify the resist.

Lithography Sequence: Steps 4-5

- 4. Alignment and Exposure: Precisely align the mask to existing wafer patterns. Expose to intense UV light.
- 5. Development: Submerge or spray with chemical developer to dissolve the soluble photoresist areas.

Lithography Sequence: Steps 6-7

- 6. Hard Bake: High temperature bake (130°C) to harden the resist against harsh etching/implanting chemicals.
- 7. Pattern Transfer & Stripping: Perform the actual etch or implant. Finally, remove (strip) the remaining photoresist using plasma or solvents.

Resolution and Depth of Focus

- Resolution (R): The smallest feature that can be printed. R = k1 * (λ / NA)
- Depth of Focus (DOF): The range over which the image remains sharply in focus. DOF = k2 * (λ / NA^2)
- λ = light wavelength, NA = Numerical Aperture of the lens.
- Trade-off: Decreasing λ or increasing NA improves resolution but degrades DOF.

Key Takeaways

- SiO2 is crucial for isolation and gate control, grown thermally via dry or wet processes.
- Deal-Grove model defines oxidation kinetics: reaction-limited (thin) to diffusion-limited (thick).
- Photolithography uses UV light and photoresist to transfer mask patterns to the wafer.
- Resolution is limited by diffraction (wavelength and numerical aperture).

Next Lecture Preview Etching techniques: Wet vs. Dry (Plasma) etching. Isotropic vs. Anisotropic etch profiles. Deposition methods: CVD and PVD. Metallization and interconnects.

Lecture 4.6: Etching and Deposition

Agenda Purpose of Etching Isotropic vs.

Anisotropic Profiles Wet Chemical Etching Dry

(Plasma) Etching: RIE Purpose of Deposition

Chemical Vapor Deposition (CVD) Physical Vapor

Deposition (PVD) / Sputtering Metallization

Overview

The Purpose of Etching

- Process of selectively removing material from the wafer surface.
- Uses the patterned photoresist (from lithography) as a protective mask.
- Transfers the 2D mask pattern into a 3D structural feature.
- Requires high selectivity: must etch the target layer much faster than the mask or underlying layer.

Etch Profiles: Isotropic vs. Anisotropic

- Isotropic Etching:
- Etches in all directions equally (vertical rate = horizontal rate).
- Causes undercutting beneath the mask.
- Poor dimensional control for small geometries.
- Anisotropic Etching:
- Etches preferentially in one direction (usually vertically).
- Produces straight, vertical sidewalls.
- Essential for sub-micron VLSI and high-density packing.

Wet Chemical Etching

- Wafer is submerged in a liquid chemical bath.
- Purely a chemical reaction.
- High selectivity (easy to find chemicals that etch A but not B).
- Highly isotropic (severe undercut).
- Example: Hydrofluoric acid (HF) etches SiO2 but not Si.

Dry (Plasma) Etching

- Uses ionized gases (plasma) in a vacuum chamber instead of liquids.
- Involves both chemical reactions (reactive gas radicals) and physical removal (ion bombardment).
- Can achieve highly anisotropic (vertical) profiles.
- Standard for modern VLSI fabrication.

Reactive Ion Etching (RIE)

- The most common form of dry etching.
- Balances physical sputtering (ions) and chemical etching (radicals).
- Ions accelerated by electric field provide the vertical directionality.
- Polymers can form on sidewalls to prevent lateral chemical etching (passivation).

The Purpose of Deposition

- Additive process: building new layers on the wafer surface.
- Unlike thermal oxidation, it does not consume the underlying silicon substrate.
- Used for depositing:
- Polysilicon (for gate electrodes).
- Silicon Nitride (for passivation and masking).
- Metals (Aluminum, Copper for interconnects).

Chemical Vapor Deposition (CVD)

- Gas-phase precursors react at the heated wafer surface to form a solid film.
- Excellent step coverage (coats sidewalls and bottoms of deep trenches evenly).
- Commonly used for Polysilicon, Silicon Dioxide, and Silicon Nitride.
- LPCVD (Low Pressure CVD) offers high purity and uniformity.

Physical Vapor Deposition (PVD) / Sputtering

- Purely physical process without chemical reactions.
- Material is ejected from a solid 'target' and lands on the wafer.
- Sputtering: Argon plasma ions smash into the target, knocking off atoms.
- Primarily used for depositing Metals (Al, Ti, TiN).

CVD vs. PVD

- CVD (Chemical Vapor Deposition):
- Chemical reaction at the surface.
- Excellent step coverage (conformal coating).
- High temperature process.
- Used mainly for dielectrics and polysilicon.
- PVD (Physical Vapor Deposition):
- Physical transfer of material (line-of-sight).
- Poor step coverage (shadowing effects in deep trenches).
- Lower temperature process.
- Used mainly for metals.

Metallization and Interconnects

- Connecting the billions of transistors on a chip to form a functional circuit.
- Historically: Aluminum deposited by PVD and etched via RIE.
- Modern VLSI: Copper used for lower resistance.
- Copper cannot be dry-etched easily; requires the Damascene process.

Key Takeaways

- Anisotropic (vertical) etching is mandatory for deep sub-micron pattern transfer.
- Reactive Ion Etching (RIE) combines plasma physics and chemistry for precise profiles.
- CVD is used for conformal deposition of insulators and polysilicon.
- PVD (Sputtering) is the primary method for depositing metal interconnect layers.

Next Lecture Preview Introduction to Hardware Description Languages (HDL). Why use Verilog for VLSI design? Basic features and structure of Verilog HDL. Design flow using Verilog.

Lecture 5.1: Basic Features of Verilog HDL

Agenda What is an HDL? Software vs. Hardware

Languages History of Verilog Why use Verilog?

Levels of Abstraction Concurrency in Hardware Basic

Lexical Conventions

What is a Hardware Description Language (HDL)?

- A specialized computer language used to describe the structure and behavior of electronic circuits.
- Allows designers to model, simulate, and synthesize digital systems.
- Replaces traditional schematic capture (drawing logic gates by hand).
- Two industry standards: Verilog and VHDL.

Software Language vs. HDL

- Software (C, Python, Java):
- Sequential execution (line by line).
- Runs on an existing CPU.
- Focuses on algorithms and data manipulation.
- HDL (Verilog, VHDL):
- Concurrent execution (everything happens at once).
- Used to CREATE the CPU.
- Focuses on physical hardware, timing, and structural connections.

History of Verilog

- Created in 1984 by Gateway Design Automation as a proprietary modeling language.
- Syntax is heavily influenced by the C programming language.
- Cadence Design Systems acquired it and opened it to the public domain in 1990.
- Standardized as IEEE 1364 in 1995 (Verilog-95).
- Updated to Verilog-2001 (IEEE 1364-2001) adding many useful features.

Why Use Verilog?

- Productivity: Text-based entry is vastly faster than schematic drawing.
- Reusability: Modules can be instantiated multiple times easily.
- Simulation: Test designs thoroughly before spending millions on silicon fabrication.
- Synthesis: Automated tools convert behavioral code into optimized gate-level netlists.

Levels of Abstraction in Verilog

- Verilog supports designing at multiple levels of detail:
- 1. Behavioral Level: What the circuit does (algorithms, math).
- 2. Dataflow Level: How data moves between registers (Boolean equations).
- 3. Gate Level / Structural: Connecting basic logic primitives (AND, OR).
- 4. Switch Level: Modeling at the transistor level (PMOS, NMOS).

Concurrency: The Heart of Verilog

- Hardware is inherently parallel.
- In Verilog, continuous assignments (assign statements) and module instantiations operate concurrently.
- Order of these statements in the text file does not matter.
- Event-driven simulation: a block only evaluates when one of its inputs changes.

Lexical Conventions: Syntax Basics

- Case Sensitive: 'Wire' is different from 'wire'. (All keywords are lowercase).
- Whitespace: Spaces, tabs, and newlines are ignored.
- Comments:
- // Single line comment
- /* Multi-line
- comment */
- Identifiers: Names for variables/modules. Must start with a letter or underscore.

Example: A Simple Verilog Snippet

- A brief look at how Verilog code is structured.

Logic Values in Verilog

- Unlike software which has only 0 and 1, Verilog supports a 4-value logic system to accurately model hardware:
- 0: Logic Zero (Ground)
- 1: Logic One (Vdd)
- X: Unknown value (uninitialized, or short circuit)
- Z: High Impedance (floating, disconnected)

Simulation and Testbenches

- A Design Under Test (DUT) cannot run by itself.
- We write a 'Testbench' in Verilog to verify the DUT.
- Testbench generates stimulus (input signals) and monitors outputs.
- Testbenches are simulated, but NOT synthesized into hardware.

Key Takeaways

- HDLs describe hardware structure and behavior, unlike software which defines sequential instructions.
- Verilog supports multiple abstraction levels: Behavioral, Dataflow, and Structural.
- Hardware is concurrent; Verilog models parallel execution intrinsically.
- Verilog uses a 4-value logic system (0, 1, X, Z) to represent physical electrical states.

Next Lecture Preview The anatomy of a Verilog Module. Port declarations (input, output, inout). Verilog Data Types: wire vs. reg. Number representation and constants.

Lecture 5.2: Module Definition and Data types

Agenda The Module Concept Port Declarations

Module Syntax Example Verilog Data Types

Overview The 'wire' Data Type The 'reg' Data Type

Vectors (Buses) Number Representation (Constants)

The Module Concept

- The 'module' is the basic building block in Verilog.
- Represents a logical entity or physical component (like a black box).
- A module can be a single gate, a complex adder, or an entire microprocessor.
- Modules communicate with the outside world through Ports (pins).

Port Declarations

- Ports define the interface of the module.
- Three types of direction:
- input: Data flows into the module.
- output: Data flows out of the module.
- inout: Bidirectional data flow (used for buses).
- Ports must be declared in the port list along with their direction.

Module Definition Syntax (Verilog-2001)

- Verilog-2001 allows defining the direction and data type directly in the port list.

Verilog Data Types

- Data types represent the physical connections and storage elements in hardware.
- Unlike C (int, float), Verilog types describe hardware behavior.
- Two primary data type classes for nets and variables:
- 1. Net types (most common is 'wire')
- 2. Variable types (most common is 'reg')

The 'wire' Data Type

- Represents a physical electrical connection (a simple wire).
- Cannot store a value; must be continuously driven by something (a gate output or an 'assign' statement).
- Default data type for input and inout ports.
- If left unconnected, its value defaults to 'Z' (high impedance).

The 'reg' Data Type

- Stands for 'register', but does NOT necessarily synthesize into a hardware flip-flop.
- Represents a variable that holds its value until a new value is assigned.
- Must be used for variables assigned inside procedural blocks ('always' or 'initial' blocks).
- Cannot be driven by an 'assign' statement.

wire vs. reg Summary

- wire:
- Used for continuous assignments ('assign').
- Represents physical combinational connections.
- Cannot store state.
- reg:
- Used inside procedural blocks ('always', 'initial').
- Holds its value until updated.
- Can synthesize into combinational logic OR sequential flip-flops.

Vectors (Buses)

- Used to group multiple bits into a single bus.
- Defined using square brackets [MSB:LSB] (Most Significant Bit : Least Significant Bit).
- Example:
- wire [7:0] data_bus; // An 8-bit wire bus
- reg [31:0] address; // A 32-bit register bus

Accessing Vector Elements

- You can access entire buses, parts of buses (part-select), or individual bits.

Number Representation (Constants)

- Verilog has a specific format for constants to define their bit-width and base.
- Format: <size>'<base><value>
- Size: Number of bits (optional, defaults to 32).
- Base: 'b (binary), 'h (hex), 'd (decimal), 'o (octal).
- Value: The actual number.

Constant Examples

- Examples of valid Verilog number literals.

Key Takeaways

- A 'module' is the fundamental container in Verilog, interfacing via input, output, or inout ports.
- Use 'wire' for structural connections and continuous assignments ('assign').
- Use 'reg' for variables assigned inside procedural blocks ('always').
- Constants should define size and base to ensure correct hardware mapping (e.g., 8'hA5).

Next Lecture Preview Dataflow Modeling in Verilog. Using continuous 'assign' statements. Bitwise vs. Logical operators. Implementing basic logic gates (AND, OR, NOT, etc.).

Lecture 5.3: Gate implementation in Verilog - Dataflow modelling

Agenda What is Dataflow Modelling? Continuous Assignment (assign) Verilog Operators (Bitwise vs Logical) Implementing Basic Gates (AND, OR, NOT) Implementing Universal Gates (NAND, NOR) Implementing XOR and XNOR Gates Delay in Continuous Assignment

What is Dataflow Modelling?

- Describes circuit behavior in terms of flow of data and boolean equations.
- Higher abstraction level than structural (gate-level) modelling.
- Relies heavily on the ‘assign‘ keyword.
- Models purely combinational logic circuits.

The ‘assign‘ Statement

- Continuous assignment is the fundamental construct in dataflow modelling.
- Syntax: ‘assign [delay] target = expression;‘
- Target must be of type ‘wire‘ (or other net types), never a ‘reg‘.
- Assignment is continuously evaluated; whenever inputs change, target is immediately updated.

Bitwise vs Logical Operators

- Bitwise Operators operate on corresponding bits of operands (e.g., ‘&‘, ‘—‘, ‘~‘, ‘^‘).
- Logical Operators evaluate to 1-bit true (1) or false (0) (e.g., ‘&&‘, ‘——‘, ‘¡).
- For gate implementation, we strictly use Bitwise operators to generate hardware logic.
- Example: ‘4’b1010 & 4’b1100‘ yields ‘4’b1000‘.

AND Gate Implementation

- Implementation of a 2-input AND gate using the bitwise AND ‘&‘ operator.

AND Gate Code Snippet

```
module and_gate(input A, input B, output Y);
    // Dataflow modelling using continuous assignment
    assign Y = A & B;
endmodule
```

OR and NOT Gates

- OR gate uses the bitwise OR '—' operator.
- NOT gate uses the bitwise negation '~' operator.

OR/NOT Code Snippets

```
module or_gate(input A, input B, output Y);
    assign Y = A | B;
endmodule

module not_gate(input A, output Y);
    assign Y = ~A;
endmodule
```

Universal Gates: NAND and NOR

- NAND is modeled by negating the bitwise AND: ‘~(A & B)‘
- NOR is modeled by negating the bitwise OR: ‘~(A — B)‘

NAND/NOR Code Snippets

```
module nand_gate(input A, input B, output Y);
    assign Y = ~(A & B);
endmodule

module nor_gate(input A, input B, output Y);
    assign Y = ~(A | B);
endmodule
```

XOR and XNOR Gates

- XOR gate uses the bitwise XOR '^' operator.
- XNOR gate uses negated XOR '~^' or '^~'.

XOR/XNOR Code Snippets

```
module xor_gate(input A, input B, output Y);
    assign Y = A ^ B;
endmodule

module xnor_gate(input A, input B, output Y);
    assign Y = ~(A ^ B); // Also A ~^ B
endmodule
```

Delays in Dataflow Assignments

- Real hardware gates have propagation delay.
- Dataflow supports delay specification: ‘assign #5 Y = A & B;‘
- The ‘#5‘ means Y updates 5 time units after A or B changes.
- Useful for testbenches and simulation modeling, typically ignored by synthesis tools.

Summary of Dataflow Gates

- Dataflow modelling uses continuous ‘assign‘ statements.
- Targets are of ‘wire‘ type.
- It utilizes bitwise operators: ‘&‘, ‘—‘, ‘~‘, ‘^‘.
- Ideal for modelling purely combinational logic directly from Boolean equations.

Lecture 5.4: Gate implementation in Verilog - Structural/Behavioural

Agenda Introduction to Structural Modelling

Verilog Gate Primitives Structural Code Examples

(AND, OR, NAND) Introduction to Behavioural

Modelling The ‘always‘ Block and Sensitivity List

Behavioural Code Examples Comparison of the 3

Abstraction Levels

What is Structural Modelling?

- Describes a circuit as an interconnection of components.
- Directly resembles a schematic diagram.
- Uses built-in gate primitives or user-defined modules.
- Instantiates components and connects them via ‘wire‘.

Verilog Gate Primitives

- Verilog includes predefined gate-level primitives.
- Multiple-input gates: ‘and‘, ‘nand‘, ‘or‘, ‘nor‘, ‘xor‘, ‘xnor‘.
- Single-input gates: ‘not‘, ‘buf‘.
- Syntax: ‘gatetype [instance_name] (output, input1, input2, ...);‘

Structural: AND / OR Gates

```
module structural_gates(input a, b, output y_and, y_or);
    // Syntax: primitive_name inst_name (out, in1, in2);
    and G1 (y_and, a, b);
    or  G2 (y_or, a, b);
endmodule
```

Structural: NOT, NAND, NOR

```
module struct_universal(input a, b, output y_not, y_nand,
    y_nor);
    not G3 (y_not, a);
    nand G4 (y_nand, a, b);
    nor  G5 (y_nor, a, b);
endmodule
```

What is Behavioural Modelling?

- Highest level of abstraction in Verilog.
- Describes the *behavior* of the circuit using algorithmic constructs.
- Utilizes procedural blocks like ‘always‘ and ‘initial‘.
- Outputs assigned inside these blocks MUST be declared as ‘reg‘.

The ‘always‘ Block

- Executes continuously in a loop during simulation.
- Sensitivity list ‘@(...)‘ dictates *when* the block triggers.
- For combinational logic, use ‘@(*)‘ or list all inputs: ‘@(A or B)‘.
- Inside the block, we use blocking ‘=‘ assignments for combinational logic.

Behavioural: AND Gate

```
module behav_and (input a, input b, output reg y);
    always @(a or b) begin
        if (a == 1'b1 && b == 1'b1)
            y = 1'b1;
        else
            y = 1'b0;
    end
endmodule
```

Behavioural: AND Gate (Simplified)

```
module behav_and_simple (input a, b, output reg y);
    always @(*) begin
        y = a & b;
    end
endmodule
```

Behavioural: Universal Gates

```
module behav_universal (input a, b, output reg y_nand,
    y_nor);
    always @(*) begin
        y_nand = ~(a & b);
        y_nor  = ~(a | b);
    end
endmodule
```

Behavioural: XOR / XNOR

- Using ‘case‘ statement for XOR logic:

```
module behav_xor (input a, b, output reg y);
    always @(*) begin
        case ({a, b})
            2'b00: y = 0;
            2'b01: y = 1;
            2'b10: y = 1;
            2'b11: y = 0;
        endcase
    end
endmodule
```

Comparing the Abstraction Levels

- **Structural:** Low-level, explicit gate connections. Hard to write for complex logic.
- **Dataflow:** Medium-level, Boolean equations (‘assign‘). Good for combinational arithmetic.
- **Behavioural:** High-level, algorithmic (‘always‘, ‘if‘, ‘case‘). Mandatory for sequential logic, highly readable.

When to use which?

- Use **Structural** when instantiating sub-modules or working directly with standard cell libraries.
- Use **Dataflow** for simple combinational logic, multiplexers, and continuous buses.
- Use **Behavioural** for state machines, complex decoders, ALUs, and all sequential logic (Flip-Flops, Registers).

Summary

- Structural modelling relies on primitives like ‘and‘, ‘or‘, ‘not‘.
- Behavioural modelling uses ‘always‘ blocks and ‘reg‘ data types.
- Combinational behavioural logic requires a complete sensitivity list (‘@(*)‘).
- We can implement any gate using all three modeling styles.

Lecture 5.5: Verilog code for Half/Full Adder and Subtractor

Agenda Recap of Half Adder Logic Verilog Code for Half Adder Recap of Full Adder Logic Verilog Code for Full Adder Recap of Half and Full Subtractors Verilog Code for Subtractors Simulation / Testbench Concepts

Half Adder Recap

- Adds two 1-bit inputs: A and B.
- Outputs: Sum (S) and Carry (C).
- Equations:
- $S = A \oplus B$ (XOR)
- $C = A \cdot B$ (AND)

Half Adder: Dataflow Verilog

```
module half_adder(input a, input b, output sum, output
    carry);
    // Dataflow modelling
    assign sum = a ^ b;
    assign carry = a & b;
endmodule
```

Full Adder Recap

- Adds three 1-bit inputs: A, B, and Carry-In (Cin).
- Outputs: Sum (S) and Carry-Out (Cout).
- Equations:
- $S = A \oplus B \oplus Cin$
- $Cout = (A \cdot B) + (Cin \cdot (A \oplus B))$

Full Adder: Dataflow Verilog

```
module full_adder(input a, b, cin, output sum, cout);
    // Dataflow modelling
    assign sum = a ^ b ^ cin;
    assign cout = (a & b) | (cin & (a ^ b));
endmodule
```

Full Adder using Concatenation

- An even simpler dataflow approach using Verilog's addition operator '+' and concatenation '':

```
module full_adder_concat(input a, b, cin, output sum, cout);
    assign {cout, sum} = a + b + cin;
endmodule
```

Half Subtractor Recap

- Subtracts 1-bit B from 1-bit A (A - B).
- Outputs: Difference (Diff) and Borrow (Bout).
- Equations:
- Diff $=$ A $\oplus$ B
- Bout $=$ A' $\cdot$ B

Half Subtractor: Verilog Code

```
module half_subtractor(input a, b, output diff, bout);
    assign diff = a ^ b;
    assign bout = (~a) & b;
endmodule
```

Full Subtractor Recap

- Subtracts B and Borrow-In (Bin) from A (A - B - Bin).
- Outputs: Difference (Diff) and Borrow-Out (Bout).
- Equations:
- Diff = A $\oplus$ B $\oplus$ Bin
- Bout = A'B + A'Bin + BBin = A'B + Bin(A $\oplus$ B)'

Full Subtractor: Verilog Code

```
module full_subtractor(input a, b, bin, output diff, bout);
    assign diff = a ^ b ^ bin;
    assign bout = (~a & b) | (bin & ~(a ^ b));
endmodule
```

Full Subtractor using Arithmetic Operator

- Like the adder, we can use Verilog's arithmetic '-' operator:

```
module full_sub_arithmetic(input a, b, bin, output diff,
    bout);
    // A 2-bit result is generated
    wire [1:0] temp = a - b - bin;
    assign diff = temp[0];
    assign bout = temp[1];
endmodule
```

Structural Approach (Brief Look)

- While Dataflow is easier, Structural logic is also valid for Adders.
- Example: Half Adder structurally.
- Requires instantiating ‘xor‘ and ‘and‘ primitives.

Importance of Verification

- Arithmetic circuits must be verified against their truth tables.
- A Testbench applies stimulus (all combinations of A, B, Cin) to the module.
- Simulators output waveforms to check if Sum and Carry match theoretical values.

Summary

- Half and Full Adders generate Sums and Carries using XOR, AND, and OR logic.
- Half and Full Subtractors generate Differences and Borrows.
- Verilog Dataflow (‘assign‘) handles these bitwise Boolean equations efficiently.
- Concatenation ‘cout, sum‘ and arithmetic operators ‘+‘ / ‘-‘ offer advanced shortcuts.

Lecture 5.6: Verilog code for 4-bit Full Adder and advanced arithmetic

Agenda Hierarchical Modelling Concept Full Adder using Two Half Adders The Ripple Carry Adder (RCA) Architecture 4-Bit RCA Structural Implementation 4-Bit Adder Behavioural Implementation Comparing Ripple Carry vs Behavioural Adder

Hierarchical Design in Verilog

- Complex systems are built by combining simpler sub-modules.
- Sub-modules are created once and *instantiated* multiple times.
- Connections between sub-modules are made using ‘wire‘ declarations.
- Promotes reusability and clean RTL code.

Full Adder using Half Adders

- A Full Adder can be constructed from two Half Adders and an OR gate.
- HA1 calculates partial sum/carry of A and B.
- HA2 adds Cin to the partial sum.
- The overall Cout is the OR of the carries from HA1 and HA2.

Verilog: FA using HA (Code)

```
module FA_hierarchical(input a, b, cin, output sum, cout);
    wire s1, c1, c2; // Internal connection wires

    // Instantiate two Half Adders (assume half_adder is
        defined)
    half_adder HA1 (a, b, s1, c1);
    half_adder HA2 (s1, cin, sum, c2);

    // OR gate for final carry
    assign cout = c1 | c2;
endmodule
```

Module Instantiation Syntax

- **Positional Mapping:** Order of ports matches the sub-module declaration.
- ‘half_adder HA1 (a, b, s1, c1);‘
- **Named Mapping:** Connects ports explicitly by name (Best Practice).
- ‘half_adder HA1 (.a(a), .b(b), .sum(s1), .carry(c1));‘

4-Bit Ripple Carry Adder (RCA)

- Adds two 4-bit numbers (A[3:0], B[3:0]) and a carry-in.
- Requires 4 Full Adders connected in series.
- The Cout of bit *i* becomes the Cin of bit *i+1*.
- Called 'Ripple' because the carry signal ripples from LSB to MSB.

4-Bit RCA: Structural Verilog

```
module rca_4bit(input [3:0] A, B, input Cin, output [3:0]
    Sum, output Cout);
    wire c1, c2, c3; // Intermediate carries

    full_adder FA0 (A[0], B[0], Cin, Sum[0], c1);
    full_adder FA1 (A[1], B[1], c1,  Sum[1], c2);
    full_adder FA2 (A[2], B[2], c2,  Sum[2], c3);
    full_adder FA3 (A[3], B[3], c3,  Sum[3], Cout);
endmodule
```

Understanding Vectors in Verilog

- Syntax: ‘type [MSB:LSB] identifier;‘
- Example: ‘input [3:0] A;‘ (A is a 4-bit bus: A[3], A[2], A[1], A[0])
- Little-endian ‘[3:0]‘ is standard, though ‘[0:3]‘ is valid.
- Vectors allow us to process parallel data cleanly.

4-Bit Adder: Behavioural/Dataflow

- Verilog allows multi-bit addition using the '+' operator directly.

```
module adder_4bit_behav(input [3:0] A, B, input Cin,
                        output [3:0] Sum, output Cout);
    // Concatenate Cout and Sum to catch the 5-bit result
    assign {Cout, Sum} = A + B + Cin;
endmodule
```

Why Behavioural Addition is Preferred

- **Readability:** 1 line of code vs multiple instantiations.
- **Flexibility:** Easily parameterized to N-bits (8-bit, 32-bit).
- **Synthesis Optimization:** The tool can choose faster architectures like Carry-Lookahead (CLA) instead of RCA.
- **Fewer Bugs:** No manual wiring errors for intermediate carries.

Multi-Bit Subtractors

- Similarly, a 4-bit subtractor can be structurally cascaded or written behaviourally.

```
module sub_4bit_behav(input [3:0] A, B, input Bin,
                      output [3:0] Diff, output Bout);
    assign {Bout, Diff} = A - B - Bin;
endmodule
```

Addition and Subtraction (ALU Concept)

```
module add_sub (input [3:0] A, B, input sub_flag,
                output reg [3:0] Result);
    always @(*) begin
        if (sub_flag)
            Result = A - B;
        else
            Result = A + B;
    end
endmodule
```

Pitfalls in Vector Arithmetic

- Width mismatch: ‘assign Sum_4bit = A_8bit + B_8bit;‘ (Truncation occurs!).
- Missing Carry Out: Not using ‘Cout, Sum‘ throws away the overflow bit.
- Signed vs Unsigned: By default, vectors are unsigned. Subtraction handles two’s complement, but interpretation is up to the designer.

Summary

- Hierarchical design allows us to build a Full Adder from Half Adders.
- A 4-bit Ripple Carry Adder can be built by cascading Full Adders.
- Vector instantiation simplifies multi-bit parallel signals.
- Behavioural arithmetic using ‘+‘ and ‘-‘ is the industry standard for multi-bit math.

Lecture 5.7: Verilog Programs for Combinational circuits (Mux/Demux/Decoder/Encoder)

Agenda Multiplexers (2:1 and 4:1 MUX)

Demultiplexers (1:4 DEMUX) Decoders (2-to-4

Decoder) Encoders (4-to-2 and Priority Encoders)

Parity Generators and Checkers Using ‘case‘ and

conditional operators (‘?:‘)

Multiplexer (MUX) Recap

- A MUX selects one of several input signals and forwards it into a single line.
- A 2:1 MUX has 2 data inputs (D_0, D_1), 1 select line (S), and 1 output (Y).
- Equation: $Y = (\sim S \cdot D_0) + (S \cdot D_1)$
- A 4:1 MUX has 4 data inputs and 2 select lines.

2:1 MUX using Conditional Operator

- The ternary operator ‘condition ? true_val : false_val‘ is perfect for MUX logic.

```
module mux_2to1(input d0, d1, s, output y);
    // Dataflow modelling
    assign y = s ? d1 : d0;
endmodule
```

4:1 MUX using ‘case‘ Statement

- For larger MUXes, behavioural modelling with ‘case‘ is more readable.

```
module mux_4to1(input [3:0] d, input [1:0] s, output reg y);
    always @(*) begin
        case (s)
            2'b00: y = d[0];
            2'b01: y = d[1];
            2'b10: y = d[2];
            2'b11: y = d[3];
            default: y = 1'b0;
        endcase
    end
endmodule
```

Demultiplexer (DEMUX) Recap

- Performs the reverse operation of a MUX.
- Takes a single input and routes it to one of several outputs.
- A 1:4 DEMUX has 1 data input, 2 select lines, and 4 outputs.
- Unselected outputs typically default to 0.

1:4 DEMUX Verilog Code

```
module demux_1to4(input din, input [1:0] s, output reg
    [3:0] y);
    always @(*) begin
        y = 4'b0000; // Default assignment to avoid latches
        case(s)
            2'b00: y[0] = din;
            2'b01: y[1] = din;
            2'b10: y[2] = din;
            2'b11: y[3] = din;
        endcase
    end
endmodule
```

Decoder Recap

- Translates an N-bit binary input into 2^N unique output lines.
- Only one output line is active (HIGH) at a time.
- Essentially a DEMUX where the data input is permanently tied to Logic 1.
- Often used in memory address decoding.

2-to-4 Decoder Verilog Code

```
module decoder_2to4(input [1:0] in, output reg [3:0] out);
    always @(*) begin
        case (in)
            2'b00: out = 4'b0001;
            2'b01: out = 4'b0010;
            2'b10: out = 4'b0100;
            2'b11: out = 4'b1000;
        endcase
    end
endmodule
```

Encoder Recap

- Performs the reverse operation of a Decoder.
- Translates 2^N input lines into an N-bit binary code.
- Standard Encoders assume only one input is HIGH at a time.
- Priority Encoders handle cases where multiple inputs are HIGH by prioritizing higher-order bits.

4-to-2 Priority Encoder Code

- Using ‘if-else if‘ for prioritization:

```
module priority_encoder_4to2(input [3:0] d, output reg
    [1:0] y, output reg valid);
    always @(*) begin
        valid = 1'b1;
        if (d[3])       y = 2'b11;
        else if (d[2]) y = 2'b10;
        else if (d[1]) y = 2'b01;
        else if (d[0]) y = 2'b00;
        else begin      y = 2'b00; valid = 1'b0; end
    end
endmodule
```

Parity Generator Recap

- Used for error detection in data transmission.
- Even Parity: The total number of 1s (data + parity bit) is even.
- Odd Parity: The total number of 1s is odd.
- The XOR gate determines parity. XORing all bits gives Even Parity.

Parity Generator: Verilog Reduction

- We can use the unary reduction XOR operator '^' in Verilog.

```
module parity_gen_4bit(input [3:0] data, output even_p,
    output odd_p);
    // Reduction XOR XORs all bits: data[3] ^ data[2] ^
        data[1] ^ data[0]
    assign even_p = ^data;
    assign odd_p  = ~even_p;
endmodule
```

Parity Checker Code

- The receiver checks for errors by evaluating data AND the received parity bit.

```
module parity_check(input [3:0] data, input p_rx, output
    error);
    wire parity_recalc;
    assign parity_recalc = ^data;
    // Error if recalculated parity does not match received
        parity
    assign error = (parity_recalc != p_rx);
endmodule
```

Summary

- MUXes route multiple inputs to one output; modelled beautifully with ‘case‘ or ‘?:‘.
- DEMUXes route one input to multiple outputs.
- Decoders translate binary to one-hot codes; Encoders do the reverse.
- Priority Encoders are implemented using ‘if-else if‘ constructs.
- Parity circuits leverage the powerful Verilog unary reduction XOR ‘^‘ operator.

Lecture 5.8: Verilog Programs for Combinational circuits (Parity Generator/Checker)

Agenda Concept of Parity Generation and Checking

Even vs. Odd Parity Logic Dataflow Modeling of

Parity Generator Structural Modeling of Parity

Generator Verilog Design of Parity Checker

Concept of Parity in Digital Communication

- Parity bit: An extra bit appended to a data word.
- Even Parity: Total number of 1s (data + parity) is even.
- Odd Parity: Total number of 1s is odd.
- Error Detection: Helps identify single-bit flips during transmission.

Even Parity Generator Logic

- For 3-bit data A, B, C, the even parity bit P is generated using XOR operations.
- P = A ^ B ^ C
- If data is 101 (two 1s), P = 1^0^1 = 0.
- If data is 111 (three 1s), P = 1^1^1 = 1.

Dataflow Modeling: 3-bit Even Parity Generator

```
module parity_gen_dataflow(input [2:0] data_in, output
    p_even);
  assign p_even = data_in[2] ^ data_in[1] ^ data_in[0];
endmodule
```

Dataflow Modeling: Reduction Operator

```
module parity_gen_reduction(input [7:0] data_in, output
    p_even);
  assign p_even = ^data_in;
endmodule
```

Structural Modeling: 3-bit Parity Generator

```
module parity_gen_struct(input a, b, c, output p_even);
  wire w1;
  xor x1(w1, a, b);
  xor x2(p_even, w1, c);
endmodule
```

Odd Parity Generator Logic

- Odd parity requires the total number of 1s to be odd.
- P_odd = ~(A ^ B ^ C)
- For data 101, P_odd = ~(1^0^1) = ~(0) = 1.
- Uses XNOR gates instead of XOR.

Dataflow Modeling: Odd Parity Generator

```
module parity_gen_odd(input [7:0] data_in, output p_odd);
  assign p_odd = ~^data_in;
endmodule
```

Parity Checker Logic

- Receiver receives N bits of data + 1 parity bit.
- Checker re-evaluates parity over all N+1 bits.
- For even parity protocol: Error flag E = A ^ B ^ C ^ P.
- If E=0, no single-bit error. If E=1, an error occurred.

Dataflow Modeling: Even Parity Checker

```
module parity_checker(input [3:0] rx_data, input rx_p,
    output error);
  assign error = (^rx_data) ^ rx_p;
endmodule
```

Behavioral Modeling: Parity Generator

```
module parity_gen_behav(input [3:0] in, output reg p);
  integer i;
  always @(in) begin
    p = 0;
    for(i=0; i<4; i=i+1)
      p = p ^ in[i];
  end
endmodule
```

Synthesis Comparison: Structural vs Behavioral

- Structural Modeling: Direct mapping to primitives, harder to scale for large buses.
- Behavioral/Dataflow: Highly scalable, synthesis tool optimizes the XOR tree for minimum delay.

Summary

- Parity generators append a bit to ensure even or odd parity.
- XOR and XNOR operations form the basis of parity circuits.
- Reduction operators (^, ~^) are the most efficient way to code parity in Verilog.
- Parity checkers use identical logic to flag single-bit transmission errors.

Lecture 5.9: Test bench structure and initial/always blocks

Agenda Purpose of a Testbench General Testbench Structure Instantiating the DUT The 'initial' Block for Stimulus The 'always' Block for Clock Generation System Tasks for Monitoring

What is a Testbench?

- A non-synthesizable Verilog module used for simulation.
- Provides stimulus (inputs) to the Design Under Test (DUT).
- Monitors the response (outputs) of the DUT.
- Does not have input or output ports itself.

General Testbench Structure

```
module tb_top;
  // 1. Declare variables (reg for inputs, wire for outputs)
  // 2. Instantiate DUT
  // 3. Stimulus generation (initial block)
  // 4. Clock generation (always block)
  // 5. Response monitoring ($monitor)
endmodule
```

Variable Declarations in Testbench

- Inputs to DUT -> Declared as 'reg'.
- Why? They are driven from procedural blocks (initial/always).
- Outputs from DUT -> Declared as 'wire'.
- Why? They are continuously driven by the instantiated DUT module.

Instantiating the Design Under Test (DUT)

```
// Assuming DUT: module AND2 (input a, b, output y);

module tb_and2;
  reg tb_a, tb_b;
  wire tb_y;

  // Named Instantiation
  AND2 uut (.a(tb_a), .b(tb_b), .y(tb_y));
endmodule
```

The 'initial' Block

- Executes only once at the start of simulation (t=0).
- Used for applying deterministic stimulus sequences.
- Statements execute sequentially inside a begin-end block.
- Delay control (#) is used to advance simulation time.

Stimulus Generation using 'initial'

```
initial begin
  tb_a = 0; tb_b = 0;    // At time 0
  #10 tb_a = 0; tb_b = 1; // At time 10
  #10 tb_a = 1; tb_b = 0; // At time 20
  #10 tb_a = 1; tb_b = 1; // At time 30
  #10 $finish;            // End simulation at time 40
end
```

The 'always' Block

- Executes continuously in a loop throughout the simulation.
- Requires a time delay or event control; otherwise, it creates an infinite zero-delay loop.
- Primarily used in testbenches for clock generation.

Clock Generation using 'always'

```
reg clk;

initial begin
  clk = 0; // Initialize clock
end

always begin
  #5 clk = ~clk; // Toggle every 5 units (Period = 10)
end
```

The 'forever' Loop inside initial

- Alternative to 'always' for recurring events.
- Must be placed inside an 'initial' block.
- Also requires timing control to avoid simulator hangs.

System Tasks for Display and Monitor

- $display: Prints the values immediately when executed, like 'printf' in C.
- $monitor: Continuously monitors variables and prints them whenever any variable in its list changes.
- $time: Returns the current simulation time.

Using $monitor in a Testbench

```
initial begin
  $monitor("Time=%0t␣|␣a=%b␣b=%b␣|␣y=%b", $time, tb_a,
      tb_b, tb_y);
end
```

Summary

- Testbenches provide stimulus and monitor DUT response.
- DUT inputs are driven by 'reg' variables; outputs are read via 'wire'.
- The 'initial' block runs once and is used for specific test vectors.
- The 'always' block loops continuously and is ideal for clock generation.
- $monitor is used to observe signal changes textually.

Lecture 5.10: Simple testbench example for Basic gates

Agenda Design Module: Basic Gates Testbench

Architecture Stimulus Generation Setup Simulation

Execution Flow Waveform and Output Analysis

Step 1: The Design Module

```
module basic_gates (input a, input b,
                    output y_and, output y_or, output
                        y_not);
  assign y_and = a & b;
  assign y_or  = a | b;
  assign y_not = ~a;
endmodule
```

Step 2: Testbench Setup and Variable Declaration

```
module tb_basic_gates;
  // Drive signals as reg
  reg tb_a, tb_b;

  // Observe signals as wire
  wire tb_and, tb_or, tb_not;
```

Step 3: Instantiating the DUT

```
// Instantiate DUT using named mapping
basic_gates dut (
   .a(tb_a),
   .b(tb_b),
   .y_and(tb_and),
   .y_or(tb_or),
   .y_not(tb_not)
);
```

Step 4: Monitoring System Task

```
initial begin
  $monitor("T=%0t␣|␣a=%b,␣b=%b␣|␣and=%b,␣or=%b,␣not_a=%b",
            $time, tb_a, tb_b, tb_and, tb_or, tb_not);
end
```

Step 5: Applying Stimulus Vectors

```
  initial begin
    // Initialize
    tb_a = 0; tb_b = 0; #10;
    tb_a = 0; tb_b = 1; #10;
    tb_a = 1; tb_b = 0; #10;
    tb_a = 1; tb_b = 1; #10;
    $finish;
  end
endmodule
```

Expected Simulation Console Output

- T=0 — a=0, b=0 — and=0, or=0, not_a=1
- T=10 — a=0, b=1 — and=0, or=1, not_a=1
- T=20 — a=1, b=0 — and=0, or=1, not_a=0
- T=30 — a=1, b=1 — and=1, or=1, not_a=0

Understanding the Output

- Event-driven simulation: Output only prints when a variable changes.
- At T=10, b changes from 0 to 1.
- The simulator evaluates the continuous assignments in 'basic_gates'.
- y_or evaluates to 1. y_and remains 0.
- $monitor detects the change and prints the new state.

Adding Waveform Dumping

```
initial begin
  $dumpfile("dump.vcd");
  $dumpvars(0, tb_basic_gates);
end
```

Analyzing the Waveform (VCD)

- Waveform viewer displays signals graphically over time.
- X-axis represents simulation time (0 to 40 units).
- Y-axis represents logic levels (0 or 1).
- Provides a clear visual check of propagation and logic correctness.

Handling Unknown States (X)

- If inputs are not initialized at T=0, their value is 'X' (unknown).
- Gates evaluating 'X' will often propagate 'X' to the output.
- Always initialize stimulus 'reg' variables at the start of an initial block.

Best Practices for Simple Testbenches

- Keep testbench and design in separate files or distinct module blocks.
- Use named instantiations to prevent port mismatch bugs.
- Test exhaustively for small combinational circuits (all 2^N combinations).
- Use self-checking logic for larger designs (advanced topic).

Summary

- A complete testbench instantiates the DUT, applies inputs, and monitors outputs.
- Initial blocks execute sequentially to create a timeline of input vectors.
- System tasks like $monitor and $dumpvars provide textual and graphical visibility.
- Initializing variables prevents unknown 'X' state propagation.

Lecture 5.11: Verilog programs for Sequential circuits (Latches and Flip-Flops)

Agenda Sequential Circuit Concepts in Verilog

Blocking vs Non-Blocking Assignments SR Latch

using Behavioral Modeling D Flip-Flop (Positive

Edge Triggered) T Flip-Flop and JK Flip-Flop

Implementation Synchronous vs Asynchronous

Resets

Modeling Memory: The 'always' Block

- Sequential logic is modeled using 'always' blocks with sensitivity lists.
- Level-sensitive: ‘always @(en, d)‘ -> infers Latches.
- Edge-sensitive: ‘always @(posedge clk)‘ -> infers Flip-Flops.
- Outputs assigned inside always blocks must be 'reg' type.

Crucial Syntax: Non-Blocking Assignments (<=)

- Blocking (=): Executes sequentially. Used for combinational logic in always blocks.
- Non-Blocking (<=): Evaluates RHS for all statements, then updates LHS concurrently at the end of the time step.
- GOLDEN RULE: Always use '<=' for edge-triggered sequential logic (Flip-Flops).

SR Latch Behavioral Model

```
module sr_latch (input s, r, en, output reg q, output
    q_bar);
  assign q_bar = ~q;
  always @(s, r, en) begin
    if (en) begin
      if       (s==1 && r==0) q = 1;
      else if (s==0 && r==1) q = 0;
      else if (s==1 && r==1) q = 1'bx; // Invalid state
    end
  end
endmodule
```

D Flip-Flop (Positive Edge Triggered)

```
module d_ff (input clk, d, output reg q);
  always @(posedge clk) begin
    q <= d;
  end
endmodule
```

D Flip-Flop with Synchronous Reset

```
module d_ff_sync_rst (input clk, rst, d, output reg q);
  always @(posedge clk) begin
    if (rst)
      q <= 1'b0;
    else
      q <= d;
  end
endmodule
```

D Flip-Flop with Asynchronous Reset

```
module d_ff_async_rst (input clk, rst, d, output reg q);
  always @(posedge clk, posedge rst) begin
    if (rst)
      q <= 1'b0;
    else
      q <= d;
  end
endmodule
```

T Flip-Flop Implementation

```
module t_ff (input clk, rst, t, output reg q);
  always @(posedge clk or posedge rst) begin
    if (rst)       q <= 1'b0;
    else if (t)    q <= ~q;  // Toggle if T=1
    else           q <= q;   // Hold if T=0
  end
endmodule
```

JK Flip-Flop Implementation

```
module jk_ff (input clk, rst, j, k, output reg q);
  always @(posedge clk) begin
    if (rst) q <= 1'b0;
    else begin
      case ({j, k})
        2'b00: q <= q;          // Hold
        2'b01: q <= 1'b0;       // Reset
        2'b10: q <= 1'b1;       // Set
        2'b11: q <= ~q;         // Toggle
      endcase
    end
  end
endmodule
```

Structural vs Behavioral Sequential Design

- Structural: Instantiating primitive NAND/NOR gates to build a flip-flop. Very complex and timing-sensitive.
- Behavioral: Using 'always @(posedge clk)'. Clean, readable, and perfectly synthesizable.
- Modern Design Practice: Always use behavioral modeling for memory elements.

Common Pitfall: Incomplete Sensitivity Lists

- For latches (level-sensitive), ALL read variables must be in the sensitivity list.
- Example: ‘always @(en)‘ instead of ‘always @(en, d)‘ will cause simulation mismatches.
- For flip-flops, ONLY clock and asynchronous control signals belong in the list.

Instantiating Flip-Flops in Larger Designs

- Flip-flops are building blocks for larger sequential circuits.
- Registers: Multiple D-FFs sharing a common clock.
- Counters: D, T, or JK FFs connected in sequence.
- We will build these complex structures in the next lecture.

Summary

- 'always' blocks with edge sensitivity synthesize to flip-flops.
- Level sensitivity infers latches.
- Non-blocking assignments ($<=$) are mandatory for edge-triggered logic.
- Synchronous vs Asynchronous resets are determined by the sensitivity list.
- Behavioral case statements elegantly describe complex flip-flops like JK.

Lecture 5.12: Verilog programs for Sequential circuits (Registers and Counters)

Agenda Parallel In Parallel Out (PIPO) Shift

Register 4-Bit Binary Up/Down Counter Design

Testbench Considerations for Counters Course

Recap: Unit 1 to Unit 5

Parallel In Parallel Out (PIPO) Register

- Stores a multi-bit word simultaneously.
- Inputs (Parallel In) and Outputs (Parallel Out) are vectors.
- Uses a common clock and reset for all internal flip-flops.
- In behavioral Verilog, it's just an N-bit D-Flip-Flop.

PIPO Register in Verilog

```
module pipo_reg #(parameter N=4)
  (input clk, rst, input [N-1:0] d_in, output reg [N-1:0]
      q_out);

  always @(posedge clk or posedge rst) begin
    if (rst)
      q_out <= 0;
    else
      q_out <= d_in;
  end
endmodule
```

Binary Counters Overview

- A counter is a sequential circuit that traverses a specific state sequence.
- Up Counter: 0, 1, 2, ... N
- Down Counter: N, N-1, ... 0
- Can be built structurally from T or JK FFs, but behavioral modeling uses arithmetic (+, -).

4-Bit Binary Up/Down Counter

```
module up_down_counter (input clk, rst, up_down,
                        output reg [3:0] count);
  always @(posedge clk or posedge rst) begin
    if (rst)
      count <= 4'b0000;
    else if (up_down)          // If 1, count UP
      count <= count + 1'b1;
    else                       // If 0, count DOWN
      count <= count - 1'b1;
  end
endmodule
```

Counter Behavior and Rollover

- A 4-bit vector holds values from 0 to 15 (4'b1111).
- What happens at count + 1 when count is 15?
- In Verilog, standard arithmetic on a fixed-width vector overflows gracefully.
- $1111 + 0001 = (1)0000$. The carry is discarded, result rolls over to 0.

Testbench Considerations for Counters

```
initial begin
  clk = 0; rst = 1; up_down = 1;
  #15 rst = 0; // Release reset
  #200 up_down = 0; // Switch direction
  #200 $finish;
end
```

Course Recap: Unit 1 (MOS Transistor)

- Physics and operation of Enhancement MOSFETs.
- I-V characteristics and channel length modulation.
- Advanced concepts: High-K metal gates, FIN-FET, and Gate All Around (GAA) FETs.

Course Recap: Unit 2 (MOS Inverters)

- The Ideal Inverter and Noise Margin analysis.
- Resistive and Depletion load NMOS inverters.
- The CMOS Inverter: Structure, operation, and VTC.
- Why CMOS won: Zero static power dissipation.

Course Recap: Unit 3 (MOS Circuits)

- CMOS Combinational Logic: NAND, NOR, and Complex AOI/OAI gates.
- PUN (PMOS) and PDN (NMOS) design rules (Series/Parallel duality).
- CMOS Sequential Logic: SR latches, D-latches, and clocking strategies.

Course Recap: Unit 4 (Fabrication & Layout)

- The Y-Chart and VLSI Design Flow.
- CMOS Fabrication Steps: Photolithography, oxidation, etching, doping.
- Lambda based design rules and Stick Diagrams.
- Translating transistor schematics into physical silicon layout.

Course Recap: Unit 5 (Introduction to Verilog)

- Hardware Description Languages vs Software.
- Lexical conventions, data types (reg/wire), and modeling styles (Dataflow/Behavioral).
- Combinational Logic Design (Muxes, Decoders, Parity).
- Sequential Logic Design (Flip-flops, Counters) and Testbenches.

Course Conclusion

- From semiconductor physics (Unit 1) to RTL programming (Unit 5).
- You now understand the complete VLSI abstraction stack.
- The foundation is set for advanced FPGA design and ASIC synthesis.