Finite State Machine

Design Steps

STEP 1: Based on the problem statement draw a top-level block diagram showing the inputs to (what you have) and the outputs from (what you want) the digital logic device to be designed. In many cases this may simply be a box with inputs and outputs. In any case the FSM(s) should be identified with its inputs and outputs clearly defined.

STEP 2: Working from the problem statement and your top-level block diagram select a Mealy or a Moore (Asynchronous or Synchronous) FSM and draw the corresponding block diagram.

STEP 3: Design the corresponding state transition diagram based on the problem and the FSM model selected. Verify that for each state all transition conditions are defined.

STEP 4: Assign binary codes to the states and, if needed, the input conditions.

STEP 5: Obtain the state transition table from the state diagram. Skip this step if you are implementing One-Hot State Encoding.

STEP 6: Design the Next State Decoder

a) Draw a K-Map for each next-state column

b) Define groups and check unused states. If needed adjust group(s) to create a stable design (i.e. design does not circulate among unused states) – Update state diagram to show unused states.

c) Derive D Flip-Flop input equations from the maps.

One-Hot State FSM: “You only care about the arrows coming at you.”

STEP 7: Design the Output Decoder

a) Draw a K-Map for each output column

b) Define groups

c) Derive output equations from the maps.

STEP 8: Draw the Logic Diagram or write HDL Description.

STEP 9: Simulate (Functional and Inertial) your circuit.

STEP 10: Prototype your design.

NOTES:

a) Do not forget to show the reset state

b) For each state the next state must be defined for all input conditions associated with the state.

Selecting a FSM Model

	Moore		**Mealy**	
Model Attributes	Synchronous	Asynchronous	Asynchronous	Synchronous
Simplicity	very simple	simple	complex	most complex
Stability	very stable	stable	least stable	stable
Speed – response to change in input	average – next clock cycle	slow – next clock cycle with delay	very fast – current clock cycle	average – next clock cycle
Optimization – maximum number of outputs per state n = state bits i = inputs	n	2^n	$2^{(n+i)}$	$2^{(n+i)}$

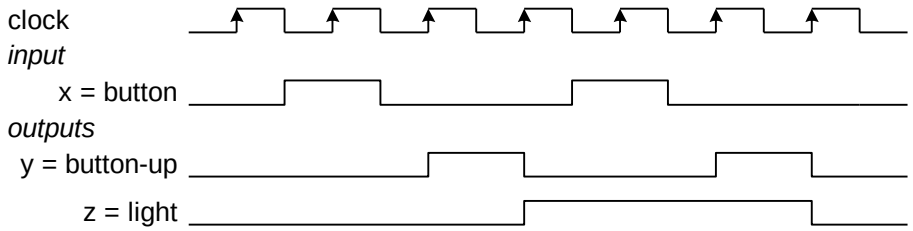
FSM Design Examples

Based on the "Design Steps" previously provided, design the eight FSM problems described below. Your team should provide the following:

In Class
- ⇒ A system level block diagram clearly defining inputs and outputs.
- ⇒ A model of your FSM with all components and busses (including size) identified.
- ⇒ A drawing of your state transition diagram.

At Home
- ⇒ From the state transition diagram fill-in the state transition table.
- ⇒ From the state transition table, construct K-maps, and write the Boolean expressions for the next state and output decoder circuits

1. Convert a D Flip-Flop into a JK Flip-Flop.
2. Convert a D Flip-Flop into a T Flip-Flop
3. You can electronically roll a die (design two and you can roll a pair of dice) by designing a counter that counts 1 – 4 – 5 – 6 – 7 – 0. Assume a single input (roll) that when true causes the die to be rolled. Design the FSM and determine if your FSM is stable (returns to a known state).
4. To launch a rocket the fire control officer must throw the arm switch and depress the fire button. The launch sequence is to be stopped if this exact sequence is not followed. Design the fire control sequencer.
5. An cruise control for a car has three outputs: accelerate, decelerate, and constant_speed. The FSM is modeled with four states (stopped, low, medium, and high); transitions between states depend on the state and the control inputs (brake and accelerate). The brake overrides the accelerator in the event both are asserted. If neither is asserted the cruise control outputs constant_speed.
6. A serial packet of data always begins with the start nibble 0 – 1 – 0 – 1. Design a FSM to detect the beginning of a serial packet and set a start flag.
7. An amusement ride operator has a button (0 = up, 1 = down) to toggle a stoplight (0 = green light, 1 = red light). In addition a second output is sent to the ride control computer indicating that a button-up event has occurred. Design this ride control sequencer.



8. Design a FSM to play a simple game of Tic-Tac-Toe. In this educational version of the game, the computer always goes first and takes square 8, the player must follow by taking square 1. The numbering of the board is illustrated at right. To enter a move the player sets three switches, allowing him to enter any one of the eight available squares (0 to 7), and presses a move button. Although the board has nine squares, the computer always takes the center square, leaving only eight. The computer output includes 4 lines, allowing it to place it's marker (traditionally an X) into any of the nine squares comprising the board. In addition to these 4 lines the computer has 2 lines for indicating an input error or a win condition. Do to the simplifying assumptions made in the order of play, the game will always result in a win by the computer. *For this problem you only need to complete the steps up to and including writing the Verilog Behavioral Description*.

0	1	2
7	8	3
6	5	4