**Basic FSM Concepts**

1. **What is a finite state machine? How is it used in digital design?**

The finite state machine is a mathematical model which tells about the design behavior of a digital system. FSMs are widely used as state-based systems, for example, Traffic controllers.

1. **What are the different types of FSMs?**  
   (Mealy and Moore machines – explain the difference.)

**There are two kinds of FSMs:** Mealy and Moore machines.

**Mealy Machine:** where output is dependent on the present state and the input.

* Output value changes when there is a input change.
* Mealy Machine has less number of states
* It is complex and react faster
* Output changes asynchronous and state changes to synchronous to clock
* Output and input are both placed outside of the state

**Moore Machine:** Where output is dependent only on on present state

* Output value changes once state changes at the clock edge.
* Moore Machine has more number of states
* Its slower and simple.
* Output and state both changes synchronously to clock
* Output placed within the state

1. **Can you explain the difference between a combinational circuit and a sequential circuit, and where FSM fits in?**

**combinational circuit:**

* Output is dependent only on input.
* No feedback from output to input.
* Clock independent
* Easy to design and implement
* Hardware cost is less
* Faster as its output changes immediately when a change in input
* Building block is Logic gates
* Example is adder

**Sequential circuits:**

* Output is depend on present and past inputs
* Memory element is there
* Feedback is there from output to input
* Synchronous to clock
* Slower compared to combinational circuit
* Design is slightly complex so hardware and cost is more
* Sequential circuit's purpose is to store the value, combinational can not
* Basic building block is FF
* Example is counter

FSM is a sequential circuit because it has a memory element which stores the current state and the combination of the current state and input determines the output and next state. As FSM depends on the clock.

1. **What are the essential components of an FSM?**

Essential components of FSM are

* Inputs: inputs are the external signal which is responsible in the transitioning of states,
* Outputs: Outputs are produced by the system depending on the current state and input.
* State transition: The transition of states occurs at the clock edge which means the design is transitioning from the previous state to the next state.
* State diagram: the graphical representation of states.
* Registers: which are responsible for holding the state.

**Design and Implementation**

1. **How do you design a finite state machine for a given problem?**

I would follow the systematic approach to design an

FSM:

* 1. Understand the problem carefully, such as what can be the input-output, and what are design requirements.
  2. Instead of working with complex problems. Breakdown the problem into smaller to ensure that are you working on the right direction
  3. List all the inputs and outputs for each stage.
  4. make a state diagram and do the state transition based on encoding and identify which FSM I am going to use.
  5. Then make a state table
  6. Implement on software and hardware by writing code for FSM.
  7. Verify the FSM by functional verification

1. **How would you model an FSM for a traffic light controller?**
2. **Explain how you would design a vending machine FSM with inputs like coin insertion and outputs like dispensing items or change.**
3. **Describe how to implement an FSM in SystemVerilog. How do you handle states and transitions in your code?**
4. **What is the importance of a state encoding scheme (e.g., binary, gray, one-hot)? Which would you use in your design and why?**

State encoding is a crucial part of FSM because it represents the state, that says the impact on Area, power consumption, timing, and debugging.

If area is priority, then I will go with binary encoding as less number of FF, impacts the area. So this is suitable for design which has area is limiting factor.

If performance is priority than I would prefer one hot encoding , because its simpler,so this will I use with protocols or processors.

If power consumption is priority then I would prefer to go with grey encoding as it has less number of transition , so this would be useful for low power devices

If design has unique requirements then I would say to use the custom encoding style.

Example Processor: one hot encoding and gray encoding can be good for this.

1. **What challenges arise when designing FSMs for hardware, and how do you resolve them?**
   1. Challenge: The complexity of design increases, then the FSM transition grows exponentially, which is harder to implement.

Resolution: follow the hierarchical FSM approach, and breakdown the FSM into smaller manageable ones.

* 1. Challenge: Due to asynchronous input many times, FSM goes into a metastable state.

Resolution: To ensure that FSM does not go into the metastable state we have to provide the synchronous to the input by using various CDC techniques such as a two-state synchronizer.

* 1. Challenge: Choosing the wrong encoding.

Resolution: To carefully understand and analyze the requirement and then use the appropriate encoding: for area: Binary encoding, for power: gray encoding, for performance: one hot encoding.

* 1. Challenge: Debugging and verify the Larger FSM on hardware is challenging

Resolution: We have to use the functional coverage matrix

**Verification of FSMs**

1. **How do you verify the functionality of an FSM? Similar like below**

 Understand the FSM design and identify test scenarios.

 Build a testbench with input drivers, monitors, and scoreboards.

 Write directed and constrained-random tests.

 Use functional coverage to track testing progress.

 Write assertions to detect illegal states or transitions.

 Use waveform analysis for debugging.

 Employ formal verification to ensure exhaustive correctness.

1. **What kind of coverage metrics are relevant for FSM verification?**  
   1. State coverage
   2. State transition coverage
   3. Input coverage
   4. Reset coverage
   5. Illegal coverage
   6. Path coverage
   7. Code coverage has branch statement, toggle and FSM also ensure that code has been excersised

Combination of functional and code coverage ensure that FSM is working properly.

1. **How would you test for unreachable states or illegal transitions in an FSM?**

Understand and analyze the FSM.

Write all the direct cases systematically for all possible inputs and transitions

To identify hidden scenarios, perform the constraint random test cases

Later I would write the covermodels with all the states and inputs as bins

Write the assertions to get any error or flag

Functional coverage and waveform analysis would tell me the illegal

1. **How do you verify the correctness of state transitions in UVM test benches?**

To verify the correctness of FSM by using UVM TB architecture needs to be developed that contains various components like driver, monitor, and scoreboard.

I would start the stimulus generation and send it to the driver via sequencer, my xtn includes the inputs, constraints, and methods. The sequencer sends the sequence to the driver and drives to the DUT of FSM. The monitor takes the output, transitions sends it to the scoreboard, and compares the results with the reference model.

I would also write the functional coverage and assertions to ensure the functionality of fSM.

1. **What kind of directed or constrained-random test cases would you write to verify an FSM?**

**Direct test cases: to try specific scenarios**

* 1. Check the initialization,
  2. Check the reset
  3. Check the state-to-state transition
  4. Check the end-to-end transition
  5. Verify the Output
  6. Provide the invalid inputs to get the verification

**Constrained Random test cases: to** **uncover the hidden scenarios**

1. Write the constraints
2. Randomize the inputs
3. Write the cover models or functional coverage
4. Write the assertions to check initialization, reset, and output

By using the combination of direct and random test the verification can be done.

**Debugging FSMs**

1. **What are common issues you might face in FSM design and implementation?**
2. **If an FSM enters an illegal state, how would you debug and fix it?**
3. **How would you ensure that all states in your FSM are reachable?**
4. **What happens if a state transition condition is ambiguous or undefined? How do you handle such cases in hardware?**

 **Explicit Design**: Define transition conditions clearly and unambiguously.

 **Default/Error States**: Add fallback states to handle undefined or ambiguous transitions.

 **Conflict Resolution**: Ensure mutually exclusive conditions for each transition.

 **Assertions**: Implement assertions to catch ambiguous transitions early.

 **Encoding**: Use clear and efficient state encoding schemes.

 **Verification**: Use simulation, functional coverage, and formal verification tools to detect and handle ambiguous conditions

1. **Have you encountered a scenario where the FSM got stuck in a state? How did you debug and resolve it?**

**Advanced Topics**

1. **How do you handle asynchronous inputs in an FSM?**

Async signal in FSM means the input is not following the FSM clock. That can lead to timing issues, metastability, or any incorrect transition. To handle the asynchronous input, we must ensure that your asynchronous input is synchronized prior fed into FSM. that is possible by two-stage ff synchronizer. In this initial input fed into the first ff and captures the asynchronous one but in the second ff captures the synchronous output which can be fed into FSM.

Later we can use Edge detection after synchronizing the input for reliable detection

1. **What is metastability, and how can it affect FSMs? How would you mitigate it?**
2. **What are the differences between synchronous and asynchronous FSMs? Which is preferred in design and why?**
3. **How would you design an FSM to handle priority inputs (e.g., interrupt handling)?**
4. **What is state explosion in FSMs, and how can you reduce it during design or verification?**

**Optimization**

1. **How do you optimize an FSM design for area, power, or timing?**
2. **What are the trade-offs between one-hot and binary state encoding for an FSM?**
3. **How would you reduce the number of states in an FSM without changing its functionality?**

**Real-World Application**

1. **Can you describe a real-world scenario where you used an FSM in one of your projects?**
2. **Explain how FSMs are used in protocols like AXI or PCIe.**
3. **Have you ever verified an FSM as part of a larger design, such as a packet router or a controller? What challenges did you face?**

**Scenario-Based Questions**

1. **Design an FSM to detect a specific sequence of bits (e.g., 10101) on a serial input. What are the states and transitions?**
2. **How would you design a FSM that generates a specific output pattern based on input conditions?**
3. **Suppose you have a 3-bit FSM with 8 states. If you find only 6 are being reached, how would you debug the issue?**
4. **Design an FSM that controls a 1x3 packet router, ensuring proper routing of packets based on the destination address.**