

**(i) Define ASIC and classify its types.**

An ASIC (Application Specific Integrated Circuit) is an integrated circuit custom-designed for a particular application or function.

ASICs are classified as:

- **Full-custom:** Every layer of the chip is custom-designed for maximum optimization.
- **Standard-cell:** Built using a library of pre-designed logic cells, offering a balance of customization and productivity.
- **Gate-array:** Uses prefabricated arrays of gates; only the interconnect layer is customized for the application. maven-silicon

**Difference between CPLD and FPGA**

Feature	CPLD	FPGA
Architecture	Few, large programmable macrocells	Many small logic blocks (LUT-based)
Complexity	Best for simple, control-oriented designs	Suitable for complex digital systems
Timing	Predictable, fixed delays	Less predictable due to routing
Flexibility	Limited; reconfiguration less versatile	Highly flexible and reconfigurable
Usage	Glue logic, state machines	Data processing, computation-heavy

**(i) Define FPGA and explain its basic structure.**

A Field Programmable Gate Array (FPGA) is an integrated circuit that can be programmed and reprogrammed by the user after manufacturing to perform a variety of digital functions. ibm +1  
Its basic structure includes:

- Configurable Logic Blocks (CLBs): Perform logic operations using LUTs and flip-flops.
- Programmable interconnects: Link logic blocks together as needed.
- Input/Output blocks (I/O): Interface the FPGA with external devices.

## Difference between Design-Capture Tools and Synthesis Tools

- **Design-Capture Tools:**

Used for entering and describing the circuit design, either by schematic drawing or writing HDL (Hardware Description Language) code.

Examples: Schematic editor, Verilog/VHDL code editor.

- **Synthesis Tools:**

Convert the HDL code into a netlist of logic gates, optimizing the design for area, speed, and power.

Examples: Synopsys Design Compiler, Xilinx Vivado synthesis.

### (i) Purpose of simulation tools in VLSI design flow

Simulation tools are used to verify that a digital circuit behaves as intended before physical implementation.

They check logical correctness (functional simulation) and timing behavior (timing simulation) by running the design code with test vectors.

This helps detect logical bugs, timing errors, and ensures the design meets specifications before hardware is built. einfochips +1

### (ii) Two limitations of gate-array ASICs compared to standard-cell ASICs

1. **Lower Logic Density:** Gate-array ASICs have less efficient use of chip area, leading to lower packing of logic functions than standard-cell ASICs. maven-silicon +1
2. **Reduced Performance:** Gate-array designs are slower and less optimized for speed and power than standard-cell ASICs, as gate arrays have predefined layouts that limit critical path optimization.

### (i) Role of Place-and-Route Tools in ASIC Implementation:

Place-and-route tools arrange the physical locations of logic cells and automatically route the wires connecting them on the chip layout. These tools optimize for area, signal timing, and minimize wire length, ensuring the ASIC meets performance, power, and manufacturing requirements. anySilicon +1

### (ii) Two Factors that Affect Timing Closure in Synthesis:

1. **Cell and Library Choice:** The speed of selected cells and the quality of the timing libraries influence whether paths can meet timing.
2. **Load Capacitance and Wire Length:** High output loads and long interconnects increase delays, making timing closure more difficult.

### (i) Design Flow Steps of Standard-Cell Based ASIC

1. **Specification:** Define chip functions, performance, and constraints.
2. **RTL Design:** Create HDL code (Verilog/VHDL) describing logic.
3. **Functional Verification:** Simulate RTL to check correct behavior.
4. **Synthesis:** Convert HDL to gate-level netlist using standard cell libraries.
5. **Place & Route:** Physically arrange standard cells and route wires on silicon.
6. **Timing Analysis & Optimization:** Check timing and improve paths for speed.
7. **Physical Verification:** Run checks like DRC and LVS to ensure manufacturing correctness.
8. **Tape-out and Fabrication:** Prepare finalized design files for chip manufacturing. [einfochips +1](#)

### (ii) Role of Design Verification Tools in VLSI Design Methodology

Design verification tools ensure the IC functions as intended before fabrication.

- **Functional Simulation:** Tools simulate the logic code, helping catch errors early.
- **Formal Verification:** Tools mathematically prove correctness for certain logic properties.
- **Static Timing Analysis:** Confirms timing constraints for reliable speed.
- **Physical Verification:** DRC (Design Rule Check) and LVS (Layout vs Schematic) validate layout matches logic/silicon rules.

Using these tools avoids costly bugs, confirms design integrity, and ensures the final chip meets specifications and manufacturing requirements. [slideshare +1](#)

**(i) Outline the types of ASICs with examples (full-custom, standard-cell, gate array):**

- **Full-Custom ASIC:**

Every layer and transistor is custom-designed for the specific application, providing maximum performance and area optimization.

*Example:* Advanced microprocessors, GPUs, or high-end networking chips designed from scratch. [wikipedia +1](#)

- **Standard-Cell ASIC:**

Built using a library of pre-designed logic cells, offering a good balance of customization, time to market, and efficiency.

*Example:* Commercial DSPs (Digital Signal Processors), application processors in smartphones using standard cell libraries for faster development. [sciencedirect +1](#)

- **Gate-Array ASIC:**

Consists of prefabricated arrays of unconnected gates or transistors, customized for the application by designing only the final metal layers.

*Example:* Control ICs for low-volume custom applications, implemented quickly and cost-effectively with moderate performance. [anysilicon +1](#)

**(ii) Explain the importance of synthesis in ASIC design flow:**

Synthesis is the process of translating high-level design code (HDL like Verilog/VHDL) into a gate-level netlist that can be implemented in hardware.

- It automates logic optimization for speed, area, and power, converting functionality from abstract RTL to actual gates and flip-flops.
- Synthesis checks for logical correctness, applies timing constraints, and helps meet chip specifications.
- The synthesized netlist serves as the foundation for physical design steps, ensuring efficient and reliable hardware realization for the ASIC.

**(i) VLSI Design Cycle from Specification to Fabrication**

1. **Specification:** Define the functions, performance, and constraints of the chip.
2. **Design Entry:** Write RTL code (Verilog/VHDL) or create schematics for circuit blocks.
3. **Functional Verification:** Simulate the design to catch logic errors before synthesis.
4. **Synthesis:** Convert RTL to a gate-level netlist using cell libraries.
5. **Place & Route:** Arrange logic cells physically and connect them with optimized wires.
6. **Timing and Physical Verification:** Confirm that the layout meets timing, DRC, and LVS rules.
7. **Tape-Out and Fabrication:** Send the verified layout to the foundry to physically manufacture the chip.

## (ii) Role of EDA Tools at Each Stage of the Cycle

- **Specification:** Project management and requirements capture tools help document chip goals.
- **Design Entry:** HDL editors and schematic capture tools enable efficient design input.
- **Functional Verification:** Simulation tools (e.g., ModelSim) verify logic and timing.
- **Synthesis:** Synthesis tools (e.g., Design Compiler) optimize logic and generate gate-level netlists.
- **Place & Route:** P&R tools (e.g., Cadence Innovus) manage cell layout and wire routing for performance and area.
- **Verification:** Static timing analysis and physical verification tools (DRC, LVS) ensure rules and timings are met.
- **Tape-Out:** Data packaging and format conversion tools prepare files for foundry manufacturing.

## (i) Operation and Design Features of FPGA Fabric Based on Lookup Tables

An FPGA uses **Lookup Tables (LUTs)** as the main logic blocks in its fabric.

- **Operation:** Each LUT can implement any logic function of its input bits by storing truth table values. Logic signals are fed into LUTs and the pre-programmed output is used for further computation.
- **Structure:** LUTs are combined with flip-flops for sequential logic, and are surrounded by programmable interconnects that allow flexible connections between blocks.
- **Features:**
  - Highly reconfigurable: logic and routing are both programmable.
  - Fast prototyping and updates after manufacturing.
  - Suited for implementing custom digital circuits without fabricating a new chip.

## (ii) Compare FPGA and CPLD Architectures: Speed, Area and Power

Feature	FPGA	CPLD
Speed	Generally faster for complex and highly parallel designs due to more routing resources and deep pipelining.	Faster for simple control logic, as propagation delay is predictable and small.
Area	Larger area for big designs because lots of logic cells/routing; scales well for complex systems.	Smaller area for moderate logic functions; less scalable for large or complex tasks.
Power	Can use more power due to many clocked elements and dense routing. Power-saving modes available but less effective in large logic.	Lower power for small/medium control circuits; predictable static power, minimal dynamic switching.

### (i) Explain Static CMOS Logic with an Example

Static CMOS logic uses complementary pairs of pMOS and nMOS transistors to realize digital gates.

For every input combination, either the pull-up (pMOS) or pull-down (nMOS) network connects the output to Vdd or GND, giving strong logic levels and low static power consumption.

*Example: CMOS Inverter*

- pMOS connects output to Vdd when input is LOW.
- nMOS connects output to GND when input is HIGH.
- Output is always strongly driven, so power is only used during input switching.

Schematic:



## (ii) Distinguish between Pass-Transistor Logic and Transmission-Gate Logic

- **Pass-Transistor Logic:** Uses only nMOS (or only pMOS) transistors to pass signals, often resulting in voltage degradation (output may not reach full Vdd/GND due to threshold voltage drop).
- **Transmission-Gate Logic:** Uses both nMOS and pMOS transistors in parallel, allowing both high and low signals to pass without voltage loss; controlled by complementary gate signals, providing better signal integrity and full voltage swing.

Aspect	Pass-Transistor Logic	Transmission-Gate Logic
Devices Used	Only nMOS or pMOS	nMOS and pMOS together
Signal Swing	May degrade	Full Vdd/GND levels
Control Signals	Single	Complementary pair
Usage	Multiplexers, simple gates	Latches, flip-flops, advanced mux

## (i) Explain pseudo-NMOS logic with a neat diagram.

Pseudo-NMOS logic uses a set of nMOS transistors to realize pull-down logic and a single, always-ON pMOS transistor as a pull-up device. It reduces transistor count and area compared to full CMOS but consumes static power continuously.

*Diagram:*

```
text
Vdd
|
pMOS (always ON)
|
-----| Output
|
nMOS network (inputs A, B, etc.)
|
GND
```

When any nMOS turns ON, output goes LOW; otherwise, the pMOS pulls the output HIGH.

**(ii) List two limitations of dynamic logic circuits.**

1. Dynamic logic is susceptible to charge leakage, causing logic levels to decay if not regularly refreshed.
2. It is sensitive to noise and requires strict timing control, making it less robust than static logic for large and slow circuits.

**(i) Define logical effort in CMOS design and state its use.**

Logical effort is a measure of how much more input capacitance a gate presents compared to a reference inverter, to achieve the same output current. It is used in CMOS design to analyze and minimize delay when sizing gates in logic paths, which helps engineers design faster digital circuits.

**(ii) Explain how transistor sizing affects propagation delay and noise margin.**

Increasing the width of a transistor reduces its resistance, lowering propagation delay and speeding up switching. However, larger transistors also increase the input and output capacitance, which can reduce noise margin and make the circuit more sensitive to voltage fluctuations. Thus, transistor size must be optimized for both speed and reliable operation.

**(i) Describe the principle of operation of transmission-gate logic.**

Transmission-gate logic uses a parallel combination of nMOS and pMOS transistors, controlled by complementary gate signals. When the control input is active, both transistors turn ON, allowing signals to pass through bidirectionally with full voltage levels. This overcomes threshold voltage losses seen in single-transistor pass logic, making transmission-gates ideal for high-fidelity digital switching and multiplexers. buzztech +1

**(ii) Mention two advantages of static CMOS over pseudo-NMOS logic.**

1. Static CMOS logic consumes negligible static power since only one network (pull-up or pull-down) is on at a time, avoiding constant current flow.
2. It provides full logic voltage swing at the output, ensuring robust and reliable signal levels, whereas pseudo-NMOS may suffer degraded high voltage and increased static power consumption.

**(i) Draw and explain the working of a dynamic NAND gate.**

A dynamic NAND gate consists of an nMOS pull-down network, a pMOS precharge transistor connected to the output node, and a clock signal controlling both.

*Working:*

- During the **precharge phase** (clock LOW), the pMOS turns ON, charging the output to HIGH.
- During the **evaluation phase** (clock HIGH), the nMOS network evaluates the inputs. If all inputs are HIGH, the output is discharged to LOW; otherwise, the output remains HIGH.

*Simple Diagram:*



**(ii) State the need for precharge and evaluation phases in dynamic logic.**

Precharge ensures the output starts at a known logic level (usually HIGH) before evaluation. The evaluation phase allows the pull-down network to conditionally discharge the node based on inputs. These phases prevent undefined states and loss of stored charge, enabling reliable switching and operation of dynamic gates.

**(i) Explain charge-sharing effect in dynamic logic circuits.**

Charge-sharing occurs when, during evaluation, a precharged output node unintentionally shares its charge with connected internal capacitances in the nMOS network. This can reduce the output voltage, potentially causing incorrect logic levels and compromising circuit reliability.

**(ii) Describe methods used to minimize charge loss in precharged nodes.**

Charge loss in precharged nodes is minimized by:

- Minimizing the number of series-connected nMOS transistors in the evaluation path.
  - Adding keeper transistors or weak feedback devices to sustain the output voltage.
  - Using shielding or guard devices to isolate the precharged node from other capacitances.
- These methods help maintain correct output levels throughout the clock cycle

### (i) Design a 2-input NAND Gate Using Transmission Gates and Draw Its Structure

To design a 2-input NAND gate with transmission gates:

- The transmission gates are used as controlled switches to implement the logic. Input signals control the transmission gates, enabling/disabling paths connecting the output to Vdd or GND.
- A simple way is to realize the NAND truth table: output is LOW only if both inputs are HIGH.
- For a practical CMOS circuit, a transmission gate arrangement is used for the switching and output pull-up/pull-down.

*Structure Diagram (ASCII representation)*

```
text
Vdd
|
Transmission Gate (enabled when either A or B is LOW)
|
output
|
Output
|
Transmission Gate (enabled when both A and B are HIGH)
|
GND

Control signals:
- First TG: Enable = NOT(A AND B), connects output to Vdd
- Second TG: Enable = (A AND B), connects output to GND
```

When both A and B are HIGH, the lower transmission gate turns ON, output connects to GND (logic LOW). For all other input combinations, the upper transmission gate is ON, output stays HIGH (logic HIGH).

## (ii) Analyze the Advantages of CMOS Transmission-Gate Logic over Pass-Transistor Logic

- **Full Voltage Swing:** Transmission gate logic uses both nMOS and pMOS in parallel, ensuring that both logic HIGH and LOW are passed without voltage degradation, unlike pass-transistor logic, which suffers threshold voltage loss.
- **Bidirectional Signal Transfer:** Transmission gates allow signals to propagate in both directions; pass-transistor logic is unidirectional and typically weaker for logic HIGH (with nMOS).
- **Robustness:** Transmission-gate circuits provide better noise margin and are less sensitive to supply fluctuations, improving reliability.
- **Drive Capability:** Because both nMOS and pMOS are used, the output drive strength is higher, supporting more loads and complex logic interconnections.
- **Versatility:** Transmission gates are widely used in multiplexers, latches, and flip-flops, offering more flexibility in digital design than the simpler pass-transistor structures.

## (i) Construct a 2-input XOR Gate Using CMOS Complex Logic

A CMOS XOR gate uses both nMOS and pMOS networks to realize the output function  $Y = A \oplus B = A' B + AB'$ :

- The pull-up (pMOS) network generates output HIGH when A and B are different (one HIGH, one LOW).
- The pull-down (nMOS) network connects output to GND for all other combinations.
- Proper arrangement of series and parallel transistors implements the XOR logic fully, ensuring full logic voltage swing and minimal static power consumption.
- The networks are optimized so only necessary paths are active, mirroring the sum-of-products logic for XOR.

## (ii) Evaluate Area and Power Trade-Offs of Complex CMOS Logic vs. Static CMOS

### • Area:

Complex CMOS logic (with multiple series and parallel transistors) is more area-efficient for implementing compound logic functions, since it can combine several gates within a single layout and reduce total transistor count compared to building with basic gates.

In contrast, pure static CMOS logic built only with inverters/NAND/NOR requires more gates and larger layout to achieve the same function, increasing area.

### • Power:

Complex CMOS logic typically reduces switching activity and wire lengths by minimizing intermediate signals, which lowers dynamic power consumption for large functions. However, its longer series paths can slow down transitions and incur higher short-circuit current during switching, potentially increasing transient power.

Static CMOS logic, with simpler structures, usually has lowest static power but may consume more dynamic power due to higher gate count and switching effort.

### (i) Explain Dynamic CMOS Logic Operation with Timing Waveforms

Dynamic CMOS logic operates using clocked phases:

- **Precharge Phase (Clock LOW):** The output node is charged to HIGH ( $V_{dd}$ ) by a pMOS transistor controlled by the clock. Input signals are ignored; output is set to high regardless of inputs.
- **Evaluation Phase (Clock HIGH):** The precharge pMOS turns OFF and the nMOS evaluation network turns ON. If inputs activate the nMOS path, the output node is discharged to LOW (GND). If the pull-down network isn't activated, the output remains HIGH.

*Timing Waveforms Explanation:*

- The clock signal alternates between LOW (precharge) and HIGH (evaluation).
- Output waveform shows periodic charging (HIGH) during precharge, potentially transitioning to LOW upon evaluation depending on input conditions.
- Any delay or leakage during evaluation may cause slow or incomplete discharge, showing a rounded or sloped transition in the output waveform.

### (ii) Discuss Design Measures for Reducing Charge Leakage in Precharged Nodes

To minimize charge leakage:

- **Keeper Transistor:** A weak pMOS (keeper) holds the charge on the output node, compensating for leakage current during evaluation.
- **Minimize Capacitance:** Limit the number of series nMOS transistors in the evaluation path to reduce capacitance and leakage sources.
- **Guard Devices:** Shield or isolate sensitive nodes from parasitic capacitance and leakage paths using protective transistors.
- **Short Evaluation Interval:** Design circuits so that evaluation happens quickly after precharge, reducing the time window for leakage.
- **Optimized Physical Design:** Careful layout and transistor sizing decrease leakage by minimizing unused and high-leakage areas.

### (i) Design a Full Adder Using Pass-Transistor Logic

A full adder can be built using nMOS pass-transistor logic by creating paths that selectively pass input signals to the output depending on the state of the control signals ( $A$ ,  $B$ ,  $C_{in}$ ).

- Sum output is realized using gates that selectively connect logic levels depending on XOR conditions between  $A$ ,  $B$ , and  $C_{in}$ .
- Carry output is formed by passing the correct logic state when at least two inputs are HIGH.
- The design uses fewer transistors by avoiding complementary structures and control signals directly drive the gates.

*Note: Actual diagram should be drawn separately; block arrangement includes series/parallel nMOS pass-gates for XOR and AND functions used in sum and carry.*

## (ii) Compare Its Transistor Count and Speed with Static CMOS Implementation

- **Transistor Count:**

Pass-transistor logic uses significantly fewer transistors to implement full adder functions since single transistors can act as switches and replace blocks of CMOS gates. For sum and carry, the count is much lower than static CMOS, which requires complete complementary pull-up/pull-down networks for each logic function.

- **Speed:**

Pass-transistor logic can offer faster operation for simple circuits since there are fewer gate delays and less capacitive load on outputs. However, speed advantage may decrease for cascaded stages due to signal degradation and loss in voltage swing (threshold drop), which requires restoration stages.

- **Summary:**

Pass-transistor designs save area and boost speed for simple arithmetic circuits, but suffer from limited signal levels and drive capability compared to static CMOS, which guarantees full voltage swing, noise margin, and better reliability for large-scale integration. Static CMOS trades off higher transistor count for robustness and consistency.

## (i) Describe the working principle of a CMOS D latch.

A CMOS D latch is a sequential circuit that stores one bit of data. It uses transmission gates and inverters. When the clock (Enable) is HIGH, the output Q directly follows the input D. When the clock is LOW, the circuit "latches" and holds its last output value, regardless of changes in input D.

---

## (ii) State two differences between level-triggered and edge-triggered flip-flops.

1. Level-triggered flip-flops accept and transfer input data as long as the clock remains at the active level; edge-triggered flip-flops update output only at the moment of a clock edge.
2. Level-triggered flip-flops are more susceptible to unwanted changes from input glitches during the active clock level, while edge-triggered flip-flops are less sensitive and provide precise data storage at each clock transition.

---

## (i) Define Setup Time and Hold Time for Sequential Circuits

- **Setup Time:** The minimum time interval before the clock edge during which the input data must remain stable to be correctly captured by the flip-flop.
- **Hold Time:** The minimum time interval after the clock edge during which the input data must continue to remain stable to avoid data corruption.

Both ensure reliable data capture and proper operation of sequential circuits. nationin

---

## (ii) Explain the Operation of an SR Latch with Truth Table

An SR (Set-Reset) latch is a basic memory circuit with two inputs, S and R, and two outputs, Q and  $\bar{Q}$ :

- When S=1 and R=0, the latch is set ( $Q=1$ ).
- When S=0 and R=1, the latch is reset ( $Q=0$ ).
- When S=0 and R=0, it holds the previous state.
- When S=1 and R=1, the output is undefined or invalid.

S	R	Q(next)	Description
0	0	Q(prev)	Hold
0	1	0	Reset
1	0	1	Set
1	1	Undefined	Invalid/Forbidden

The SR latch stores one bit of data based on the  $\downarrow$  and R inputs.

---

## (i) Identify the function of a Schmitt trigger in sequential circuits.

A Schmitt trigger is a comparator circuit with hysteresis used to clean up noisy or slow-rising input signals. In sequential circuits, it ensures reliable transitions by rejecting spurious noise, so the output toggles sharply only when inputs cross defined thresholds, improving signal integrity and timing accuracy.

---

## (ii) Explain the design of a 6-T SRAM cell using cross-coupled inverters.

A 6-T SRAM cell consists of two cross-coupled CMOS inverters forming a bistable latch. Two nMOS access transistors connect each inverter to bit-lines, controlled by a word-line. During a read or write, the access transistors turn ON to allow interaction with external circuitry; otherwise, the cross-coupled inverters hold the stored bit stably and efficiently.

**(i) State the function of a sense amplifier in memory design.**

A sense amplifier is a crucial circuit in memory arrays (such as SRAM and DRAM) used to reliably detect and amplify the small voltage difference generated by the selected memory cell on the bit-lines during read operations. The sense amplifier quickly turns a weak analog signal—which otherwise could not be interpreted as logic '1' or '0'—into a clear digital output, ensuring fast and error-free memory access.

**(ii) Illustrate the design of a row decoder for a memory array.**

A row decoder is built from a combination of logic gates (AND, NAND, or NOR) that take an n-bit binary address input and activate exactly one out of  $2^n$  word-lines, connecting the selected row in the memory cell array. For example, a 3-to-8 decoder takes 3 address inputs and enables only one of 8 word-lines at a time. The gates are arranged so that each unique input combination sets one output HIGH, ensuring only the addressed row of memory is accessed during read or write operations.

**(i) Explain the master–slave configuration of a D flip-flop**

- A master–slave D flip-flop consists of two cascaded latches: the master and the slave.
- The master latch is controlled by the clock signal, and the slave latch is controlled by the inverted clock signal.
- When the clock is HIGH, the master is transparent and captures the input D, while the slave holds its previous output.
- When the clock transitions LOW, the master latch holds its value, and the slave latch becomes transparent, outputting the stored value from the master to the output Q.
- This arrangement ensures data is latched on one clock edge and prevents intermediate glitches, producing stable output signals.

**(ii) Differentiate between asynchronous and synchronous latches**

Feature	Asynchronous Latch	Synchronous Latch
Control Signal	Controlled by an asynchronous enable	Controlled by a synchronous clock signal
Timing Sensitivity	Responds immediately when enable is active	Responds only on clock edges or levels
Application	Used for simple control and gating	Used in clocked sequential circuits
Behavior	Output can change spontaneously	Output changes synchronized with clock

### (i) Define Metastability in Flip-Flops

Metastability is a condition in flip-flops where the output enters an undefined or unstable state, neither logic '0' nor '1', typically due to violations of setup and hold time. In this quasi-stable state, the flip-flop's output may oscillate or remain indeterminate for a period before settling to a valid logic level. Metastability can cause unpredictable circuit behavior and timing errors in sequential systems. asic-world +1

### (ii) Explain How Setup and Hold Violations Lead to Metastability

Setup and hold violations occur when the input data does not remain stable long enough before and after the clock transition, violating the flip-flop's timing requirements. If data changes too close to the clock edge:

- The flip-flop cannot resolve the input to a definite '0' or '1' immediately.
- This causes the output to enter the metastable state.
- The circuit then requires extra time to resolve the output, increasing the risk of incorrect operation and delays in sequential logic.

Ensuring proper setup and hold times avoids triggering metastability and maintains reliable circuit operation.

### (i) Draw and Explain the Circuit of an Edge-Triggered D Flip-Flop

An edge-triggered D flip-flop consists of a combination of latches (or transmission gates and inverters) arranged so that it only changes output Q on a clock edge (usually rising or falling).

- *Circuit Structure:* The circuit typically includes two latches in master-slave configuration. The master latch captures data D when the clock is LOW, and the slave latch transfers the stored value to output Q when the clock transitions HIGH.
- *Operation:* The flip-flop is 'locked' against input glitches during most of the clock cycle, updating Q only at the instant of the clock edge. This ensures precise timing and stable data storage in synchronous circuits.
- *Features:* Minimizes timing errors and race conditions, making it ideal for synchronous registers and counters.
- *Diagram:* (To be drawn separately—master-slave D flip-flop using transmission gates, inverters, and clocked control lines.)

Feature	Static RAM Cell (SRAM)	Dynamic RAM Cell (DRAM)
Cell Structure	6 transistors (no capacitor)	1 transistor + 1 capacitor
Data Retention	As long as powered	Needs periodic refresh
Circuit Density	Lower (large cell)	Higher (small cell)
Power Usage	Higher active, low standby	Lower active, more standby due to refresh
Speed	Faster	Slower
Application	Caches, registers	Main memory, large arrays

### (i) Design a CMOS Edge-Triggered JK Flip-Flop

A CMOS edge-triggered JK flip-flop is built using master-slave latch configuration (pMOS and nMOS transmission gates and inverters).

- The circuit receives JK inputs, clock, and provides Q and  $\bar{Q}$  outputs.
- Operation:*
  - On each clock edge, the JK logic controls switching:
    - J=1, K=0 sets the output Q.
    - J=0, K=1 resets Q.
    - J=K=1 toggles Q.
    - J=K=0 holds Q unchanged.
  - A master stage captures the JK inputs when the clock is LOW.
  - A slave stage transfers the result to Q on the clock's rising (or falling) edge.

**(ii) Compare Latch-Based and Flip-Flop-Based Storage Elements**

Aspect	Latch-Based Storage	Flip-Flop-Based Storage
Timing	Level-sensitive, responds when enable/clock is high or low	Edge-triggered, responds only at clock transition (rising/falling edge)
Data Capture	Multiple changes possible during the active level	Only one update per clock event, more stable
Glitch Risk	Higher, sensitive to input noise or glitches during active level	Reduced, only updates at edge, immune to glitches between transitions
Circuit Simplicity	Fewer gates, simpler design	More gates, complex (requires two latches and clocking logic)
Usage	Fast transparent registers, timing pipelines	Synchronous registers, counters, sequential logic blocks

**(i) Explain the Operation of a 6-T SRAM Cell and Its Read/Write Cycle**

A 6-T SRAM cell contains:

- Two cross-coupled CMOS inverters that form a bistable latch and hold one bit of data.
- Two access nMOS transistors that connect the cell to bit-lines for reading and writing, controlled by the word-line.

*Operation:*

- **Write Cycle:** The word-line is activated, turning on access transistors. The desired value is applied to the bit-lines, overwriting the latched data in the cell.
- **Read Cycle:** The word-line enables the access transistors, allowing the stored data to drive the bit-lines. A sense amplifier detects the small voltage difference and amplifies it to a full logic level.
- The cell holds its data as long as power is supplied due to positive feedback of the inverters.

*(Diagram should be drawn with two inverters cross-coupled, joined to bit-lines via access transistors, and a word-line activating those transistors.)*

## (ii) Discuss the Function and Timing Requirements of the Sense Amplifier

- **Function:**

The sense amplifier in memory arrays (SRAM and DRAM) detects and amplifies the tiny voltage difference between the bit-lines during a read operation. It converts minute analog signals from the selected memory cell into robust digital outputs that are correctly interpreted as logic '1' or '0'.

- **Timing Requirements:**

- The sense amplifier must activate after the word-line enables the cell and the stored value has started differentiating the bit-lines.
- Activating too early leads to insufficient voltage difference, causing errors; too late slows read speed.
- Optimal timing is crucial to guarantee fast, reliable, and error-free memory sensing.
- Must be precisely synchronized with word-line and bit-line signals for maximum memory speed and integrity.



## (i) Describe the Concept of Setup and Hold Violations in Synchronous Systems

Setup and hold violations occur when input data to flip-flops or registers fails to meet critical timing requirements:

- **Setup Violation:** Input data changes too close to the clock edge and does not remain stable for the specified setup time before the clock transition, potentially causing incorrect data to be captured.
- **Hold Violation:** Input data changes immediately after the clock edge and does not remain stable for the required hold time, risking corruption of the captured value.  
Both violations lead to unpredictable or metastable output states, causing logic and timing errors in synchronous circuits, and may result in data loss or system malfunction.

## (ii) Suggest Methods to Minimize Clock Skew and Timing Hazards

- **Clock Network Optimization:** Design balanced clock distribution trees or grids using matched-length and matched-load clock routes to deliver the clock signal uniformly to all registers.
- **Buffer Insertion:** Use carefully placed clock buffers and repeaters to manage propagation delays and balance clock arrival times.
- **Careful Layout:** Physically place synchronous elements (flip-flops, registers) close together to reduce wire delay differences.
- **Low-Jitter Clock Sources:** Use stable and low-jitter clock sources such as PLLs (Phase-Locked Loops) or crystal oscillators for consistent timing.
- **Timing Analysis Tools:** Employ EDA tools for static timing analysis and clock domain crossing checks to detect and correct hazards early in the design flow.