VLSI IA3

M4Q1) Explain body effect in multiple input NAND Gate.

The **body effect** (also known as the **substrate bias effect**) is a phenomenon in MOS transistors where the **threshold voltage (V_T) increases** when there is a **voltage difference between the source and the substrate**.

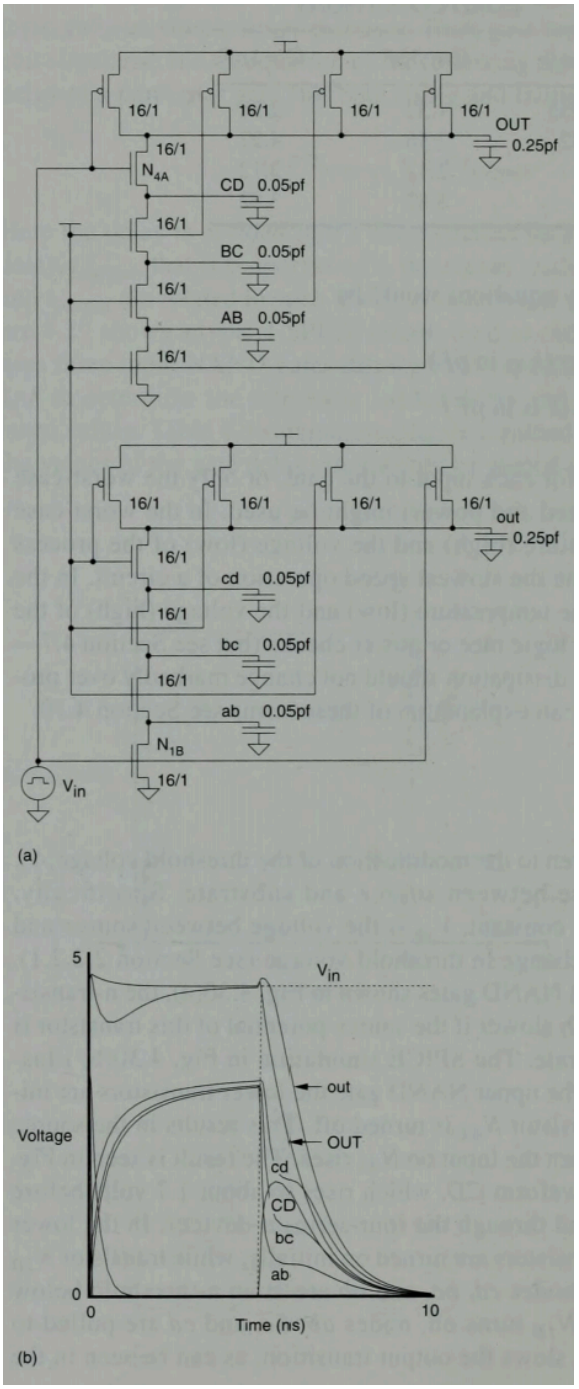
This occurs because the **depletion region widens** when the source potential rises above the body (substrate) potential, requiring a **higher gate voltage** to turn the transistor on.

Body Effect in NAND Gates:

In a **multiple-input CMOS NAND gate**, multiple **nMOS transistors are connected in series**. The key body effect occurs in the **upper transistors** of this nMOS pull-down chain.

Why?

- Only the **bottom nMOS** transistor has its **source directly connected to ground**, so its V_{SB} **= 0V**, meaning **no body effect**.
- For **upper nMOS transistors**, their sources sit **above ground** due to the voltage drop across lower transistors. So V_{SB} **> 0**,



SPICE circuit for observing the result of body effect on gate delay

causing an **increase in** V_T due to body effect.

Impact of Body Effect:

- **Slower Switching:** Increased threshold voltage reduces drive current, which **slows down the output transitions**.
- **Delay Accumulation:** This delay accumulates across the series chain, especially significant in gates with **3 or more inputs**.
- **Voltage Drop:** Internal nodes in the pull-down path discharge more slowly, resulting in **delayed logic LOW** at the output.

Nodes *cd*, *bc*, and *ab* are at an n-threshold below Vdd (~3.1 volts). When N_{1B} turns on, nodes *ab*, *bc*, and *cd* are pulled to ground in that order.

Design Strategies to Mitigate:

1. **Place transistors with the latest arriving signals nearest to the output**—reduces switching delay caused by body effect.
2. **Minimize internal node capacitances** using optimized layout and metal routing instead of diffusion wires.
3. **Keep series stack length small** (2–4) in critical paths to reduce cumulative delay.

M4Q2) Write a short note on I/O pads.

I/O (Input/Output) pads form the interface between the external environment and the internal circuitry of a CMOS chip. They are essential components in VLSI design because they ensure proper signal transmission and protect internal logic from external disturbances.

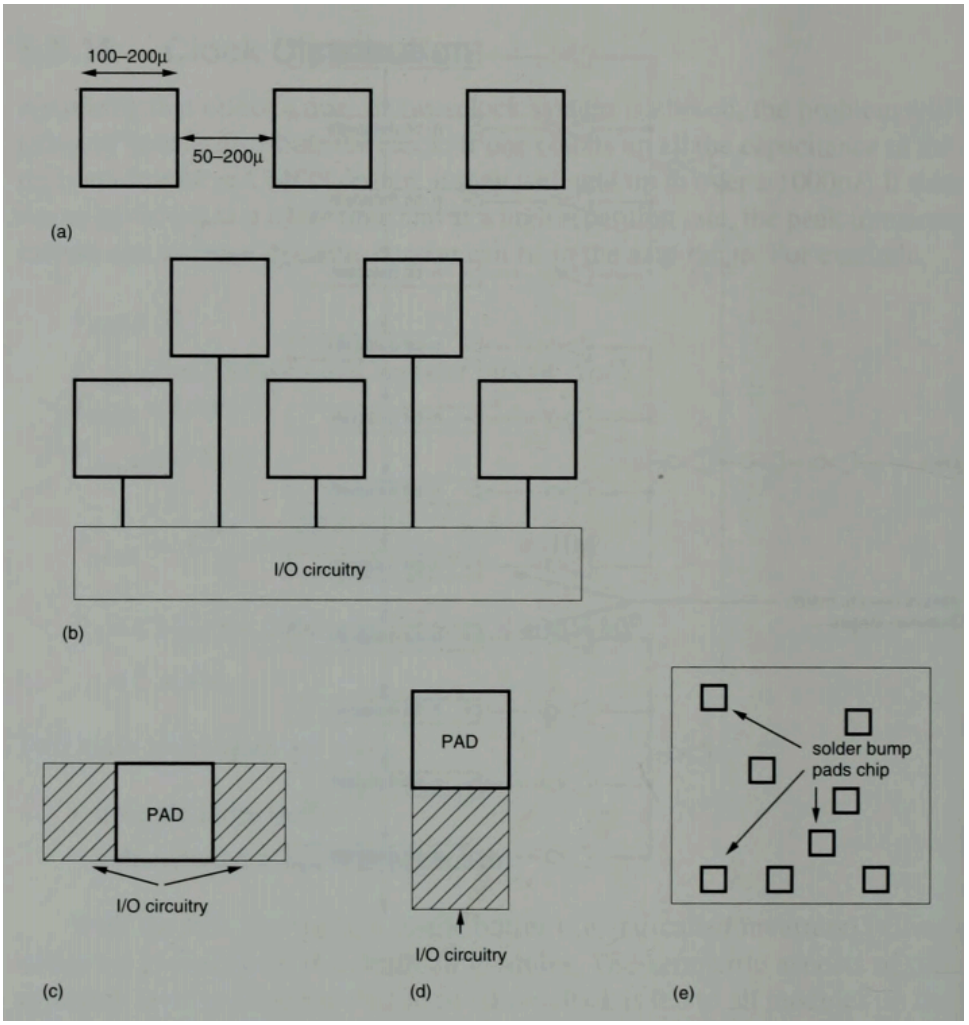
Key Features of I/O Pads

1. **Pad Size and Spacing**:
    - Determined by wire bonding constraints.
    - Typically sized around 100–150 µm² with 150–200 µm pitch between them.

    - High pad counts use **interdigitated** or **pad-limited** designs (see *Fig. 5.83*).
2. **Types of Pads**:
    - **Input Pads**: Receive external signals; require protection against overvoltage (e.g., via clamping diodes).
    - **Output Pads**: Drive external loads and must provide high drive strength.
    - **Tristate/Bidirectional Pads**: Can act as input or output depending on control signals (see *Fig. 5.89*).
3. **Design Considerations**:
    - **Drive Strength**: Must be sufficient for expected load capacitance (important in output pads).
    - **Buffering**: Multi-stage inverters are used to drive large capacitive loads.

- **Latch-up Protection**: Use of guard rings and proper separation of n- and p-transistors is crucial.
- **Noise Reduction**: Pads use "dirty" VDD and VSS to isolate high switching noise from internal logic.

4. **Advanced Features**:
    - **ESD Protection**: Most I/O pads include ESD protection circuits like resistors and diode clamps.
    - **Controlled Slew Rate**: Reduces RF interference (RFI).
    - **On-pad Latches/Registers**: Improve timing performance by reducing delay between pad and internal logic.
    - **Programmability**: Some pads (e.g., in Actel/QuickLogic) are reconfigurable using antifuses.



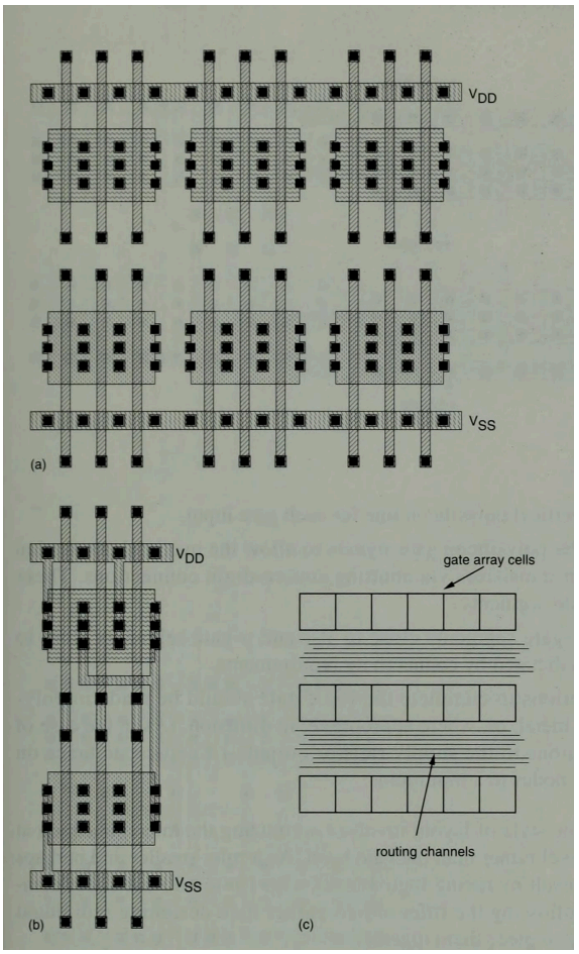
I/O pad options: (a) pad spacing; (b) interdigitated pads; (c) core limited pads; (d) pad limited pads; (e) solder bump I/O

M5Q1) Write a short note on Gate Array Design.

Gate Array Design is a **semi-custom** VLSI design technique that allows fast prototyping and moderate chip customization. It uses **pre-fabricated silicon wafers** with arrays of uncommitted transistors. The final logic function is defined during the last steps of manufacturing by adding **custom metal interconnections**.

Structure and Working:

- **Base Wafer:** Consists of fixed patterns of diffusion and polysilicon layers. Only **metal layers and contacts** are customized for specific logic functions.
- The layout includes **regular rows** of transistor pairs, with routing channels between them for interconnections.
- Only **2–5 masks** are needed during customization, compared to full custom design which needs masks for all layers.



Gate array layout: (a) unprogrammed base array; (b) personalized cell; (c) routing strategy

Advantages:

1. **Short Turnaround Time:** Faster than full-custom; suitable for low-to-medium volume production.
2. **Lower Cost:** Shared base wafers reduce manufacturing and mask costs.
3. **Simplified Testing:** Common test procedures can be reused across designs.
4. **Automation-Friendly:** Layout and routing can be done using automated CAD tools.

Comparison with Sea-of-Gates (SOG):

Feature	Gate Array (GA)	Sea-of-Gates (SOG)
Routing	Uses fixed routing channels	Routing occurs over unused transistors
Flexibility	Moderate	Higher (better utilization)
Metal layers	Limited	Can use multiple

M5Q2) Write a short note on Floor Planning & Layout

What is Floorplanning?

Floorplanning is the process of organizing **functional blocks** of a chip **spatially** before final layout and routing. It's a key step in **physical design** and determines the **performance, area, and routability** of the chip.

- It involves placing:
  - **Major blocks** (e.g., datapaths, memory, I/O pads)
  - **Routing channels**
  - **Power/ground rails**

Step	Description
Partitioning	Break down the chip into functional blocks.
Block Placement	Place high-communication blocks closer.
Aspect Ratio Adjustment	Match the chip's desired shape (usually square/rectangular).
I/O Pad Placement	Distribute pads around the periphery.
Power Planning	Add VDD/VSS rails to each block.
Routing Channels	Allocate space between blocks for metal routing.

Objectives of Floorplanning:

- **Minimize interconnect delay**
- **Reduce chip area**
- **Avoid routing congestion**
- Improve **modularity** and **locality**
- Enhance **power distribution** and **thermal performance**

Layout Design:

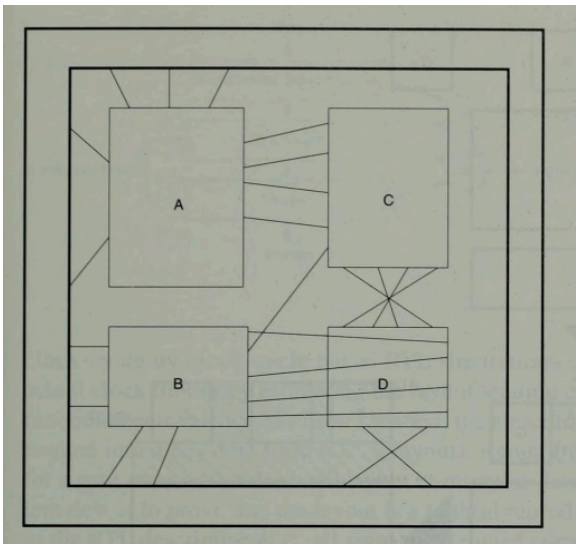
- **Definition:** Layout design is the detailed geometric representation of the circuit components and interconnections using various layers (diffusion,

polysilicon, metal).

- It transforms the **schematic** into a **mask-level drawing** used for fabrication.

Steps in Layout:

1. **Transistor placement**
2. **Interconnect routing** (metal layers)
3. **Design rule check (DRC)**
4. **Layout versus Schematic (LVS) verification**

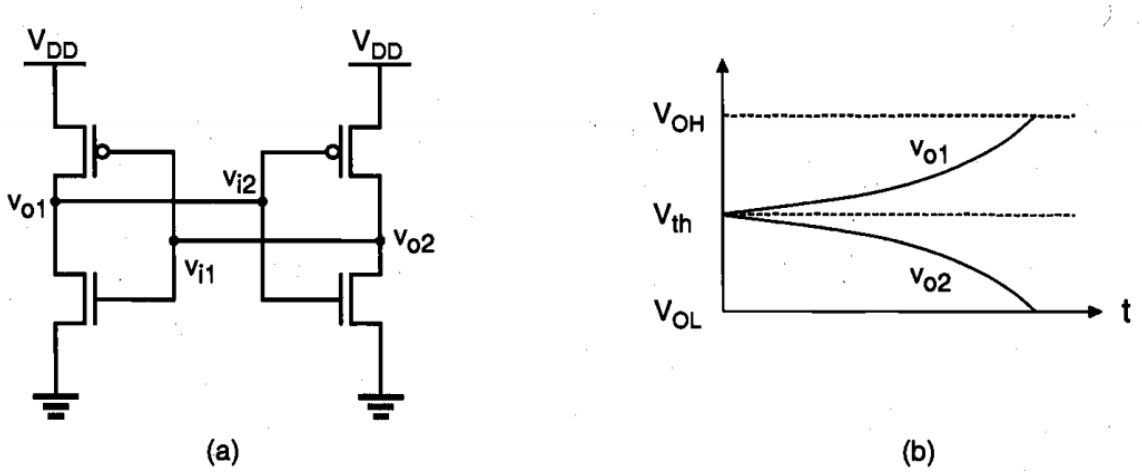


Floorplan example

This shows that module D should be flipped around the 7 axis to improve the routing. Some editors provide shaded color displays of routing density that allows designers to re-place and "rip-up-and-reroute" congested areas of the chip.

M5Q3) What is CMOS 2-Inverter bistable element.

A **CMOS 2-Inverter Bistable Element** is a simple sequential logic circuit that forms the **basic building block of memory elements** such as latches and flip-flops. It consists of **two CMOS inverters connected in a feedback loop**, creating a system with two stable operating points (bistability).



(a) Circuit diagram of a CMOS bistable element. (b) One possibility for the expected time-domain behavior of the output voltages, if the circuit is initially set at its unstable
operating point.

Structure:

- Two **CMOS inverters** are cross-coupled:
  - Output of inverter 1 → Input of inverter 2
  - Output of inverter 2 → Input of inverter 1

This structure ensures that the output of one inverter directly controls the input of the other.

Functionality (Bistability):

- The circuit has **two stable states**:
  - **State 1**: Output of inverter 1 = 1, inverter 2 = 0
  - **State 2**: Output of inverter 1 = 0, inverter 2 = 1
- There is also one **unstable equilibrium point** (where both outputs are at intermediate voltage, near threshold voltage V_{th}).

Any small disturbance at this point causes the circuit to resolve into one of the two stable states — hence the **memory effect**.

Memory Application:

- This bistable behavior is exploited to **store a single bit** of information.
- The circuit **remains in its current state indefinitely** until forced to switch by an external signal.

- It is a foundational structure for SR latches, D latches, and flip-flops.

Behavioral Note:

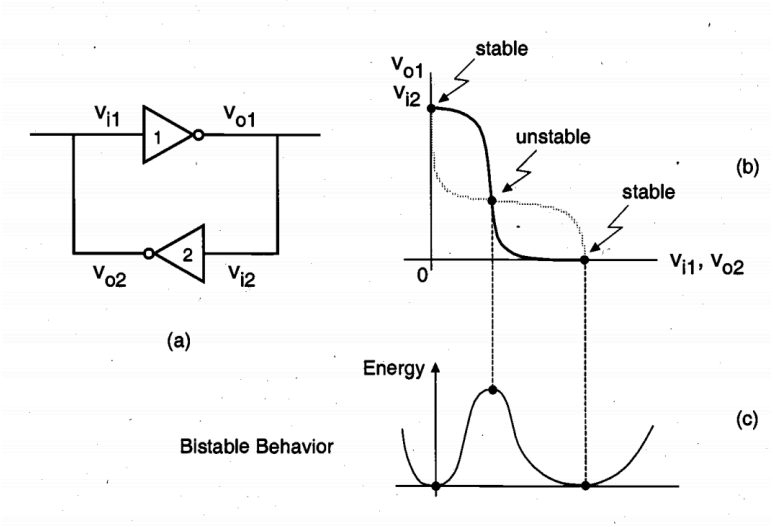
- When initialized or perturbed near the unstable point, the internal positive feedback causes the outputs to diverge exponentially, settling into a stable state.

M5Q4) Explain behaviour of 2-inverter basic bistable element.

A 2-inverter basic bistable element consists of two cross-coupled CMOS inverters, forming a feedback loop. This configuration exhibits *bistable* behavior, meaning it has two stable operating states and one unstable state.

Structure and Working

- In this circuit, the output of inverter (1) is connected to the input of inverter (2), and vice versa (i.e., the output of inverter 2 feeds back into the input of inverter 1).
- As a result, the voltage at the output of one inverter determines the input of the other and vice versa, enforcing a strong feedback loop.



Static behavior of the two-inverter basic bistable element:
(a) Circuit schematic; (b) Intersecting voltage transfer curves of the two inverters, showing the three possible operating points.; (c) Qualitative view of the potential energy levels corresponding to the three operating points.

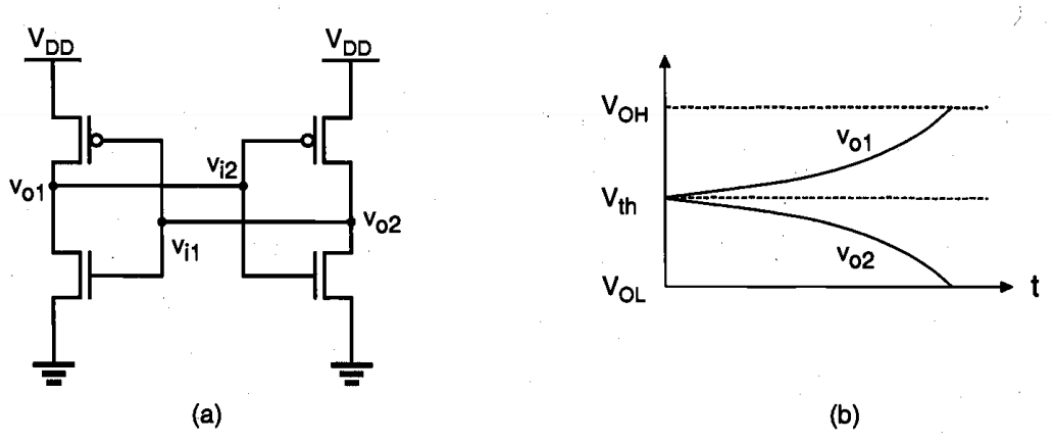
Static Behavior:

- The static voltage transfer characteristics (VTCs) of the two inverters intersect at **three points**:
  - **Two stable points** where the slope of both VTCs is less than one.
  - **One unstable point** where the slope (gain) of both VTCs is greater than one.

> When the system is at a stable point, small perturbations do not change the state. However, if at the unstable point, even a small disturbance can cause the state to flip to one of the stable points.

Operating Points:

- **Stable Point 1:** Output of inverter 1 is VOH, and inverter 2 is VOL.
- **Stable Point 2:** Output of inverter 1 is VOL, and inverter 2 is VOH.
- **Unstable Point:** Both outputs are at intermediate voltage (around threshold voltage, Vth).



(a) Circuit diagram of a CMOS bistable element. (b) One possibility for the expected time-domain behavior of the output voltages, if the circuit is initially set at its unstable operating point.

Time-Domain Behavior:

- If initialized at the unstable point, any small input disturbance will cause exponential divergence of the outputs — one heading toward logic high

(VOH) and the other toward logic low (VOL), depending on the polarity of initial perturbation.

Memory Function:

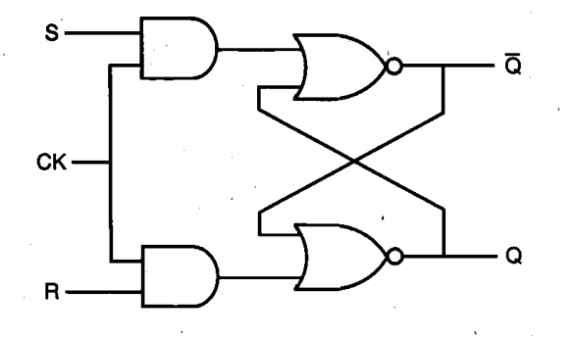
- The bistable element can retain one of the two stable states indefinitely, acting as a **1-bit memory element**. However, it cannot change state by itself — it requires external forcing (triggering) to move from one stable state to another.

M5Q5) Explain Clocked NOR-based SR Latch.

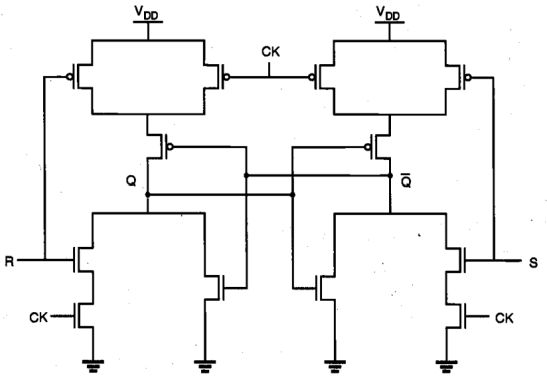
A **Clocked NOR-based SR Latch** is a sequential logic circuit designed to store a binary state while responding to input signals only when a **clock (CK)** is active. This design helps implement **synchronous behavior** in digital systems.

Working Principle:

- The circuit consists of:
  - **Two AND gates** feeding into
  - **A basic SR latch (NOR-based)**
- The **Set (S)** and **Reset (R)** signals are each ANDed with the **Clock (CK)** signal.
- When **CK = 0**, both AND gate outputs are 0, hence the SR latch holds its previous state.
- When **CK = 1**, the current values of S and R propagate to the latch.



Gate-level schematic of the clocked NOR-based SR latch

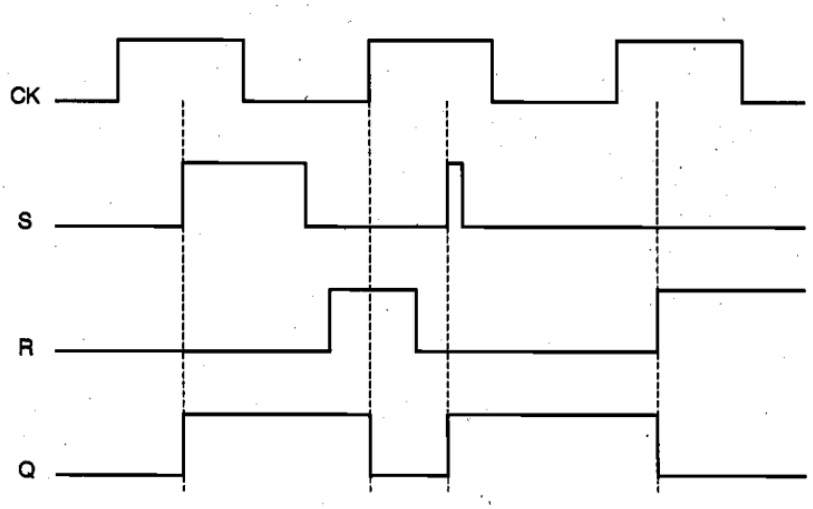


AOI-based implementation of the clocked NOR-based SR latch circuit.

Truth Table (for CK = 1):

S	R	Output (Q)	State
0	0	No change	Hold
1	0	1	Set
0	1	0	Reset
1	1	Undefined	Invalid state

> ⚠️ When S = R = 1 while CK = 1, both outputs go to 0 temporarily — leading to an indeterminate final state after the clock goes low.



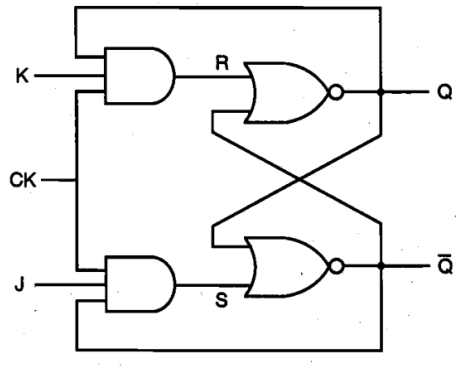
Sample input and output waveforms illustrating the operation of the clocked NOR-based SR latch circuit.

Level Sensitivity:

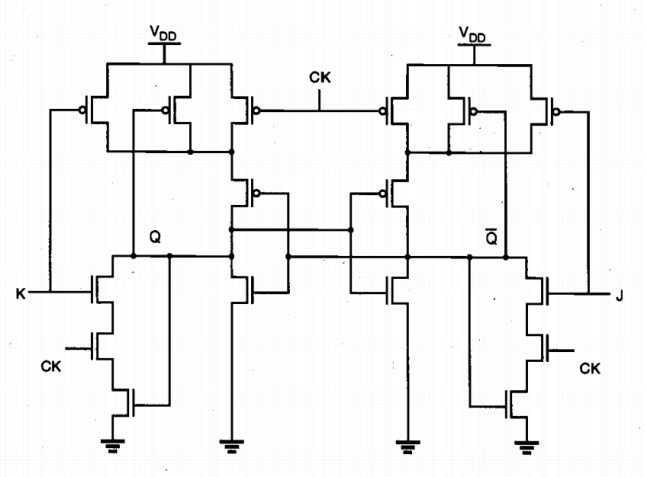
- The latch is **level-sensitive**, not edge-triggered.
- Any glitch in S or R while CK = 1 can affect the output if the glitch duration is longer than the internal delay.

M5Q6) Explain Clocked NOR-based JK Latch.

The **Clocked NOR-based JK Latch** is a refinement of the SR latch that removes the invalid state associated with **S = R = 1**, by using feedback to allow controlled toggling. It is level-sensitive and operates synchronously with a clock signal.



Gate-level schematic of the clocked NOR-based JK latch circuit.



CMOS AOI realization of the JK latch

Circuit Operation:

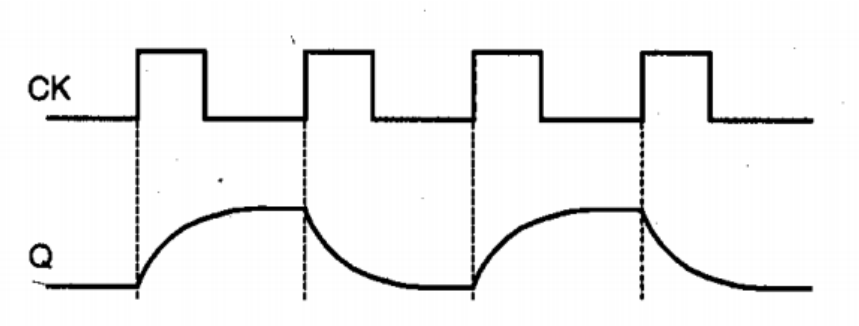
- The JK latch uses two NOR gates configured similarly to an SR latch.
- Additional feedback is introduced from the outputs **Q** and **Q̄** back to the inputs, creating **J** and **K** input logic.
- The latch is **clock-enabled**: the inputs J and K affect the latch **only when the clock (CK) is HIGH**.

Truth Table (when CK = 1):

J	K	Q(next)	Operation
0	0	Q	Hold
0	1	0	Reset

J	K	Q(next)	Operation
1	0	1	Set
1	1	$\bar{Q}$	Toggle

- **Hold**: The latch maintains its current state.
- **Set**: $Q \rightarrow 1$, $\bar{Q} \rightarrow 0$
- **Reset**: $Q \rightarrow 0$, $\bar{Q} \rightarrow 1$
- **Toggle**: Q toggles its state on every clock pulse while J = K = 1



Operation of the JK latch as a toggle switch

Important Notes:

- **No undefined state** like in SR latch.
- However, **if clock pulse is too wide** and **J = K = 1**, the output may **toggle uncontrollably**. To avoid this:
  - Keep **clock pulse width short** (shorter than propagation delay).
  - Or use edge-triggered configurations (like master-slave).

M5Q7) Write a note on built-in Self Test.

Self-test techniques are **Design-for-Testability (DFT)** methods that allow integrated circuits to **test themselves** using on-chip logic. These methods improve test coverage and enable **in-field** or **at-speed testing** without the need for expensive external testers.

Signature Analysis and BILBO (Built-In Logic Block Observation)

- **Signature Analysis** uses a **Linear Feedback Shift Register (LFSR)** to compact output responses into a **unique signature**.
- At the end of testing, the signature is compared to a reference value.
- **BILBO** combines LFSR and scan register functionality and can operate in four modes:
  - PRSG (test generation)
  - Signature analyzer (response compaction)
  - Scan register
  - Normal operation

Memory Self-Test

- Used for testing embedded RAM and ROM.
- Involves writing known patterns (like checkerboard, complement) and reading them back.
- Needs:
  - Address counter
  - Multiplexers
  - Simple state machine

Iterative Logic Array (ILA) Testing

- Used for testing regular structures like datapaths.
- Exploits **repetition** in logic modules to reduce test vectors.
- An array is:
  - **C-testable** if constant number of vectors are sufficient regardless of size.
  - **I-testable** if fault effect is the same for all modules with one test vector.

IDDQ Testing (Current Monitoring Test)

- Based on the principle that CMOS draws **no DC current** in steady state.
- Bridging faults or gate oxide shorts cause abnormal static current (IDDQ).
- The current is monitored to detect defects.

→ Low-cost method with high fault coverage but slow test speed.

Summary Table:

Technique	Purpose	Key Hardware Used
Signature Analysis	Response compaction	LFSR
BILBO	PRSG + Signature + Scan	LFSR-based multi-mode logic
Memory Self-Test	RAM/ROM validation	Address counter, MUX, FSM
Iterative Logic Array	Test regular logic arrays	Repetition and comparison
IDDQ Testing	Detect static faults	Current monitor circuit

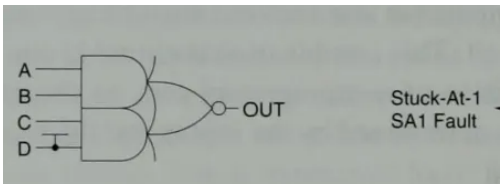
M5Q8) Explain Fault Models.

Fault models are abstract representations of physical defects in an integrated circuit (IC) that help in simulating and detecting errors. These models bridge the gap between real-world manufacturing defects and logical simulation for testing purposes.

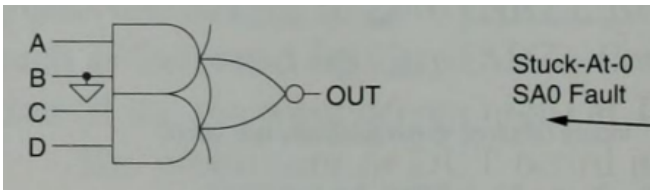
Common Fault Models:

1. Stuck-At Fault (SAF) Model

- The most widely used model in IC testing.
- Assumes a signal line or gate input/output is permanently stuck at logical **0 (S-A-0)** or **1 (S-A-1)**.
- **Example Causes:** Metal shorts, oxide defects, gate-to-substrate leakage.



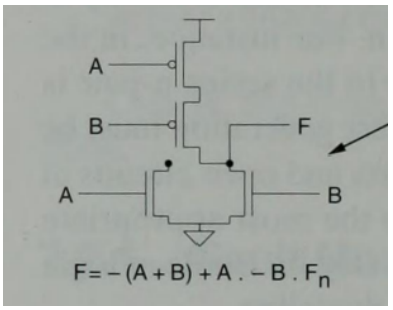
Stuck at 1



Stuck at 0

2. Stuck-Open Fault

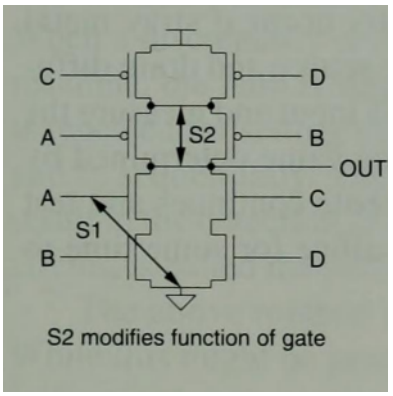
- Specific to CMOS circuits, where a **transistor fails to conduct** due to broken connections.
- The node **retains its previous charge**, leading to unpredictable behavior.
- Often requires **two-vector testing**.



Occurs due to a missing source, drain, or gate connection.

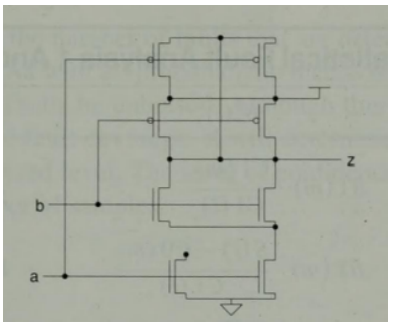
3. Bridging Faults

- Occur when two signal lines that should be separate are **shorted together**, altering the logic.
  - Can be modeled as **wired-AND or wired-OR faults**, depending on the circuit.



4. Delay Faults

- Arise when signal transitions are **too slow**, leading to incorrect logic operation at high speeds.
- Common causes include **process variation**, **interconnect issues**, and **hot-carrier effects**.



Link is open → works but with delay

5. Stuck-On / Stuck-Off Transistor Faults

- **Stuck-on:** Transistor conducts all the time, possibly causing short circuits or leakage.

- **Stuck-off:** Transistor never conducts, leading to logic failure or floating nodes.

Why Fault Models Are Important:

- Enable **automatic test pattern generation (ATPG)**.
- Help in **fault simulation**, **diagnostics**, and **test coverage estimation**.
- Form the basis for **design-for-testability (DFT)** strategies.

M5Q9) Techniques for reducing complexity of IC designs based on structured design strategies.

Structured design strategies are systematic methods used in VLSI design to **manage and reduce complexity**, **improve modularity**, and **facilitate team-based development**. These techniques mirror those used in software design and help ensure scalability, reliability, and reusability of ICs.

Key Techniques in Structured Design:

1. **Hierarchy** – "Divide and Conquer"
    - Large systems are broken down into **submodules**, which are further decomposed.
    - Each level of hierarchy represents a different abstraction (e.g., behavioral, structural, or physical).
    - Improves **comprehension**, **parallel development**, and **debuggability**.
    - Example: An 8-bit adder can be built from smaller 4-bit or 1-bit adders.
2. **Regularity** – Use of Similar Structures
    - Encourages **repetition** of similar or identical modules, such as arrays of gates or memory cells.
    - Reduces design effort and simplifies **verification**, **testing**, and **layout**.
    - Designs can be "**correct by construction**" due to predictable structure.
    - Example: Memory arrays or datapaths using identical processing elements.
3. **Modularity** – Well-Defined Interfaces

- Each module performs a **specific function** with a clear **interface**.
- Enables **independent development**, **reuse**, and **design validation**.
- Helps in team collaboration as designers can work on separate blocks simultaneously.
- Interface includes: I/O names, signal types, logic behavior, and power specs.

4. **Locality** – Minimize Global Interconnects
    - Promotes **local communication** among neighboring blocks rather than long-distance wiring.
    - Reduces **interconnect delays**, **power consumption**, and **noise**.
    - Supports faster clocking and more efficient routing.

Benefits of Structured Design Strategies:

- Shorter design cycles and **faster time-to-market**
- Improved **design productivity** and **testability**
- Better **scalability** to larger and more complex chips
- Easier debugging and **fault isolation**

********** EOF **********