

VLSI Technology (4353206) - Summer 2025 Solution

Milav Dabgar

May 19, 2025

Question 1(a) [3 marks]

Draw neat labeled diagram of physical structure of n-channel MOSFET.

Solution

Physical Structure of n-channel MOSFET:

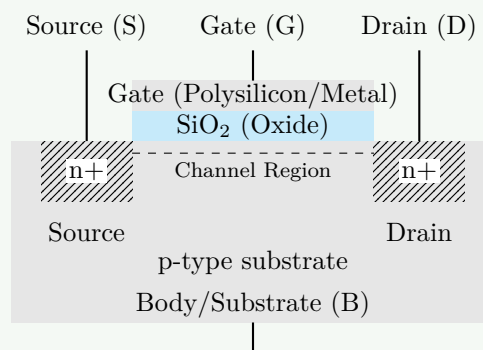


Figure 1. n-channel MOSFET Structure

Key Components:

- **Source:** n+ doped region providing electrons
- **Drain:** n+ doped region collecting electrons
- **Gate:** Metal electrode controlling channel
- **Oxide:** SiO₂ insulating layer
- **Substrate:** p-type silicon body

Mnemonic

“SOGD - Source, Oxide, Gate, Drain”

Question 1(b) [4 marks]

Draw energy band diagram of depletion and inversion of MOS under external bias with MOS biasing diagram. Explain inversion region in detail.

Solution

MOS Biasing Circuit:

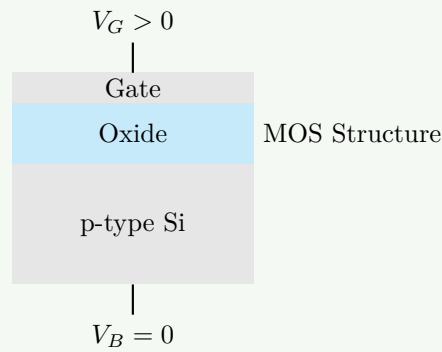
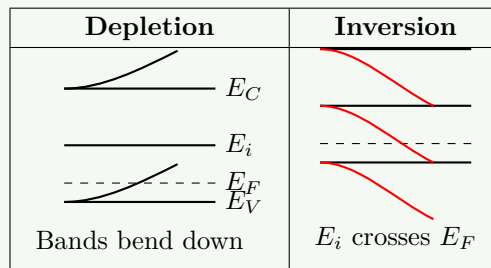


Figure 2. MOS Biasing

Energy Band Diagrams:



Inversion Region Details:

- **Strong inversion:** $V_G > V_T$ (threshold voltage)
- **Electron channel:** Forms at Si-SiO₂ interface as minority carriers (electrons) accumulate.
- **Channel conductivity:** Increases with gate voltage.
- **Threshold condition:** Surface potential $\phi_s = 2\phi_F$.

Mnemonic

“DIVE - Depletion, Inversion, Voltage, Electrons”

Question 1(c) [7 marks]

Explain I-V characteristics of MOSFET.

Solution

I-V Characteristic Regions:

Table 1. Operating Regions

Region	Condition	Drain Current (I_D)
Cutoff	$V_{GS} < V_T$	$I_D \approx 0$
Linear	$V_{GS} > V_T, V_{DS} < V_{GS} - V_T$	$I_D = \mu_n C_{ox} \frac{W}{L} [(V_{GS} - V_T)V_{DS} - \frac{V_{DS}^2}{2}]$
Saturation	$V_{GS} > V_T, V_{DS} \geq V_{GS} - V_T$	$I_D = \frac{1}{2} \mu_n C_{ox} \frac{W}{L} (V_{GS} - V_T)^2$

Characteristic Curve:

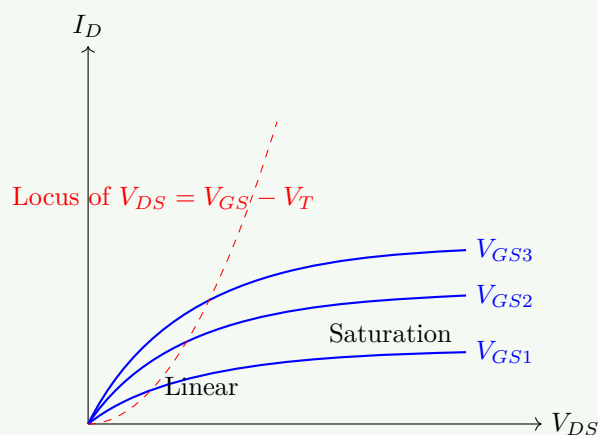


Figure 3. MOSFET I-V Characteristics

Key Parameters:

- μ_n : Electron mobility
- C_{ox} : Gate oxide capacitance per unit area
- W/L : Channel Width to Length ratio
- V_T : Threshold voltage

Mnemonic

“CLS - Cutoff, Linear, Saturation”

Question 1(c) OR [7 marks]

Define scaling. Explain the need of scaling. List and explain the negative effects of scaling.

Solution

Definition: Scaling is the systematic reduction of MOSFET dimensions to improve performance and density.

Need for Scaling:

- **Higher Density:** More transistors per chip area, following Moore's Law.
- **Faster Speed:** Reduced channel length reduces carrier transit time ($t = L^2/\mu V$).
- **Lower Power:** Smaller parasitic capacitances decrease switching energy.
- **Cost Reduction:** More chips per wafer reduces cost per die.

Scaling Types:

Table 2. Scaling Strategies

Type	Gate Length	Supply Voltage	Oxide Thickness
Constant Voltage	$\downarrow \alpha$	Constant	$\downarrow \alpha$
Constant Field	$\downarrow \alpha$	$\downarrow \alpha$	$\downarrow \alpha$

Negative Effects:

- **Short Channel Effects (SCE):** V_T roll-off and drain-induced barrier lowering (DIBL).
- **Hot Carrier Effects:** High electric fields near drain inject carriers into oxide, degrading device.
- **Gate Leakage:** Thin oxide leads to quantum tunneling current.
- **Process Variations:** Difficulty in controlling dimensions at nano-scale.
- **Power Density:** Increased heat per unit area creates thermal management issues.

Mnemonic

“SHGPP - Short channel, Hot carrier, Gate leakage, Process, Power”

Question 2(a) [3 marks]

Implement $Y' = (AB' + A'B)$ using CMOS.

Solution

Logic Analysis: $Y' = AB' + A'B = A \oplus B$ (XOR function).

CMOS Implementation:

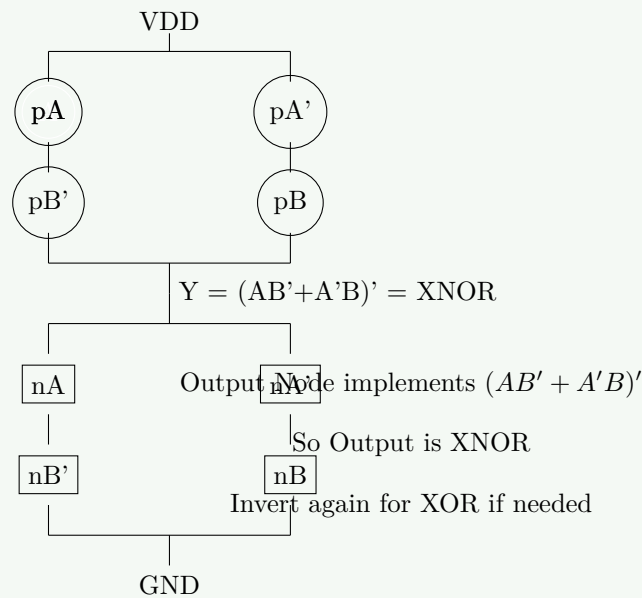


Figure 4. Static CMOS Implementation for XNOR (inverted XOR)

Note: The circuit above implements $(AB' + A'B)'$. To get $Y' = AB' + A'B$, we technically need an inverter at the output, or this represents the structure where pull-down matches the expression.

Mnemonic

“XOR needs complementary switching”

Question 2(b) [4 marks]

Explain enhancement load inverter with its circuit diagrams.

Solution

Circuit Diagram:

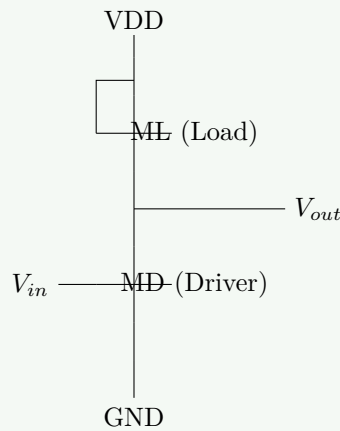


Figure 5. Saturated Enhancement Load Inverter

Configuration:

- **Load (ML):** Enhancement NMOS with Gate connected to Drain ($V_{GS} = V_{DS}$). Always in Saturation or Cutoff.
- **Driver (MD):** Enhancement NMOS with Gate as input.

Operation:

- **High Output (V_{OH}):** Limited to $V_{DD} - V_T$ because load turns off when V_{out} rises to that level.
- **Low Output (V_{OL}):** Close to 0V (depends on ratio K_D/K_L).
- **Disadvantage:** Degraded logic high ($V_{DD} - V_T$) and continuous power dissipation when V_{in} is high.

Mnemonic

“ELI - Enhancement Load Inverter has threshold Issues”

Question 2(c) [7 marks]

Explain Voltage Transfer Characteristic of inverter.

Solution

VTC Curve:

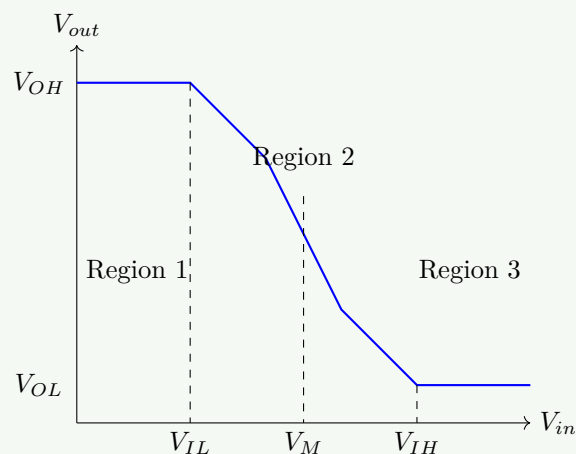


Figure 6. Ideally Inverter VTC

Key Parameters:

Parameter	Description	Ideal Value
V_{OH}	Output High Voltage	V_{DD}
V_{OL}	Output Low Voltage	$0V$
V_{IH}	Input High Voltage	$\approx V_{DD}/2$
V_{IL}	Input Low Voltage	$\approx V_{DD}/2$
V_M	Switching Threshold	$V_{DD}/2$

Noise Margins:

- $NM_H = V_{OH} - V_{IH}$ (High noise margin)
- $NM_L = V_{IL} - V_{OL}$ (Low noise margin)

Regions:

- **Region 1:** V_{in} low, V_{out} high (PMOS Linear, NMOS Cutoff)
- **Region 2:** Transition region (Both Saturation)
- **Region 3:** V_{in} high, V_{out} low (PMOS Saturation, NMOS Linear)

Mnemonic

“VTC shows VOICE - V_{OH} , V_{OL} , Input thresholds, Characteristics, Everything”

Question 2(a) OR [3 marks]

Explain NAND2 gate using CMOS.

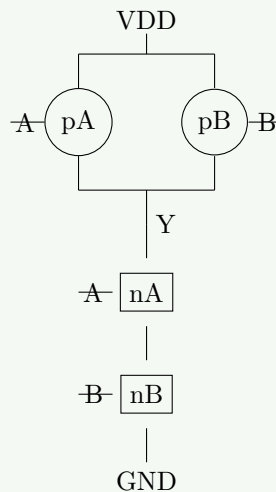
Solution**CMOS NAND2 Circuit:**

Figure 7. CMOS NAND2 Gate

Operation:

- **Pull-Up Network:** PMOS inputs A and B in parallel. If $A=0$ OR $B=0$, output pulled to V_{DD} .
- **Pull-Down Network:** NMOS inputs A and B in series. Only if $A=1$ AND $B=1$, output pulled to GND .

Truth Table:

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Mnemonic

“NAND - Not AND, Parallel PMOS, Series NMOS”

Question 2(b) OR [4 marks]

Explain operating mode and VTC of Resistive load inverter circuit.

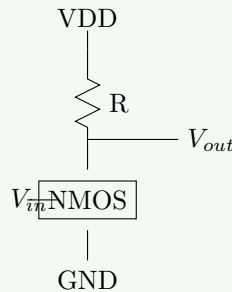
Solution**Circuit Configuration:**

Figure 8. Resistive Load Inverter

Operating Modes:

- $V_{in} = 0$ (**Low**): NMOS OFF. $V_{out} = V_{DD}$.
- $V_{in} = V_{DD}$ (**High**): NMOS ON (Linear). $V_{out} = V_{VOL}$.
- V_{OL} **Calculation**: $V_{OL} = \frac{R_{ON}}{R_{ON} + R} V_{DD}$.

Design Trade-offs:

- **Large R**: Better V_{OL} , lower power, but slower switching (high RC delay).
- **Small R**: Faster switching, but poor V_{OL} and high power consumption.
- **Area**: Resistors consume significant silicon area compared to transistors.

Mnemonic

“RLI - Resistive Load has Inevitable power consumption”

Question 2(c) OR [7 marks]

Draw CMOS inverter and explain its operation with VTC.

Solution**CMOS Inverter Circuit:**

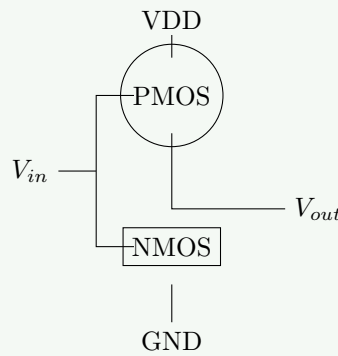


Figure 9. CMOS Inverter

Operation Regions:

V_{in} Range	PMOS	NMOS	V_{out}	Region
0 to V_{TN}	ON	OFF	V_{DD}	1
V_{TN} to $V_{DD} - V_{TP} $	ON	ON	Transition	2
$V_{DD} - V_{TP} $ to V_{DD}	OFF	ON	0V	3

VTC Analysis: The VTC is nearly ideal square-like.

- **Region 1:** Output pulled hard to V_{DD} .
- **Region 2:** Both transistors in saturation (shortly) causing steep drop. Gain is very high.
- **Region 3:** Output pulled hard to GND.

Key Features:

- **Zero Static Power:** In steady states (high/low), one transistor is always OFF.
- **Rail-to-Rail Swing:** Logic levels are exact V_{DD} and GND.
- **Symmetric Design:** If $\beta_n = \beta_p$, switching threshold $V_M = V_{DD}/2$.

Mnemonic

“CMOS has Zero Static Power with Full Swing”

Question 3(a) [3 marks]

Realize $Y = (\bar{A} + \bar{B})\bar{C} + \bar{D} + \bar{E}$ using depletion load.

Solution

Logic Simplification: $Y = (\bar{A} + \bar{B})\bar{C} + \bar{D} + \bar{E}$ Wait, let's check exact expression: $Y = (\bar{A} + \bar{B})\bar{C} + \bar{D} + \bar{E}$. This looks like sum of products. Depletion load implies NMOS logic. Y is the output of proper logic. Usually we implement the complement in Pull Down. Complement of Y? If Y is the target, and we use NMOS with Depletion load (which is basically an inverter structure for the logic block), the Pull Down network should minimize \bar{Y} . $\bar{Y} = \overline{(\bar{A} + \bar{B})\bar{C} + \bar{D} + \bar{E}} = (\bar{A} + \bar{B})\bar{C} \cdot \bar{D} \cdot \bar{E}$. $\bar{Y} = (\bar{A} + \bar{B})\bar{C} \cdot D \cdot E = (\bar{A} + \bar{B} + \bar{C}) \cdot D \cdot E = (AB + C)DE = ABDE + CDE$.

This seems complex. Let's assume the question implies implementing the function directly as written using "Depletion Load NMOS Logic". Standard NMOS logic implements NOR/NAND inherently. Output $F = \overline{PDN}$. So if we want Output Y, we need PDN function equal to \bar{Y} . Let's re-read: "Realize Y=..". Usually implies: Design a gate where Output is Y. So PDN must implement \bar{Y} . If $\bar{Y} = ABDE + CDE = (AB + C)DE$. PDN: D and E in series with (C parallel (A series B)).

Depletion Load Implementation:

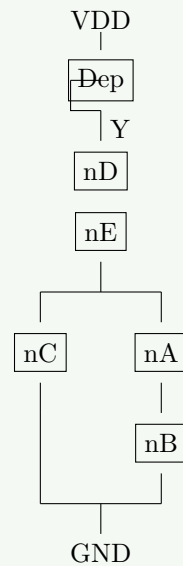


Figure 10. Depletion Load Logic for Y

Mnemonic

“Depletion Load with Parallel/Series pull-down Paths”

Question 3(b) [4 marks]

Write a short note on FPGA.

Solution

FPGA Definition: Field Programmable Gate Array - A reconfigurable integrated circuit that can be programmed by the customer after manufacturing.

Architecture Components:

- **CLB (Configurable Logic Block):** Basic logic unit (LUTs, Flip-flops).
- **IOB (Input/Output Block):** Interface with external pins.
- **Interconnects:** Programmable routing channels (Vertical/Horizontal).
- **Switch Matrix:** Programmable connections between routing tracks.

Programming Technologies:

- **SRAM-based:** Volatile, fast reconfiguration, high density.
- **Antifuse:** Non-volatile, one-time programmable, secure.
- **Flash-based:** Non-volatile, reprogrammable.

Advantages vs ASIC:

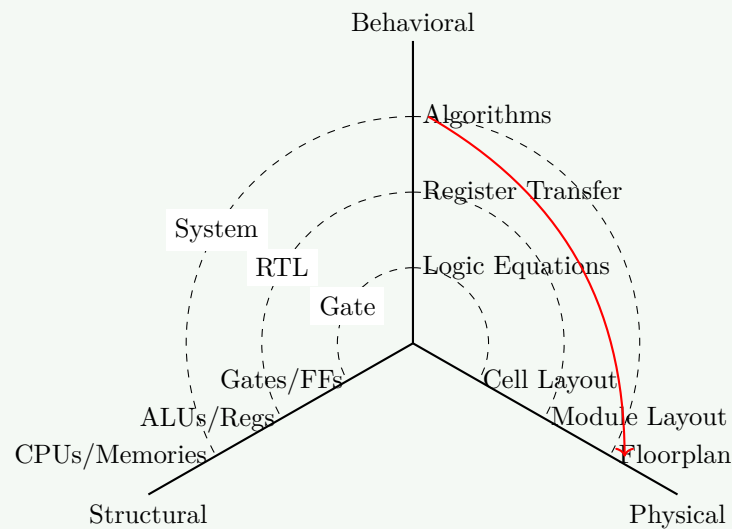
- **Flexibility:** Can be updated/fixed post-deployment.
- **Time-to-market:** No physical fabrication wait time.
- **Cost:** Low NRE (Non-Recurring Engineering) cost, good for low volume.

Mnemonic

“FPGA - Flexible Programming Gives Advantages”

Question 3(c) [7 marks]

Draw and explain Y chart design flow.

Solution**Y-Chart Diagram:****Figure 11.** Gajski-Kuhn Y-Chart**Design Domains:**

- **Behavioral:** Describes *what* simple system does (Algorithms, Equations).
- **Structural:** Describes *how* components are connected (Netlists, Gates).
- **Physical:** Describes *where* components are placed (Layout, Geometry).

Abstraction Levels:

- **System Level:** High level architectural blocks.
- **RTL:** Data flow between registers.
- **Gate Level:** Boolean logic implementation.
- **Transistor/Layout Level:** Physical device details.

Mnemonic

“Y-Chart: Behavioral, Structural, Physical - BSP domains”

Question 3(a) OR [3 marks]

Explain NOR2 gate using depletion load.

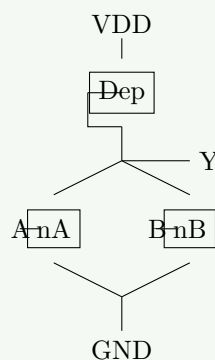
Solution**Depletion Load NOR2 Circuit:**

Figure 12. Depletion Load NOR2**Operation:**

- **Both inputs Low (0,0):** Both pull-down NMOS OFF. Output pulled High to VDD by Depletion load.
- **Any input High:** Corresponding Pull-Down NMOS is ON. Output pulled Low to GND (ratioed logic).

Mnemonic

“NOR with Depletion - Parallel NMOS pull-down”

Question 3(b) OR [4 marks]

Compare full custom and semi-custom design styles.

Solution**Comparison Table:**

Parameter	Full Custom	Semi-Custom
Design Time	Long (6-18 months)	Short (2-6 months)
Performance	Optimal	Good
Area	Minimum	Moderate
Power	Optimized	Acceptable
Cost	High NRE	Lower NRE
Flexibility	Maximum	Limited
Risk	High	Lower

Full Custom Characteristics:

- **Every transistor:** Manually designed and placed.
- **Layout optimization:** Maximum density achieved.
- **Applications:** High-volume, performance-critical (e.g., Microprocessors).

Semi-Custom Types:

- **Gate Array:** Pre-defined transistor array.
- **Standard Cell:** Library of pre-designed cells.
- **FPGA:** Field programmable logic.

Mnemonic

“Full Custom - Maximum control, Semi-Custom - Speed compromise”

Question 3(c) OR [7 marks]

Draw and explain ASIC design flow in detail.

Solution**ASIC Design Flow:**

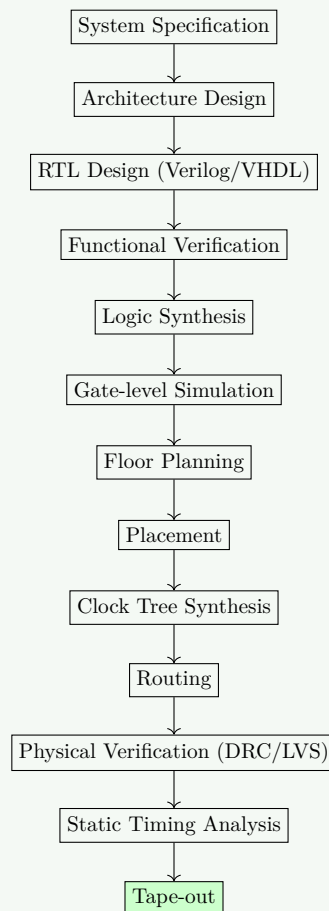


Figure 13. ASIC Design Flow

Design Stages:

- **RTL Design:** Describing hardware behavior using HDL.
- **Synthesis:** Converting RTL to gate-level netlist.
- **Physical Design:** Floorplanning, Placement, and Routing (P&R).
- **Verification:** ensuring functionality (Functional) and manufacturability (Physical/Timing).

Mnemonic

“ASIC flow: RTL to GDSII”

Question 4(a) [3 marks]

Implement the logic function $G = \overline{A(D + E)} + \overline{BC}$ using CMOS

Solution

Logic Analysis: $G = \overline{A(D + E)} + \overline{BC}$. PDN implements $A(D + E) + BC$.

- $D + E \rightarrow D, E$ parallel.
- $A(D + E) \rightarrow A$ in series with (D parallel E).
- $BC \rightarrow B, C$ in series.
- Final OR \rightarrow The two branches in parallel.

PUN implements dual:

- Dual of $(D + E)$ is $D \cdot E$ (series).
- Dual of $A(\dots)$ is $A + \dots$ (parallel).

- Dual of BC is $B + C$ (parallel).
- Dual of major OR is AND (series).
- So: $(A \text{ parallel } (D \text{ series } E)) \text{ series } (B \text{ parallel } C)$.

CMOS Implementation:

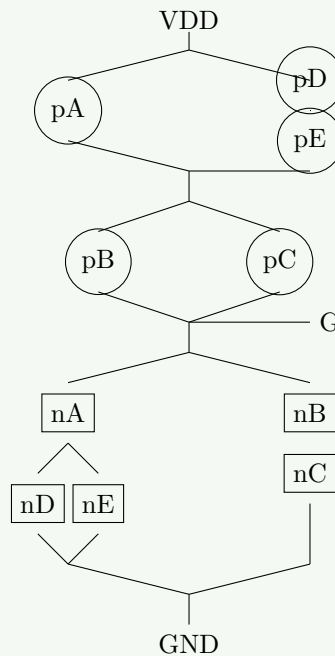


Figure 14. Complex CMOS Gate

Mnemonic

“Complex CMOS - PMOS series, NMOS parallel”

Question 4(b) [4 marks]

Write a Verilog code for 3 bit parity checker.

Solution

```

1  module parity_checker_3bit(
2      input [2:0] data_in,
3      output parity_even,
4      output parity_odd
5  );
6
7      // Even parity checker
8      assign parity_even = ^data_in;
9
10     // Odd parity checker
11     assign parity_odd = ~(^data_in);
12
13     endmodule

```

Truth Table:

Input [2:0]	# of 1s	Even Parity	Odd Parity
000	0	0	1
001	1	1	0
111	3	1	0

Question 4(c) [7 marks]

Implement: 1) $G = (AD + BC + EF)$ using CMOS [3 marks] 2) $Y' = (ABCD + EF(G + H) + J)$ using CMOS [4 marks]

Solution

Part 1: $G = (AD + BC + EF)$ [3 marks]

CMOS Circuit Analysis: $G = (AD + BC + EF)$. This is a non-inverting function. Using strict static CMOS, we implement $\bar{G} = \overline{AD + BC + EF}$ and add an inverter. PDN for \bar{G} matches the function directly (Parallel branches of Series pairs).

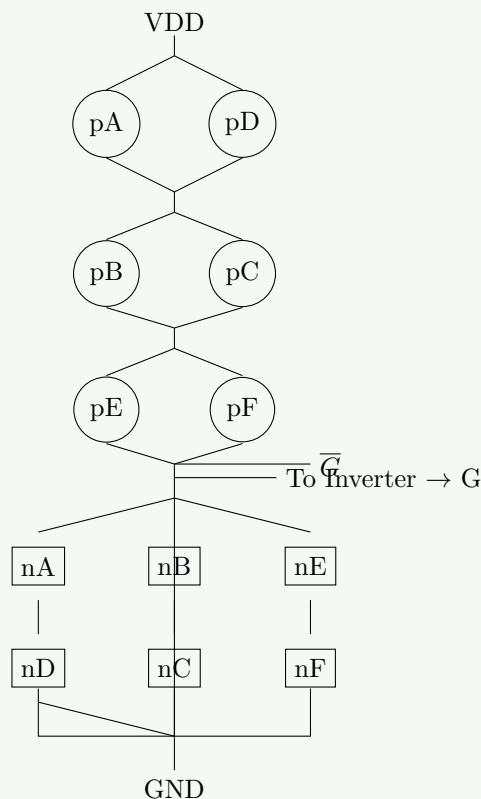


Figure 15. CMOS Logic for G (Output \bar{G})

Part 2: $Y' = (ABCD + EF(G + H) + J)$ [4 marks] This requires a complex single stage. PDN Implementation:

- $ABCD$: 4 NMOS in series.
- $EF(G + H)$: G parallel H , series with E and F .
- J : Parallel to everything.
- All combined in parallel structure.

Question 4(a) OR [3 marks]

Explain AOI logic with example.

Solution

AOI Definition: **AND-OR-Invert** logic implements functions of form: $Y = \overline{(AB + CD + \dots)}$

Example: $Y = \overline{(AB + CD)}$

AOI Implementation:

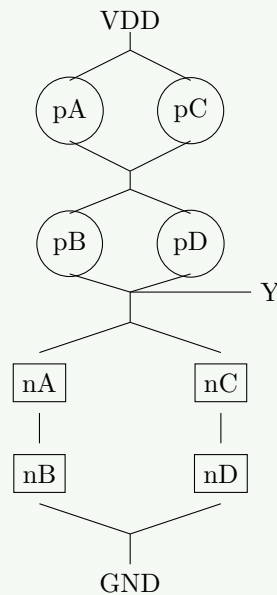


Figure 16. AOI Gate Implementation

Advantages:

- **Single stage:** Direct implementation.
- **Fast:** No propagation through multiple levels.
- **Area efficient:** Fewer transistors than separate gates.

Mnemonic

“AOI - AND-OR-Invert in one stage”

Question 4(b) OR [4 marks]

Write Verilog Code for 4-bit Serial IN Parallel out shift register.

Solution

```

1  module sipo_4bit(
2      input clk,
3      input reset,
4      input serial_in,
5      output reg [3:0] parallel_out
6  );
7
8  always @(posedge clk or posedge reset) begin
9      if (reset) begin
10         parallel_out <= 4'b0000;
11     end else begin
12         // Shift left and insert new bit at LSB
13         parallel_out <= {parallel_out[2:0], serial_in};
14     end
15 end

```

```

15 end
16
17 endmodule

```

Question 4(c) OR [7 marks]

Implement clocked NOR2 SR latch and D-latch using CMOS.

Solution

Clocked NOR2 SR Latch:

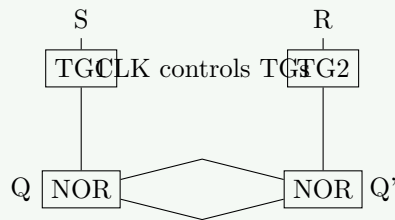


Figure 17. Clocked SR Latch Concept

CMOS D-Latch Circuit:

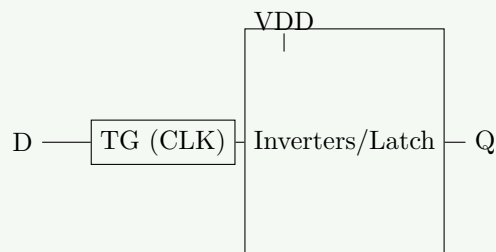


Figure 18. D-Latch

Question 5(a) [3 marks]

Draw the stick diagram for $Y = (PQ + U)'$ using CMOS considering Euler path approach.

Solution

Logic Analysis: $Y = \overline{PQ + U}$. PDN: $P \cdot Q + U$ (P series Q, parallel U). PUN: $\overline{PQ} \cdot \overline{U} = (\bar{P} + \bar{Q}) \cdot \bar{U}$ (P parallel Q, series U).

Euler Path: Common sequence: U - P - Q.

Stick Diagram:

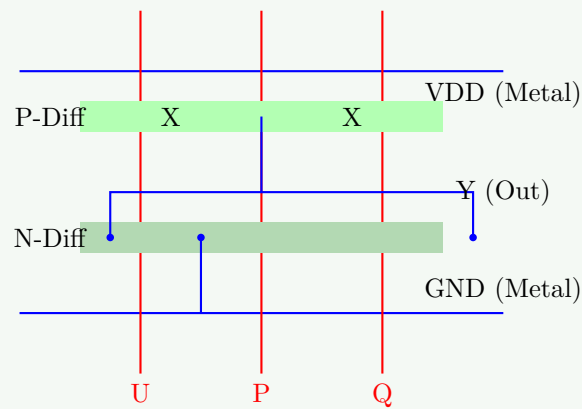


Figure 19. Stick Diagram for Y

Mnemonic

“Stick diagram shows physical layout with Euler path optimization”

Question 5(b) [4 marks]

Implement 8x1 multiplexer using Verilog.

Solution

```

1  module mux_8x1(
2      input  [7:0] data_in,
3      input  [2:0] select,
4      output reg data_out
5  );
6
7  always @(*) begin
8      case (select)
9          3'b000: data_out = data_in[0];
10         3'b001: data_out = data_in[1];
11         3'b010: data_out = data_in[2];
12         3'b011: data_out = data_in[3];
13         3'b100: data_out = data_in[4];
14         3'b101: data_out = data_in[5];
15         3'b110: data_out = data_in[6];
16         3'b111: data_out = data_in[7];
17         default: data_out = 1'b0;
18     endcase
19 end
20
21 endmodule

```

Question 5(c) [7 marks]

Implement full adder using behavioral modeling style in Verilog.

Solution

```

1  module full_adder_behavioral(
2      input A,
3      input B,
4      input Cin,
5      output reg Sum,
6      output reg Cout
7  );
8
9  always @(*) begin
10     case ({A, B, Cin})
11         3'b000: begin Sum = 0; Cout = 0; end
12         3'b001: begin Sum = 1; Cout = 0; end
13         3'b010: begin Sum = 1; Cout = 0; end
14         3'b011: begin Sum = 0; Cout = 1; end
15         3'b100: begin Sum = 1; Cout = 0; end
16         3'b101: begin Sum = 0; Cout = 1; end
17         3'b110: begin Sum = 0; Cout = 1; end
18         3'b111: begin Sum = 1; Cout = 1; end
19         default: begin Sum = 0; Cout = 0; end
20     endcase
21 end
22
23 endmodule

```

Question 5(a) OR [3 marks]

Implement NOR2 gate CMOS circuit with its stick diagram.

Solution

CMOS NOR2 Circuit:

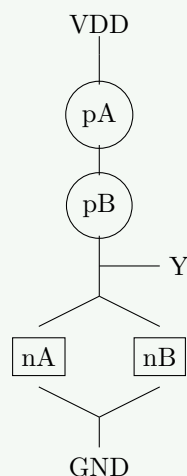


Figure 20. CMOS NOR2 Circuit

Stick Diagram:

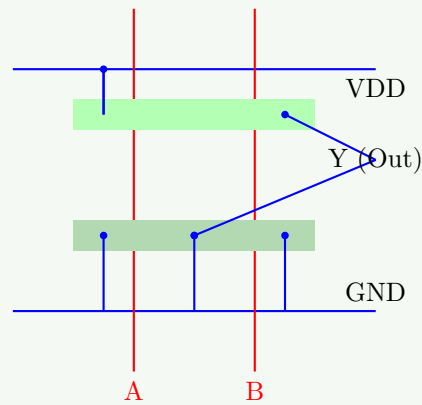


Figure 21. NOR2 Stick Diagram

Question 5(b) OR [4 marks]

Implement 4 bit up counter using Verilog

Solution

```

1 module counter_4bit_up(
2     input clk,
3     input reset,
4     input enable,
5     output reg [3:0] count
6 );
7
8 always @(posedge clk or posedge reset) begin
9     if (reset) begin
10         count <= 4'b0000;
11     end else if (enable) begin
12         if (count == 4'b1111) begin
13             count <= 4'b0000;
14         end else begin
15             count <= count + 1;
16         end
17     end
18 end
19
20 endmodule

```

Question 5(c) OR [7 marks]

Implement 3:8 decoder using behavioral modeling style in Verilog.

Solution

```

1 module decoder_3x8_behavioral(
2     input [2:0] address,
3     input enable,
4     output reg [7:0] decode_out
5 );

```

```
6
7 always @(*) begin
8     if (enable) begin
9         case (address)
10            3'b000: decode_out = 8'b00000001;
11            3'b001: decode_out = 8'b00000010;
12            3'b010: decode_out = 8'b00000100;
13            3'b011: decode_out = 8'b00001000;
14            3'b100: decode_out = 8'b00010000;
15            3'b101: decode_out = 8'b00100000;
16            3'b110: decode_out = 8'b01000000;
17            3'b111: decode_out = 8'b10000000;
18            default: decode_out = 8'b00000000;
19        endcase
20    end else begin
21        decode_out = 8'b00000000;
22    end
23 end
24
25 endmodule
```