Vlsi (4353206) - Problem Solver

Antigravity AI

June 4, 2026

Contents

CHAPTER 1

Introduction to VLSI

Although the introduction to Very Large Scale Integration (VLSI) is heavily rooted in theoretical concepts such as design flow and architecture, several quantitative aspects are essential for understanding scaling, performance, and manufacturing. This chapter focuses on the computational principles underlying Moore's Law, power dissipation in CMOS circuits, and basic manufacturing yield calculations.

1.1 Moore's Law and Transistor Scaling

Moore's Law states that the number of transistors on a microchip doubles approximately every 18 to 24 months. This exponential growth allows for the estimation of future transistor counts based on historical data.

Methodology:

Predicting Transistor Count using Moores Law To calculate the expected number of transistors at a future date based on Moore's Law:

1. Identify the initial transistor count N_0 at the initial year Y_0.
2. Identify the target year Y_t.
3. Determine the doubling period T_d in years (usually 1.5 years for 18 months or 2.0 years for 24 months).
4. Calculate the elapsed time $\Delta Y = Y_t - Y_0$.
5. Calculate the number of doubling periods $n = \frac{\Delta Y}{T_d}$.
6. Compute the expected number of transistors $N_t = N_0 \times 2^n$.

Worked Example:

Calculating Future Transistor Count A microprocessor introduced in 2010 had 1.2 billion transistors. Assuming Moore's Law holds with a doubling period of 2 years, estimate the number of transistors on a similar class microprocessor in the year 2020.

Solution:

1. **Identify given parameters:** Initial count $N_0 = 1.2 \times 10^9$ transistors. Initial year $Y_0 = 2010$. Target year $Y_t = 2020$. Doubling period $T_d = 2$ years.
2. **Calculate elapsed time:** $\Delta Y = 2020 - 2010 = 10$ years.
3. **Calculate number of doubling periods:** $n = \frac{10}{2} = 5$ periods.
4. **Compute expected transistor count:**

$$\begin{aligned} N_t &= N_0 \times 2^n \\ &= 1.2 \times 10^9 \times 2^5 \\ &= 1.2 \times 10^9 \times 32 \\ &= 38.4 \times 10^9 \end{aligned}$$

The estimated number of transistors in 2020 is 38.4 billion.

1.2 Dynamic Power Dissipation

In CMOS circuits, one of the primary sources of power consumption is dynamic power dissipation, which occurs during the switching of transistors.

Methodology:

Calculating Dynamic Power Dissipation The dynamic power dissipation ($P_{dynamic}$) of a CMOS logic gate or circuit is calculated using the formula:

$$P_{dynamic} = \alpha \cdot C_L \cdot V_{DD}^2 \cdot f$$

Where:

- α is the activity factor (the probability that a node transitions from 0 to 1 during a clock cycle).
- C_L is the total load capacitance (in Farads).
- V_{DD} is the supply voltage (in Volts).
- f is the operating clock frequency (in Hertz).

Steps to calculate:

1. Convert all given values to their standard SI units (e.g. pF to F, MHz to Hz).
2. Square the supply voltage V_{DD}.
3. Multiply the activity factor, load capacitance, squared voltage, and frequency.
4. Express the final result in Watts (W) or milliWatts (mW).

Worked Example:

Dynamic Power of a CMOS Logic Gate A CMOS logic gate operates at a clock frequency of 500 MHz with a supply voltage of 1.8 V. The load capacitance is 15 pF and the activity factor is 0.2. Calculate the dynamic power dissipation.
Solution:

1. **Identify and convert given values:**
   - $f = 500$ MHz $= 500 \times 10^6$ Hz
   - $V_{DD} = 1.8$ V
   - $C_L = 15$ pF $= 15 \times 10^{-12}$ F
   - $\alpha = 0.2$
2. **Apply the dynamic power formula:**

$$\begin{aligned} P_{dynamic} &= \alpha \cdot C_L \cdot V_{DD}^2 \cdot f \\ &= 0.2 \times (15 \times 10^{-12}) \times (1.8)^2 \times (500 \times 10^6) \end{aligned}$$

3. **Compute the result step-by-step:**

$$\begin{aligned}(1.8)^2 &= 3.24\\ 0.2 \times 15 \times 10^{-12} &= 3 \times 10^{-12}\\ P_{dynamic} &= 3 \times 10^{-12} \times 3.24 \times 500 \times 10^6\\ &= 9.72 \times 10^{-12} \times 500 \times 10^6\\ &= 4860 \times 10^{-6} \text{ W}\end{aligned}$$

4. **Final Answer:** $P_{dynamic} = 4.86$ mW.

1.3 Die Yield Calculation

In VLSI manufacturing, not all fabricated chips (dies) on a silicon wafer function correctly due to defects. Yield is the fraction of functional dies. The Poisson yield model is one of the foundational models used to estimate this.

Methodology:

Calculating Die Yield using Poisson Model The Poisson yield model assumes that defects are randomly distributed across the wafer. The yield (Y) is given by:

$$Y = e^{-A \cdot D_0}$$

Where:

- Y is the die yield (percentage of good dies).
- A is the die area (in cm^2).
- D_0 is the defect density (number of defects per cm^2).

Steps to calculate:

1. Determine the die area A in cm^2 (convert from mm^2 if necessary, noting that 100 mm^2 = 1 cm^2).
2. Determine the defect density D_0 in defects/cm^2.
3. Multiply the area by the defect density to find the expected number of defects per die ($A \cdot D_0$).
4. Compute the exponential $e^{-A \cdot D_0}$.
5. Multiply by 100 to express the yield as a percentage.

Worked Example:

Estimating Die Yield A silicon wafer is manufactured with a defect density of 0.5 defects per cm^2. A single microprocessor die has dimensions of 12 mm by 12 mm. Calculate the expected die yield using the Poisson model.

Solution:

1. **Calculate Die Area:**

$$\begin{aligned}\text{Length} &= 12 \text{ mm} = 1.2 \text{ cm}\\ \text{Width} &= 12 \text{ mm} = 1.2 \text{ cm}\\ A &= 1.2 \times 1.2 = 1.44 \text{ cm}^2\end{aligned}$$

2. **Identify Defect Density:** $D_0 = 0.5$ defects/cm^2.

3. **Calculate Expected Defects per Die:**

$$A \cdot D_0 = 1.44 \times 0.5 = 0.72$$

4. **Apply Poisson Yield Formula:**

$$Y = e^{-A \cdot D_0}$$
$$Y = e^{-0.72}$$
$$Y \approx 0.4867$$

5. **Final Answer:** The expected die yield is approximately 48.67%.

1.4 Wafer Yield and Cost Estimation

To estimate the cost of a single functional die, we must first determine how many total dies can fit on a wafer and then apply the die yield.

Methodology:

Estimating Dies per Wafer and Die Cost To calculate the total number of dies per wafer (N_{dies}) and the cost per functional die (C_{die}):

1. Calculate the wafer area $A_{wafer} = \pi \cdot r^2$, where r is the wafer radius.
2. Approximate the total dies per wafer using the formula:

$$N_{dies} \approx \frac{\pi \cdot r^2}{A_{die}} - \frac{\pi \cdot 2r}{\sqrt{2 \cdot A_{die}}}$$

   (The second term accounts for the wasted area at the edges of the round wafer).
3. Calculate the number of functional dies: $N_{good} = N_{dies} \cdot Y$.
4. Calculate the cost per functional die: $C_{die} = \frac{C_{wafer}}{N_{good}}$, where C_{wafer} is the total manufacturing cost of the wafer.

Worked Example:

Calculating Cost per Functional Die A fabrication plant processes 300 mm diameter wafers at a cost of \$4000 per wafer. The die area is 150 mm^2, and the calculated die yield is 60%. Estimate the number of dies per wafer and the manufacturing cost of a single functional die.
Solution:

1. **Identify parameters:** Wafer diameter = 300 mm $\Rightarrow$ radius $r = 150$ mm. Die area $A_{die} = 150$ mm^2. Wafer cost $C_{wafer} = \$4000$. Yield $Y = 0.60$.
2. **Estimate Total Dies per Wafer (N_{dies}):**

$$\begin{aligned} N_{dies} &\approx \frac{\pi \cdot (150)^2}{150} - \frac{\pi \cdot 300}{\sqrt{2 \cdot 150}} \\ &\approx \frac{\pi \cdot 22500}{150} - \frac{942.48}{\sqrt{300}} \\ &\approx 150\pi - \frac{942.48}{17.32} \\ &\approx 471.24 - 54.42 \\ &\approx 416 \text{ dies (rounded down to nearest whole die)} \end{aligned}$$

3. **Calculate Functional (Good) Dies:**

$$\begin{aligned} N_{good} &= N_{dies} \times Y \\ &= 416 \times 0.60 \\ &= 249.6 \approx 249 \text{ good dies} \end{aligned}$$

4. **Calculate Cost per Functional Die:**

$$\begin{aligned} C_{die} &= \frac{4000}{249} \\ &\approx \$16.06 \end{aligned}$$

The cost per functional die is approximately \$16.06.

CHAPTER 2

MOS Transistor

This chapter focuses on the computational aspects of MOS transistors, including drain current calculations, device parameter determination, and the effects of different scaling methodologies on MOSFET parameters.

2.1 MOSFET Current-Voltage Characteristics

The operation of a MOS transistor is governed by its terminal voltages. Depending on the values of the Gate-to-Source voltage (V_{GS}), Drain-to-Source voltage (V_{DS}), and the Threshold voltage (V_T), the transistor operates in different regions.

Methodology:

MOSFET Drain Current Calculation To calculate the drain current (I_D) for an n-channel MOSFET (nMOS), follow these steps:

Step 1: Calculate Process Parameters Determine the oxide capacitance per unit area (C_{ox}) and the process transconductance parameter (k'_n):

$$C_{ox} = \frac{\epsilon_{ox}}{t_{ox}}$$

$$k'_n = \mu_n C_{ox}$$

where ϵ_{ox} is the permittivity of the oxide ($3.9 \times 8.85 \times 10^{-14}$ F/cm), t_{ox} is the oxide thickness, and μ_n is the electron mobility.

Step 2: Calculate Device Transconductance Parameter

$$k_n = k'_n \frac{W}{L} = \mu_n C_{ox} \frac{W}{L}$$

where W is the channel width and L is the channel length.

Step 3: Determine the Region of Operation

1. **Cut-off Region:** If $V_{GS} < V_T$, the transistor is OFF.
2. **Linear (Triode) Region:** If $V_{GS} \geq V_T$ and $V_{DS} < V_{GS} - V_T$.
3. **Saturation Region:** If $V_{GS} \geq V_T$ and $V_{DS} \geq V_{GS} - V_T$.

Step 4: Apply the Appropriate Current Equation

- **Cut-off:**

$$I_D = 0$$

- **Linear:**

$$I_D = k_n \left[(V_{GS} - V_T) V_{DS} - \frac{V_{DS}^2}{2} \right]$$

- **Saturation:**

$$I_D = \frac{1}{2} k_n (V_{GS} - V_T)^2 (1 + \lambda V_{DS})$$

(Note: If the channel-length modulation parameter λ is zero or not given, assume $1 + \lambda V_{DS} = 1$).

Worked Example:

Drain Current in Different Operating Regions An nMOS transistor has a threshold voltage $V_T = 1.0$ V, device transconductance parameter $k_n = 50\ \mu\text{A/V}^2$, and channel-length modulation parameter $\lambda = 0\ \text{V}^{-1}$. Calculate the drain current I_D for the following cases: (a) $V_{GS} = 0.5$ V, $V_{DS} = 2.0$ V (b) $V_{GS} = 2.0$ V, $V_{DS} = 0.5$ V (c) $V_{GS} = 2.0$ V, $V_{DS} = 3.0$ V

Solution:

Case (a):

1. **Given:** $V_{GS} = 0.5$ V, $V_T = 1.0$ V.
2. **Check region:** Since $V_{GS}(0.5\text{ V}) < V_T(1.0\text{ V})$, the transistor is in the **Cut-off region**.
3. **Calculate current:**
$$I_D = 0\text{ A}$$

Case (b):

1. **Given:** $V_{GS} = 2.0$ V, $V_{DS} = 0.5$ V, $V_T = 1.0$ V.
2. **Check condition 1:** $V_{GS} \geq V_T$ (2.0 V $\geq$ 1.0 V). Transistor is ON.
3. **Check condition 2:** Calculate overdrive voltage $V_{ov} = V_{GS} - V_T = 2.0 - 1.0 = 1.0$ V.
4. **Check region:** Since $V_{DS}(0.5\text{ V}) < V_{GS} - V_T(1.0\text{ V})$, it operates in the **Linear region**.
5. **Calculate current:**
$$I_D = k_n \left[(V_{GS} - V_T)V_{DS} - \frac{V_{DS}^2}{2}\right]$$
$$I_D = 50 \times 10^{-6} \left[(1.0)(0.5) - \frac{0.5^2}{2}\right]$$
$$I_D = 50 \times 10^{-6}\,[0.5 - 0.125] = 50 \times 10^{-6} \times 0.375$$
$$I_D = 18.75\ \mu\text{A}$$

Case (c):

1. **Given:** $V_{GS} = 2.0$ V, $V_{DS} = 3.0$ V, $V_T = 1.0$ V.
2. **Check condition 1:** $V_{GS} \geq V_T$. Transistor is ON.
3. **Check condition 2:** $V_{ov} = V_{GS} - V_T = 1.0$ V.
4. **Check region:** Since $V_{DS}(3.0\text{ V}) \geq V_{GS} - V_T(1.0\text{ V})$, it operates in the **Saturation region**.
5. **Calculate current:**
$$I_D = \frac{1}{2}k_n(V_{GS} - V_T)^2$$
$$I_D = \frac{1}{2}(50 \times 10^{-6})(1.0)^2$$
$$I_D = 25\ \mu\text{A}$$

Worked Example:

Calculation of Device Parameters Consider an n-channel MOSFET with the following parameters: electron mobility $\mu_n = 500\ \text{cm}^2/\text{V} \cdot \text{s}$, oxide thickness $t_{ox} = 10$ nm, channel length $L = 1\ \mu$m, and channel width $W = 5\ \mu$m. The permittivity of silicon dioxide is $\epsilon_{ox} = 3.45 \times 10^{-13}$ F/cm. Determine the oxide capacitance per unit area C_{ox}, the process transconductance parameter k_n', and the device transconductance parameter k_n.

Solution:

Step 1: Calculate C_{ox} First, convert t_{ox} to cm to match the units of ϵ_{ox}:

$$t_{ox} = 10 \text{ nm} = 10 \times 10^{-7} \text{ cm} = 10^{-6} \text{ cm}$$

$$C_{ox} = \frac{\epsilon_{ox}}{t_{ox}} = \frac{3.45 \times 10^{-13} \text{ F/cm}}{10^{-6} \text{ cm}}$$

$$C_{ox} = 3.45 \times 10^{-7} \text{ F/cm}^2$$

Step 2: Calculate k'_n

$$k'_n = \mu_n C_{ox}$$

$$k'_n = (500 \text{ cm}^2/\text{V} \cdot \text{s}) \times (3.45 \times 10^{-7} \text{ F/cm}^2)$$

$$k'_n = 1.725 \times 10^{-4} \text{ A/V}^2 = 172.5 \ \mu\text{A/V}^2$$

Step 3: Calculate k_n

$$k_n = k'_n \frac{W}{L}$$

$$k_n = (172.5 \ \mu\text{A/V}^2) \times \frac{5 \ \mu\text{m}}{1 \ \mu\text{m}}$$

$$k_n = 172.5 \times 5 = 862.5 \ \mu\text{A/V}^2$$

2.2 MOSFET Scaling Models

Scaling reduces the physical dimensions of the MOSFET to increase packing density and improve performance. A scaling factor $S > 1$ (often represented as α in some texts) is used to scale down the device dimensions.

Methodology:

MOSFET Scaling Computations Let $S > 1$ be the scaling factor. To find the parameters of a scaled device, multiply the original parameter value by the corresponding scaling multiplier based on the chosen scaling model.

1. Full-Voltage (Constant-Field) Scaling In this model, physical dimensions and power supply voltages are scaled down by the same factor S, keeping the internal electric fields constant.

- **Dimensions** (L, W, t_{ox}): Scaled by $1/S$
- **Voltages** (V_{DD}, V_T): Scaled by $1/S$
- **Gate Oxide Capacitance** (C_{ox}): Scaled by S (since $1/t_{ox}$ scales by S)
- **Total Gate Capacitance** $(C_g = C_{ox}WL)$: Scaled by $S \times (1/S) \times (1/S) = 1/S$
- **Drain Current** (I_D): Scaled by $1/S$
- **Power Dissipation** $(P = V \times I)$: Scaled by $(1/S) \times (1/S) = 1/S^2$
- **Gate Delay** $(\tau = C_g V/I_D)$: Scaled by $(1/S \times 1/S)/(1/S) = 1/S$

2. Constant-Voltage Scaling In this model, only the physical dimensions are scaled by S, while the power supply voltages remain constant to maintain compatibility with existing components.

- **Dimensions** (L, W, t_{ox}): Scaled by $1/S$
- **Voltages** (V_{DD}, V_T): Scaled by 1 (Unchanged)
- **Gate Oxide Capacitance** (C_{ox}): Scaled by S
- **Total Gate Capacitance** (C_g): Scaled by $1/S$
- **Drain Current** (I_D): Scaled by S (since V is constant and C_{ox} scales by S)

- **Power Dissipation** ($P = V \times I$): Scaled by $1 \times S = S$
- **Gate Delay** ($\tau = C_g V / I_D$): Scaled by $(1/S \times 1)/S = 1/S^2$

Worked Example:

Full-Voltage Scaling Effects A MOSFET has a gate capacitance $C_g = 50$ fF, a drain current $I_D = 2$ mA, power dissipation $P = 10$ mW, and gate delay $\tau = 100$ ps. The device is scaled down using full-voltage (constant-field) scaling with a scaling factor $S = 2$. Calculate the new values for gate capacitance, drain current, power dissipation, and gate delay.

Solution:

For full-voltage scaling with $S = 2$:

1. **New Gate Capacitance (C_g'):** Scales by $1/S$.

$$C_g' = C_g \times \frac{1}{S} = 50 \text{ fF} \times \frac{1}{2} = 25 \text{ fF}$$

2. **New Drain Current (I_D'):** Scales by $1/S$.

$$I_D' = I_D \times \frac{1}{S} = 2 \text{ mA} \times \frac{1}{2} = 1 \text{ mA}$$

3. **New Power Dissipation (P'):** Scales by $1/S^2$.

$$P' = P \times \frac{1}{S^2} = 10 \text{ mW} \times \frac{1}{2^2} = 10 \text{ mW} \times \frac{1}{4} = 2.5 \text{ mW}$$

4. **New Gate Delay (τ'):** Scales by $1/S$.

$$\tau' = \tau \times \frac{1}{S} = 100 \text{ ps} \times \frac{1}{2} = 50 \text{ ps}$$

Worked Example:

Constant-Voltage Scaling Effects Assume the same initial device from the previous example: $C_g = 50$ fF, $I_D = 2$ mA, $P = 10$ mW, and $\tau = 100$ ps. This time, the device is scaled down using constant-voltage scaling with a scaling factor $S = 2$. Calculate the new values for these four parameters.

Solution:

For constant-voltage scaling with $S = 2$:

1. **New Gate Capacitance (C_g'):** Scales by $1/S$.

$$C_g' = C_g \times \frac{1}{S} = 50 \text{ fF} \times \frac{1}{2} = 25 \text{ fF}$$

2. **New Drain Current (I_D'):** Scales by S.

$$I_D' = I_D \times S = 2 \text{ mA} \times 2 = 4 \text{ mA}$$

3. **New Power Dissipation (P'):** Scales by S.

$$P' = P \times S = 10 \text{ mW} \times 2 = 20 \text{ mW}$$

4. **New Gate Delay (τ'):** Scales by $1/S^2$.

$$\tau' = \tau \times \frac{1}{S^2} = 100 \text{ ps} \times \frac{1}{4} = 25 \text{ ps}$$

2.3 Gradual Channel Approximation and Sizing

The gradual channel approximation simplifies the analysis of MOSFET current-voltage behavior by assuming that the variation of the electric field along the channel (x-direction) is much less than the variation perpendicular to the channel (y-direction). This allows for 1D analysis of the channel potential.

Methodology:

Determining MOSFET Dimensions for a Target Current Often, circuit designers must size a transistor (choose W/L) to provide a specific drain current under given voltage conditions.
Step 1: Identify Given Parameters Identify target I_D, applied voltages V_{GS}, V_{DS}, threshold voltage V_T, and process parameter k_n'.
Step 2: Determine Region of Operation Compare V_{DS} and $V_{GS} - V_T$ to find whether the target operation is in the linear or saturation region.
Step 3: Solve for the Aspect Ratio (W/L) Rearrange the corresponding current equation:

- For **Linear Region**:

$$\frac{W}{L} = \frac{I_D}{k_n' \left[(V_{GS} - V_T)V_{DS} - \frac{V_{DS}^2}{2} \right]}$$

- For **Saturation Region**:

$$\frac{W}{L} = \frac{2I_D}{k_n'(V_{GS} - V_T)^2(1 + \lambda V_{DS})}$$

Step 4: Find W or L If L is given (usually the minimum feature size of the process), find $W = L \times (W/L)$.

Worked Example:

MOSFET Transistor Sizing An nMOS transistor in a 0.18 μm process must conduct a saturation current of $I_D = 100$ μA when a gate-to-source voltage of $V_{GS} = 1.2$ V is applied. The process parameters are $k_n' = 200$ μA/V^2, $V_T = 0.4$ V, and $\lambda = 0$ V^{-1}. Assuming the channel length is kept at the minimum size ($L = 0.18$ μm), determine the required channel width W.
Solution:
Step 1: Identify operation region and parameters

- $I_D = 100$ μA
- $V_{GS} = 1.2$ V
- $V_T = 0.4$ V
- $k_n' = 200$ μA/V^2
- The problem states the transistor is in saturation.

Step 2: Setup the saturation current equation Since $\lambda = 0$, the equation is:

$$I_D = \frac{1}{2} k_n' \frac{W}{L} (V_{GS} - V_T)^2$$

Step 3: Solve for W/L

$$\frac{W}{L} = \frac{2I_D}{k_n'(V_{GS} - V_T)^2}$$

Substitute the values:

$$\frac{W}{L} = \frac{2 \times 100 \ \mu\text{A}}{200 \ \mu\text{A/V}^2 \times (1.2 - 0.4)^2}$$

$$\frac{W}{L} = \frac{200}{200 \times (0.8)^2}$$

$$\frac{W}{L} = \frac{1}{0.64} = 1.5625$$

Step 4: Calculate the width W Given $L = 0.18$ μm:

$$W = 1.5625 \times L$$

$$W = 1.5625 \times 0.18 \ \mu\text{m} = 0.28125 \ \mu\text{m}$$

The required channel width is 0.28125 μm.

CHAPTER 3

MOS Inverters

The CMOS inverter is a fundamental building block in digital VLSI design. This chapter focuses on the core computational methods used to analyze CMOS inverters, including switching characteristics, noise margins, propagation delays, power dissipation, and symmetrical sizing.

3.1 Switching Characteristics

Methodology:

CMOS Inverter Switching Threshold Voltage The switching threshold voltage (V_M or V_{th}) is the point where the input voltage equals the output voltage. At this specific point, both the NMOS and PMOS transistors operate in the saturation region.

1. **Define the threshold condition:** Set $V_{in} = V_{out} = V_M$.
2. **Equate drain currents:** In saturation, the current through the NMOS equals the current through the PMOS ($I_{DS,n} = I_{SD,p}$).

$$\frac{1}{2}k_n(V_M - V_{tn})^2 = \frac{1}{2}k_p(V_{DD} - V_M - |V_{tp}|)^2$$

   where the transconductance parameters are $k_n = \mu_n C_{ox}(W/L)_n$ and $k_p = \mu_p C_{ox}(W/L)_p$.

3. **Solve for V_M:** Let $k_R = \frac{k_p}{k_n}$. Taking the square root of both sides and rearranging yields:

$$V_M = \frac{V_{tn} + \sqrt{k_R}(V_{DD} - |V_{tp}|)}{1 + \sqrt{k_R}}$$

Worked Example:

Calculate the switching threshold voltage for a given CMOS inverter Given a CMOS inverter with a power supply $V_{DD} = 5\text{V}$, and threshold voltages $V_{tn} = 1\text{V}$ and $V_{tp} = -1\text{V}$. The transconductance parameters are $k_n = 100\mu\text{A/V}^2$ and $k_p = 40\mu\text{A/V}^2$. Determine the switching threshold voltage V_M.

Step 1: Calculate the ratio k_R.

$$k_R = \frac{k_p}{k_n} = \frac{40}{100} = 0.4$$

Step 2: Find the square root of k_R.

$$\sqrt{k_R} = \sqrt{0.4} \approx 0.632$$

Step 3: Substitute the values into the V_M formula.

$$V_M = \frac{V_{tn} + \sqrt{k_R}(V_{DD} - |V_{tp}|)}{1 + \sqrt{k_R}}$$

$$V_M = \frac{1 + 0.632(5 - |-1|)}{1 + 0.632}$$

$$V_M = \frac{1 + 0.632(4)}{1.632}$$
$$V_M = \frac{1 + 2.528}{1.632} = \frac{3.528}{1.632}$$
$$V_M \approx 2.16\text{V}$$

3.2 Noise Margins

Methodology:

Calculation of Critical Voltages and Noise Margins Noise margins determine the maximum noise voltage that can be added to an input signal without altering the logic state. For an ideal symmetric CMOS inverter ($k_n = k_p$ and $V_{tn} = |V_{tp}| = V_t$):

1. **Identify Output Voltages:** For a CMOS inverter, $V_{OH} = V_{DD}$ and $V_{OL} = 0\text{V}$.
2. **Calculate V_{IL} (Maximum LOW Input Voltage):** This is the input voltage where the slope of the voltage transfer characteristic (VTC) is -1.
$$V_{IL} = \frac{3V_{DD} + 2V_t}{8}$$
3. **Calculate V_{IH} (Minimum HIGH Input Voltage):** This is the input voltage where the slope of the VTC is -1.
$$V_{IH} = \frac{5V_{DD} - 2V_t}{8}$$
4. **Compute Noise Margins:**
   - Low Noise Margin: $NM_L = V_{IL} - V_{OL}$
   - High Noise Margin: $NM_H = V_{OH} - V_{IH}$

Worked Example:

Determine the noise margins of a symmetric CMOS inverter A symmetric CMOS inverter operates at a supply voltage $V_{DD} = 3.3\text{V}$ with threshold voltages $V_{tn} = |V_{tp}| = 0.6\text{V}$. Calculate V_{IL}, V_{IH}, and the noise margins NM_L and NM_H.

Step 1: State output voltage levels.

$$V_{OH} = 3.3\text{V}, \quad V_{OL} = 0\text{V}$$

Step 2: Calculate V_{IL}.

$$V_{IL} = \frac{3(3.3) + 2(0.6)}{8} = \frac{9.9 + 1.2}{8} = \frac{11.1}{8} = 1.3875\text{V}$$

Step 3: Calculate V_{IH}.

$$V_{IH} = \frac{5(3.3) - 2(0.6)}{8} = \frac{16.5 - 1.2}{8} = \frac{15.3}{8} = 1.9125\text{V}$$

Step 4: Compute Noise Margins.

$$NM_L = V_{IL} - V_{OL} = 1.3875\text{V} - 0\text{V} = 1.3875\text{V}$$

$$NM_H = V_{OH} - V_{IH} = 3.3\text{V} - 1.9125\text{V} = 1.3875\text{V}$$

3.3 Propagation Delays

Methodology:

RC Delay Approximation for Inverters The propagation delay of a CMOS inverter can be estimated by modeling the transistors as equivalent resistors charging or discharging a load capacitance C_L.

1. **Identify the parameters:** Load capacitance (C_L), equivalent NMOS resistance ($R_{eq,n}$), and equivalent PMOS resistance ($R_{eq,p}$).
2. **Calculate High-to-Low Propagation Delay (t_{pHL}):** The time taken to discharge C_L to 50% of V_{DD} through the NMOS.
$$t_{pHL} = 0.69 R_{eq,n} C_L$$
3. **Calculate Low-to-High Propagation Delay (t_{pLH}):** The time taken to charge C_L to 50% of V_{DD} through the PMOS.
$$t_{pLH} = 0.69 R_{eq,p} C_L$$
4. **Calculate Average Propagation Delay (t_p):**
$$t_p = \frac{t_{pHL} + t_{pLH}}{2}$$

Worked Example:

Calculate inverter propagation delay using RC approximation A CMOS inverter drives a load capacitance $C_L = 50\text{fF}$. The effective ON-resistances are given as $R_{eq,n} = 3\text{k}\Omega$ for the NMOS and $R_{eq,p} = 6\text{k}\Omega$ for the PMOS. Calculate t_{pHL}, t_{pLH}, and the overall propagation delay t_p.

Step 1: Calculate High-to-Low Delay (t_{pHL}).

$$t_{pHL} = 0.69 \times R_{eq,n} \times C_L$$

$$t_{pHL} = 0.69 \times (3 \times 10^3) \times (50 \times 10^{-15})$$

$$t_{pHL} = 0.69 \times 150 \times 10^{-12} = 103.5\text{ps}$$

Step 2: Calculate Low-to-High Delay (t_{pLH}).

$$t_{pLH} = 0.69 \times R_{eq,p} \times C_L$$

$$t_{pLH} = 0.69 \times (6 \times 10^3) \times (50 \times 10^{-15})$$

$$t_{pLH} = 0.69 \times 300 \times 10^{-12} = 207.0\text{ps}$$

Step 3: Calculate Average Propagation Delay (t_p).

$$t_p = \frac{103.5\text{ps} + 207.0\text{ps}}{2} = \frac{310.5\text{ps}}{2} = 155.25\text{ps}$$

3.4 Symmetrical Sizing

Methodology:

Sizing for Equal Rise and Fall Times To ensure the inverter operates symmetrically with equal rise and fall times (which implies $t_{pHL} = t_{pLH}$), the charging and discharging currents must be equal.

1. **Equate transconductance parameters:** For symmetric operation, k_n must equal k_p.
$$\mu_n C_{ox} \left(\frac{W}{L}\right)_n = \mu_p C_{ox} \left(\frac{W}{L}\right)_p$$

2. **Cancel out oxide capacitance (C_{ox}):**

$$\mu_n \left(\frac{W}{L}\right)_n = \mu_p \left(\frac{W}{L}\right)_p$$

3. **Solve for the PMOS aspect ratio:**

$$\left(\frac{W}{L}\right)_p = \frac{\mu_n}{\mu_p} \left(\frac{W}{L}\right)_n$$

Because electron mobility μ_n is typically 2 to 3 times the hole mobility μ_p, the PMOS transistor must be wider than the NMOS by the same factor.

Worked Example:

Determine the required PMOS width for a symmetric inverter An NMOS transistor in a CMOS inverter has an aspect ratio $(W/L)_n = 3$. The fabrication process yields mobilities where $\mu_n = 2.4\mu_p$. Calculate the required aspect ratio $(W/L)_p$ of the PMOS transistor to achieve symmetric switching characteristics. If both transistors have a minimum channel length $L = 0.5\mu$m, determine the physical widths W_n and W_p.
Step 1: Find the required PMOS aspect ratio.

$$\left(\frac{W}{L}\right)_p = \frac{\mu_n}{\mu_p} \left(\frac{W}{L}\right)_n$$

Substitute $\mu_n = 2.4\mu_p$:

$$\left(\frac{W}{L}\right)_p = 2.4 \times 3 = 7.2$$

Step 2: Calculate the physical width of the NMOS (W_n).

$$W_n = \left(\frac{W}{L}\right)_n \times L = 3 \times 0.5\mu\text{m} = 1.5\mu\text{m}$$

Step 3: Calculate the physical width of the PMOS (W_p).

$$W_p = \left(\frac{W}{L}\right)_p \times L = 7.2 \times 0.5\mu\text{m} = 3.6\mu\text{m}$$

3.5 Power Dissipation

Methodology:

Total Power Dissipation Analysis Power consumed in a CMOS circuit consists of three main components: dynamic, static, and short-circuit power.

1. **Dynamic Power (P_{dyn}):** Power dissipated during the charging and discharging of the load capacitance.

$$P_{dyn} = C_L V_{DD}^2 f_{clk} \alpha$$

where f_{clk} is the switching frequency and α is the activity factor (typically 0.5 to 1).

2. **Static Power (P_{stat}):** Power consumed due to leakage current when the circuit is idle.

$$P_{stat} = I_{leakage} \times V_{DD}$$

3. **Short-Circuit Power (P_{sc}):** Power dissipated because both NMOS and PMOS are briefly ON during signal transitions. Often ignored in first-order approximations, but can be computed if rise/fall times (t_r) are known.

4. **Total Power (P_{total}):**

$$P_{total} = P_{dyn} + P_{stat} + P_{sc}$$

Worked Example:

Calculate power dissipation components for a CMOS inverter A CMOS inverter drives a load capacitance of 1.2pF and operates at a supply voltage $V_{DD} = 2.5$V. The input clock frequency is 500MHz with an activity factor $\alpha = 0.1$. The device has a leakage current of 15nA. Assuming short-circuit power is negligible, calculate the dynamic power, static power, and total power dissipation.

Step 1: Calculate Dynamic Power (P_{dyn}).

$$P_{dyn} = C_L V_{DD}^2 f_{clk} \alpha$$

$$P_{dyn} = (1.2 \times 10^{-12}) \times (2.5)^2 \times (500 \times 10^6) \times 0.1$$

$$P_{dyn} = (1.2 \times 10^{-12}) \times 6.25 \times (50 \times 10^6)$$

$$P_{dyn} = 375 \times 10^{-6}\text{W} = 375\mu\text{W}$$

Step 2: Calculate Static Power (P_{stat}).

$$P_{stat} = I_{leakage} \times V_{DD}$$

$$P_{stat} = (15 \times 10^{-9}) \times 2.5 = 37.5 \times 10^{-9}\text{W} = 0.0375\mu\text{W}$$

Step 3: Calculate Total Power (P_{total}).

$$P_{total} = P_{dyn} + P_{stat}$$

$$P_{total} = 375\mu\text{W} + 0.0375\mu\text{W} = 375.0375\mu\text{W}$$

3.6 Ring Oscillators

Methodology:

Ring Oscillator Frequency Calculation A ring oscillator is created by connecting an odd number of inverters in a continuous feedback loop. The cascaded delay determines the oscillation frequency.

1. **Determine the total delay time:** For N stages (where N is an odd integer) and an average propagation delay t_p per stage, the total time for a signal to propagate through the chain is $N \times t_p$.
2. **Calculate the time period (T):** A complete oscillation cycle requires a transition from HIGH to LOW and back to HIGH, effectively passing through the chain twice.

$$T = 2 \times N \times t_p$$

3. **Calculate the oscillation frequency (f_{osc}):**

$$f_{osc} = \frac{1}{T} = \frac{1}{2 \times N \times t_p}$$

Worked Example:

Determine the parameters of a 7-stage ring oscillator A ring oscillator is built using 7 identical CMOS inverters. If each inverter has an average propagation delay of 80ps, calculate the period and the frequency of oscillation.

Step 1: Identify given parameters.

$$N = 7, \quad t_p = 80\text{ps} = 80 \times 10^{-12}\text{s}$$

Step 2: Calculate the time period (T).

$$T = 2 \times N \times t_p$$

$$T = 2 \times 7 \times 80\text{ps} = 1120\text{ps} = 1.12\text{ns}$$

Step 3: Calculate the frequency of oscillation (f_{osc}).

$$f_{osc} = \frac{1}{T} = \frac{1}{1.12 \times 10^{-9}} \approx 892.86 \times 10^{6}\text{Hz} = 892.86\text{MHz}$$

CHAPTER 4
MOS Circuits

4.1 CMOS Inverter Voltage Characteristics and Noise Margins

The operation of a CMOS inverter is defined by its Voltage Transfer Characteristic (VTC). Analyzing the VTC allows us to determine critical threshold voltages and calculate the noise margins, which represent the circuit's immunity to electrical noise.

Methodology:

Calculating Critical Voltages and Noise Margins To analyze the noise immunity of a CMOS logic gate, follow these computational steps:

1. **Identify Output Voltage Levels:**
   - V_{OH} (Output High Voltage): The maximum voltage at the output when the logic state is high.
   - V_{OL} (Output Low Voltage): The minimum voltage at the output when the logic state is low.
2. **Identify Input Threshold Voltages:**
   - V_{IL} (Input Low Voltage): The maximum input voltage that is still interpreted as a valid logic low. It is found where the slope of the VTC $\frac{dV_{out}}{dV_{in}} = -1$.
   - V_{IH} (Input High Voltage): The minimum input voltage that is still interpreted as a valid logic high. It is found where the slope of the VTC $\frac{dV_{out}}{dV_{in}} = -1$.
3. **Calculate Noise Margin Low (NM_L):** This is the difference between the maximum valid logic low input and the maximum logic low output.

$$NM_L = V_{IL} - V_{OL}$$

4. **Calculate Noise Margin High (NM_H):** This is the difference between the minimum logic high output and the minimum valid logic high input.

$$NM_H = V_{OH} - V_{IH}$$

Worked Example:

Calculating Noise Margins of a CMOS Inverter **Problem:** A CMOS inverter operating at $V_{DD} = 5$V has the following critical voltages determined from its VTC: $V_{OH} = 4.9$V, $V_{OL} = 0.1$V, $V_{IL} = 1.8$V, and $V_{IH} = 3.2$V. Calculate the low-level noise margin (NM_L) and high-level noise margin (NM_H).
Solution:

1. **Extract given values:**
   - $V_{OH} = 4.9$V
   - $V_{OL} = 0.1$V
   - $V_{IL} = 1.8$V

- $V_{IH} = 3.2\text{V}$

2. **Calculate Noise Margin Low (NM_L):** Using the formula $NM_L = V_{IL} - V_{OL}$:

$$NM_L = 1.8\text{V} - 0.1\text{V} = 1.7\text{V}$$

3. **Calculate Noise Margin High (NM_H):** Using the formula $NM_H = V_{OH} - V_{IH}$:

$$NM_H = 4.9\text{V} - 3.2\text{V} = 1.7\text{V}$$

Final Answer: The noise margins are $NM_L = 1.7\text{V}$ and $NM_H = 1.7\text{V}$.

4.2 Delay Estimation in CMOS Circuits

Propagation delay represents the time required for a signal to pass through a logic gate. For rapid computation in VLSI design, the RC delay model is universally applied, treating transistors as equivalent resistors and wire/gate capacitance as equivalent capacitors.

Methodology:

RC Delay Model and Elmore Delay Formula Use the following steps to estimate the propagation delay of a CMOS circuit:

1. **Determine Equivalent Resistance (R_{eq}):** Identify the effective resistance of the conducting transistor network (either the pull-up PMOS network or the pull-down NMOS network).

2. **Determine Load Capacitance (C_{load}):** Sum up all capacitances driven by the output node, including intrinsic drain capacitances and extrinsic load capacitances (wire and next-stage gate capacitances).

3. **Calculate Basic RC Propagation Delay:** For a simple gate switching, use the first-order RC delay formula:

$$t_{pd} \approx 0.69 \times R_{eq} \times C_{load}$$

(Note: Often simplified directly as $t_{pd} \approx R_{eq}C_{load}$ for worst-case delay estimations depending on the specific model used).

4. **Calculate Distributed Delay Using Elmore Delay Model (For RC Trees):** If the circuit forms an RC chain (e.g. series transistors or long interconnects):

$$T_D = \sum_{i=1}^{N} R_i C_i$$

Where R_i is the total resistance from the source to node i, and C_i is the capacitance at node i.

Worked Example:

Estimating Propagation Delay of a CMOS Gate **Problem:** A CMOS logic gate is driving a load capacitance of $C_{load} = 50\text{fF}$. The equivalent resistance of the active pull-down NMOS network is $R_{eq} = 12\text{k}\Omega$. Calculate the high-to-low propagation delay (t_{pHL}) using the $0.69RC$ approximation.
Solution:

1. **List given parameters:**

- Equivalent resistance, $R_{eq} = 12\text{k}\Omega = 12 \times 10^3\Omega$
- Load capacitance, $C_{load} = 50\text{fF} = 50 \times 10^{-15}\text{F}$

2. **Apply the RC delay formula:**

$$t_{pHL} = 0.69 \times R_{eq} \times C_{load}$$

3. **Compute the final value:**

$$t_{pHL} = 0.69 \times (12 \times 10^{3}) \times (50 \times 10^{-15})$$

$$t_{pHL} = 0.69 \times 600 \times 10^{-12}$$

$$t_{pHL} = 414 \times 10^{-12}\text{s}$$

Final Answer: The high-to-low propagation delay is 414ps (picoseconds).

4.3 Power Dissipation in CMOS Logic

Total power dissipation in CMOS circuits primarily consists of dynamic power (due to switching activities and short-circuit currents) and static power (due to leakage currents). Computational analysis focuses on dynamic switching power as it typically dominates.

Methodology:

Calculating Dynamic and Static Power Dissipation To compute the total power consumption of a CMOS block, utilize the following steps:

1. **Calculate Dynamic Switching Power (P_{dyn}):** This is the power consumed to charge and discharge load capacitances.

$$P_{dyn} = \alpha \cdot C_L \cdot V_{DD}^2 \cdot f$$

Where:

- α = Activity factor (probability of switching, e.g., 0.5 for a clock, ≈ 0.1 to 0.3 for data)
- C_L = Total load capacitance
- V_{DD} = Supply voltage
- f = Operating frequency

2. **Calculate Static (Leakage) Power (P_{stat}):** This power is consumed even when the circuit is inactive.

$$P_{stat} = I_{leak} \cdot V_{DD}$$

Where I_{leak} is the total subthreshold and gate leakage current.

3. **Calculate Total Power (P_{total}):** Sum the dynamic and static power components (ignoring short-circuit power unless specifically provided).

$$P_{total} = P_{dyn} + P_{stat}$$

Worked Example:

Computing Total Power Dissipation **Problem:** A CMOS microprocessor module operates at a frequency of 500MHz with a supply voltage of 1.8V. The total load capacitance of the active nodes is 200pF and the average activity factor is $\alpha = 0.2$. The block has a constant leakage current of 50μ A. Calculate the dynamic power, static power, and total power dissipation.
Solution:

1. **Extract parameters:** $f = 500 \times 10^{6}$Hz, $V_{DD} = 1.8$V, $C_L = 200 \times 10^{-12}$F, $\alpha = 0.2$, $I_{leak} = 50 \times 10^{-6}$A.

2. **Calculate Dynamic Power:**

$$P_{dyn} = \alpha \cdot C_L \cdot V_{DD}^2 \cdot f$$
$$P_{dyn} = 0.2 \times (200 \times 10^{-12}) \times (1.8)^2 \times (500 \times 10^6)$$
$$P_{dyn} = 0.2 \times (200 \times 10^{-12}) \times 3.24 \times (500 \times 10^6)$$
$$P_{dyn} = (40 \times 10^{-12}) \times 3.24 \times (500 \times 10^6)$$
$$P_{dyn} = 129.6 \times 10^{-12} \times 500 \times 10^6$$
$$P_{dyn} = 64800 \times 10^{-6}\text{W} = 64.8\text{mW}$$

3. **Calculate Static Power:**

$$P_{stat} = I_{leak} \cdot V_{DD}$$
$$P_{stat} = (50 \times 10^{-6}) \times 1.8 = 90 \times 10^{-6}\text{W} = 0.09\text{mW}$$

4. **Calculate Total Power:**

$$P_{total} = P_{dyn} + P_{stat} = 64.8\text{mW} + 0.09\text{mW} = 64.89\text{mW}$$

Final Answer: The module dissipates 64.8mW of dynamic power, 0.09mW of static power, yielding a total power dissipation of 64.89mW.

4.4 CMOS Logic Gate Sizing

To ensure complex logic gates (like NAND or NOR) exhibit the same delay characteristics as a standard base inverter, transistor sizing is performed. The resistance of a transistor is inversely proportional to its width (W).

Methodology:

Equivalent Inverter Method for Transistor Sizing To size a complex CMOS gate to match the drive strength (resistance) of a reference inverter, follow these steps:

1. **Identify Reference Inverter Sizes:** Determine the width of the NMOS (W_n) and PMOS (W_p) in the base inverter. Usually, due to mobility differences, $W_p \approx 2W_n$.
2. **Analyze the Pull-Down Network (NMOS):**
   - For N NMOS transistors connected in **series** (e.g., in a NAND gate), the equivalent resistance increases by N. To compensate, multiply the base NMOS width by N: $W_{new} = N \times W_n$.
   - For NMOS transistors in **parallel** (e.g., in a NOR gate), the worst-case path is just a single transistor. Keep the width the same as the base inverter: $W_{new} = W_n$.
3. **Analyze the Pull-Up Network (PMOS):**
   - For P PMOS transistors connected in **series** (e.g., in a NOR gate), the equivalent resistance increases by P. Multiply the base PMOS width by P: $W_{new} = P \times W_p$.
   - For PMOS transistors in **parallel** (e.g., in a NAND gate), the worst-case path is just a single transistor. Keep the width the same as the base inverter: $W_{new} = W_p$.

Worked Example:

Sizing a 3 Input NAND Gate **Problem:** Size a 3-input CMOS NAND gate such that its worst-case rise and fall delays match those of a reference inverter. The reference inverter has an NMOS width of $W_n = 2\mu$m and a PMOS width of $W_p = 4\mu$m.
Solution:

1. **Establish base sizes:** Base inverter NMOS $W_n = 2\mu$m. Base inverter PMOS $W_p = 4\mu$m.
2. **Size the Pull-Down (NMOS) Network:** In a 3-input NAND gate, the 3 NMOS transistors are connected in **series**. To maintain the same equivalent resistance as a single NMOS, each must be 3 times wider.
$$W_{NAND\_NMOS} = 3 \times W_n = 3 \times 2\mu\text{m} = 6\mu\text{m}$$
3. **Size the Pull-Up (PMOS) Network:** In a 3-input NAND gate, the 3 PMOS transistors are connected in **parallel**. The worst-case rise delay happens when only one PMOS is turned on. Therefore, the width remains the same as the base inverter PMOS.
$$W_{NAND\_PMOS} = 1 \times W_p = 4\mu\text{m}$$

Final Answer: The 3-input NAND gate requires NMOS transistors sized at 6μm each and PMOS transistors sized at 4μm each.

Worked Example:

Sizing a 2 Input NOR Gate **Problem:** Size a 2-input CMOS NOR gate to match the same reference inverter from the previous example ($W_n = 2\mu$m, $W_p = 4\mu$m).
Solution:

1. **Establish base sizes:** Base inverter NMOS $W_n = 2\mu$m. Base inverter PMOS $W_p = 4\mu$m.
2. **Size the Pull-Down (NMOS) Network:** In a 2-input NOR gate, the 2 NMOS transistors are connected in **parallel**. The worst-case fall delay is governed by a single active NMOS transistor.
$$W_{NOR\_NMOS} = 1 \times W_n = 2\mu\text{m}$$
3. **Size the Pull-Up (PMOS) Network:** In a 2-input NOR gate, the 2 PMOS transistors are connected in **series**. To compensate for the doubled resistance, the width must be doubled.
$$W_{NOR\_PMOS} = 2 \times W_p = 2 \times 4\mu\text{m} = 8\mu\text{m}$$

Final Answer: The 2-input NOR gate requires NMOS transistors sized at 2μm each and PMOS transistors sized at 8μm each.

CHAPTER 5
Verilog Programming

Verilog is a Hardware Description Language (HDL) used to model electronic systems. In this chapter, we focus on computational procedures for writing Verilog code using various abstraction levels: Gate-Level, Dataflow, Behavioral, and Structural modeling, along with Testbench design for simulation.

5.1 Gate-Level Modeling

Gate-level modeling describes a circuit using basic logic gates as primitives. The built-in primitives include `and`, `nand`, `or`, `nor`, `xor`, `xnor`, and `not`.

Methodology:

Gate Level Modeling Procedure

1. **Module Declaration:** Start with the `module` keyword followed by the module name and port list.
2. **Port Definitions:** Declare ports as `input` or `output`.
3. **Internal Connections:** Declare intermediate connections as `wire`.
4. **Gate Instantiation:** Instantiate built-in logic gates. The syntax is: `gatetype inst_name (output, input1, input2, ...);`. Note that the output is always listed first.
5. **Termination:** End the module with the `endmodule` keyword.

Worked Example:

Half Adder Gate Level Implementation **Problem:** Write a Verilog code for a Half Adder using gate-level modeling. The inputs are A and B, and the outputs are Sum and $Carry$.
Solution:

1. **Identify the Logic Equations:** For a Half Adder: $Sum = A \oplus B$
   $Carry = A \cdot B$
2. **Declare the Module and Ports:**

```
module half_adder (
    input A,
    input B,
    output Sum,
    output Carry
);
```

3. **Instantiate the Gates:** Use an `xor` gate for the Sum and an `and` gate for the Carry.

```
    xor g1 (Sum, A, B);
    and g2 (Carry, A, B);
```

4. **Complete the Module:**

```
endmodule
```

5.2 Dataflow Modeling

Dataflow modeling describes how data flows through continuous assignments. It is commonly used for combinational logic.

Methodology:

Dataflow Modeling Procedure

1. **Module Declaration:** Define the module, inputs, and outputs.
2. **Continuous Assignment:** Use the `assign` keyword to drive values onto nets (wires).
3. **Operators:** Construct boolean expressions using Verilog bitwise and logical operators:
   - AND: `&`
   - OR: `|`
   - XOR: `^`
   - NOT: `~`
   - Conditional (Ternary): `condition ? true_expr : false_expr`
4. **Termination:** Close with `endmodule`.

Worked Example:

Full Adder Dataflow Implementation **Problem:** Write a Verilog code for a Full Adder using dataflow modeling. The inputs are A, B, and Cin. The outputs are Sum and $Cout$.
Solution:

1. **Identify the Logic Equations:** $Sum = A \oplus B \oplus Cin$
   $Cout = (A \cdot B) + (B \cdot Cin) + (A \cdot Cin)$
2. **Declare the Module:**

```
module full_adder (
    input A,
    input B,
    input Cin,
    output Sum,
    output Cout
);
```

3. **Apply Continuous Assignments:**

```
    assign Sum = A ^ B ^ Cin;
    assign Cout = (A & B) | (B & Cin) | (A & Cin);
```

4. **Complete the Module:**

```
endmodule
```

5.3 Behavioral Modeling

Behavioral modeling describes the behavior of a circuit at a high level using procedural blocks like `always`. It is used for both combinational and sequential logic.

Methodology:

Behavioral Modeling Procedure

1. **Module Declaration:** Define inputs and outputs.
2. **Register Declaration:** Any output or internal variable assigned inside an `always` block MUST be declared as type `reg`.
3. **Always Block:** Create an `always @(sensitivity_list)` block. For combinational logic, list all input variables or use `always @(*)`.
4. **Procedural Statements:** Use `if-else` or `case` statements to define the logic flow. Enclose multiple statements in `begin` and `end`.
5. **Blocking Assignments:** Use `=` for combinational logic assignments within the block.

Worked Example:

Multiplexer Behavioral Implementation **Problem:** Implement a 4-to-1 Multiplexer using behavioral modeling. The inputs are a 4-bit data bus D and a 2-bit select line S. The output is Y.
Solution:

1. **Declare the Module and Data Types:**

```
module mux4to1 (
    input [3:0] D,
    input [1:0] S,
    output reg Y
);
```

2. **Create the Always Block and Case Statement:** Trigger the block on any change to D or S.

```
    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
```

3. **Complete the Module:**

```
endmodule
```

5.4 Sequential Logic Modeling

Sequential circuits possess memory, meaning their outputs depend on the current state (or previous inputs) and a clock signal.

Methodology:

Sequential Circuit Modeling Procedure

1. **Clock and Reset Definitions:** Include a clock (`clk`) and usually a reset (`rst`) in the input list.
2. **Output Register:** Declare state variables and outputs as `reg`.
3. **Edge-Triggered Always Block:** Use `always @(posedge clk)` or `always @(negedge clk)`.
4. **Asynchronous vs Synchronous Reset:**
   - For asynchronous reset, add the reset edge to the sensitivity list: `always @(posedge clk or posedge rst)`.
   - For synchronous reset, omit it from the sensitivity list and check it inside the block.
5. **Non-Blocking Assignments:** Use `<=` for updating registers in sequential blocks to ensure simultaneous state updates.

Worked Example:

D Flip Flop Implementation **Problem:** Write a Verilog behavioral model for a D Flip-Flop with an active-low asynchronous reset.
Solution:

1. **Declare the Module:**

```
module dff_async_rst (
    input D,
    input clk,
    input rst_n, // Active-low reset
    output reg Q
);
```

2. **Write the Edge-Triggered Block:** Trigger on the positive edge of `clk` and the negative edge of `rst_n`.

```
    always @(posedge clk or negedge rst_n) begin
        if (!rst_n) begin
            Q <= 1'b0;
        end else begin
            Q <= D;
        end
    end
```

3. **Complete the Module:**

```
endmodule
```

Worked Example:

Binary Counter Implementation **Problem:** Write a Verilog code for a 4-bit up counter with a synchronous active-high reset.
Solution:

1. **Declare the Module:**

```
module up_counter (
```

```
    input clk,
    input rst,
    output reg [3:0] count
);
```

2. **Write the Edge-Triggered Block:** Since reset is synchronous, do not include it in the sensitivity list.

```
    always @(posedge clk) begin
        if (rst) begin
            count <= 4'b0000;
        end else begin
            count <= count + 1;
        end
    end
```

3. **Complete the Module:**

```
endmodule
```

5.5 Structural Modeling

Structural modeling involves connecting pre-defined sub-modules together to form a larger system.

Methodology:

Structural Modeling Procedure

1. **Top-Level Definition:** Define the inputs and outputs of the overarching top-level module.
2. **Internal Wires:** Declare `wire` types to interconnect sub-modules.
3. **Instantiation:** Create instances of the lower-level modules. The format is: `module_name instance_name (port_mapping);`.
4. **Port Mapping by Name:** Recommended method to avoid connection errors: `.submodule_port(top_level_signal)`.

Worked Example:

Ripple Carry Adder Structural Implementation **Problem:** Construct a 4-bit Ripple Carry Adder by instantiating four 1-bit Full Adder modules.
Solution:

1. **Assume Full Adder Module Exists:** The sub-module interface is: `module full_adder (input A, input B, input Cin, output Sum, output Cout);`
2. **Declare the Top-Level Module:**

```
module ripple_carry_adder_4bit (
    input [3:0] A,
    input [3:0] B,
    input Cin,
    output [3:0] Sum,
    output Cout
);
```

3. **Declare Internal Wires for Carries:**

```
wire c1, c2, c3;
```

4. **Instantiate the Full Adders:**

```
full_adder FA0 (
    .A(A[0]), .B(B[0]), .Cin(Cin), .Sum(Sum[0]), .Cout(c1)
);

full_adder FA1 (
    .A(A[1]), .B(B[1]), .Cin(c1), .Sum(Sum[1]), .Cout(c2)
);

full_adder FA2 (
    .A(A[2]), .B(B[2]), .Cin(c2), .Sum(Sum[2]), .Cout(c3)
);

full_adder FA3 (
    .A(A[3]), .B(B[3]), .Cin(c3), .Sum(Sum[3]), .Cout(Cout)
);
```

5. **Complete the Module:**

```
endmodule
```

5.6 Writing Testbenches

A testbench is a simulation environment used to apply test vectors to the Design Under Test (DUT) and verify its functionality.

Methodology:

Testbench Creation Procedure

1. **Testbench Module Declaration:** Define a module without any ports.
2. **Signal Declaration:**
   - Declare `reg` types for signals that drive the DUT inputs (stimulus).
   - Declare `wire` types for signals that monitor the DUT outputs.
3. **DUT Instantiation:** Instantiate the DUT and map the testbench signals to the DUT ports.
4. **Stimulus Generation:**
   - Use an `initial` block to apply a sequence of inputs.
   - Use delay operators (e.g., `#10`) to advance simulation time between input changes.
5. **Observation:** Optionally use system tasks like `$monitor` or `$display` to print values to the console.
6. **Finish Simulation:** Conclude the `initial` block with the `$finish` command.

Worked Example:

Combinational Logic Testbench **Problem:** Write a testbench to verify the functionality of a 2-input AND gate module named `and_gate`.
Solution:

1. **Assume DUT Interface:** `module and_gate(input a, input b, output y);`
2. **Declare Testbench Module and Signals:**

```
module tb_and_gate;
    reg tb_a;
    reg tb_b;
    wire tb_y;
```

3. **Instantiate the DUT:**

```
    and_gate DUT (
        .a(tb_a),
        .b(tb_b),
        .y(tb_y)
    );
```

4. **Generate Stimulus inside Initial Block:**

```
    initial begin
        // Setup monitoring
        $monitor("Time = %0d, a = %b, b = %b, y = %b",
                 $time, tb_a, tb_b, tb_y);

        // Apply test vectors with delays
        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;

        // End simulation
        $finish;
    end
```

5. **Complete the Module:**

```
endmodule
```

Appendix: Key Formulae

This appendix provides a quick reference to the key formulae and mathematical relationships discussed in this book.

Unit 1: VLSI Design Metrics and Economics

Yield of Good Dies:

$$Y = e^{-A \cdot D_0}$$

where A is the die area and D_0 is the defect density.

Number of Dies per Wafer:

$$N_{dies} \approx \frac{\pi \cdot r^2}{A_{die}} - \frac{\pi \cdot 2r}{\sqrt{2 \cdot A_{die}}}$$

Number of Good Dies:

$$N_{good} = N_{dies} \times Y$$

Dynamic Power Dissipation:

$$P_{dynamic} = \alpha \cdot C_L \cdot V_{DD}^2 \cdot f$$

where α is the activity factor, C_L is the load capacitance, V_{DD} is the supply voltage, and f is the clock frequency.

Transistor Count Growth:

$$N_t = N_0 \times 2^n$$

Unit 2: MOS Transistor Theory

Gate Oxide Capacitance per Unit Area:

$$C_{ox} = \frac{\epsilon_{ox}}{t_{ox}}$$

Process Transconductance Parameter:

$$k'_n = \mu_n C_{ox}$$

Transistor Transconductance Parameter (NMOS):

$$k_n = k_n' \frac{W}{L} = \mu_n C_{ox} \frac{W}{L}$$

Drain Current (Linear Region):

$$I_D = k_n \left[(V_{GS} - V_T) V_{DS} - \frac{V_{DS}^2}{2} \right]$$

Drain Current (Saturation Region, Ignoring Channel Length Modulation):

$$I_D = \frac{1}{2} k_n' \frac{W}{L} (V_{GS} - V_T)^2$$

Drain Current (Saturation Region, Considering Channel Length Modulation):

$$I_D = \frac{1}{2} k_n (V_{GS} - V_T)^2 (1 + \lambda V_{DS})$$

Aspect Ratio (W/L) from Saturation Current:

$$\frac{W}{L} = \frac{2 I_D}{k_n' (V_{GS} - V_T)^2 (1 + \lambda V_{DS})}$$

Aspect Ratio (W/L) from Linear Current:

$$\frac{W}{L} = \frac{I_D}{k_n' \left[(V_{GS} - V_T) V_{DS} - \frac{V_{DS}^2}{2} \right]}$$

Unit 3: CMOS Inverter

Symmetric CMOS Inverter Condition:

$$\mu_n C_{ox} \left(\frac{W}{L} \right)_n = \mu_p C_{ox} \left(\frac{W}{L} \right)_p$$
$$\left(\frac{W}{L} \right)_p = \frac{\mu_n}{\mu_p} \left(\frac{W}{L} \right)_n$$

Transconductance Ratio:

$$k_R = \frac{k_p}{k_n}$$

Switching Threshold Voltage (V_M):

$$\frac{1}{2} k_n (V_M - V_{tn})^2 = \frac{1}{2} k_p (V_{DD} - V_M - |V_{tp}|)^2$$
$$V_M = \frac{V_{tn} + \sqrt{k_R} (V_{DD} - |V_{tp}|)}{1 + \sqrt{k_R}}$$

Approximate Input Threshold Voltages:

$$V_{IL} \approx \frac{3V_{DD} + 2V_t}{8}$$
$$V_{IH} \approx \frac{5V_{DD} - 2V_t}{8}$$

Noise Margins:

$$NM_L = V_{IL} - V_{OL}$$
$$NM_H = V_{OH} - V_{IH}$$

Propagation Delay:

$$t_{pLH} = 0.69 R_{eq,p} C_L$$
$$t_{pHL} = 0.69 R_{eq,n} C_L$$
$$t_p = \frac{t_{pHL} + t_{pLH}}{2}$$

Ring Oscillator Frequency:

$$T = 2 \times N \times t_p$$
$$f_{osc} = \frac{1}{T} = \frac{1}{2 \times N \times t_p}$$

Power Dissipation:

$$P_{stat} = I_{leakage} \times V_{DD}$$
$$P_{total} = P_{dyn} + P_{stat} + P_{sc}$$

Unit 4: Combinational Logic Design

Elmore Delay:

$$T_D = \sum_{i=1}^{N} R_i C_i$$

Propagation Delay Approximation:

$$t_{pd} \approx 0.69 \times R_{eq} \times C_{load}$$

Equivalent Sizing for N-input NAND Gate (Worst-case):

$$W_{NAND\_NMOS} = N \times W_n$$
$$W_{NAND\_PMOS} = 1 \times W_p$$

Equivalent Sizing for N-input NOR Gate (Worst-case):

$$W_{NOR\_NMOS} = 1 \times W_n$$
$$W_{NOR\_PMOS} = N \times W_p$$

Unit 5: Sequential and Arithmetic Logic Circuits

Half Adder:

$$Sum = A \oplus B$$
$$Carry = A \cdot B$$

Full Adder:

$$Sum = A \oplus B \oplus Cin$$
$$Cout = (A \cdot B) + (B \cdot Cin) + (A \cdot Cin)$$