Problem Statement:
FSM: Sequential Logic Operator Selector

Objective
Design and verify a finite state machine (FSM) that selects among logic operators (OR, AND, XOR) to operate on two 1-bit inputs. The active operator is determined by the FSM's current state, and the machine advances to the next operator based on a control input.

Specifications

Interfaces
- Inputs: clk, rst (active-high), c (control), a (1-bit), b (1-bit)
- Outputs: y (1-bit)
- Optional status outputs (recommended for debugging): or_flag, and_flag, xor_flag (one-hot)

States
Define exactly three operational states and one reset state:

- S_RST: Reset/idle state (Moore output y=0)
- S_OR: y = a | b
- S_AND: y = a & b
- S_XOR: y = a ^ b

Reset & Sequencing
- On rst=1, the FSM must go to S_RST immediately.
- From S_RST: if c=1, move to S_OR; if c=0, stay in S_RST.
- Operational cycle (when c=1): S_OR → S_AND → S_XOR → S_OR → …
- If c=0, hold the current state (no change).

Datapath Behavior
Let a and b be 1-bit inputs.

- In S_OR: y = a | b
- In S_AND: y = a & b
- In S_XOR: y = a ^ b
- In S_RST: y = 0

Design Tasks

1) Draw a state diagram (bubbles and labeled transitions).

2) Write the Verilog RTL (single module or controller+datapath) using parameter-encoded states

3) Provide a concise testbench that:

- Applies reset
- Holds c=0 to show no state change
- Pulses c to step S_OR → S_AND → S_XOR → S_OR
- Toggles a and b to demonstrate correct y in each state

Deliverables

- Verilog source files (.v)
- Testbench (.v)
- Simulation waveforms