Assignment #14

Finite State Machines

Practical Application of FSM (Mealy)

1. **Detect sequence 1011 in a serial bit stream.** *Hint: Use a 4-state FSM and output 1 when the full sequence is matched.* **(overlap allowed)**
2. **Detect sequence 110 with overlap allowed.** *Hint: Track overlapping by transitioning back to state for last 1.*
3. **Detect alternating pattern 101010...** *Hint: Toggle between expecting 1 and 0 alternately.*
4. **Detect any sequence ending with 1001.** *Hint: Maintain a 4-bit shift history in state transitions.* (Overlapping Not Allowed)
5. **Detect three consecutive 1's in a stream.** *Hint: Use states to count how many 1's have appeared in a row.*
6. **Detect pattern 0110 used in bit-stuffing protocols.** *Hint: Design FSM to output 1 upon detecting the full pattern.*
7. **Detect two 0's followed by a 1 (i.e., 001).** *Hint: Use 3 states to monitor input history.*
8. **Detect binary palindrome pattern (e.g., 1001, 0110).** *Hint: Track mirrored positions via FSM logic.*

State Transition Tables for Mealy FSM Exercises

1. Detect sequence 1011

Present State	Next State,Output	
	Input = 0	**Input = 1**
S0	(S0, 0)	(S1, 0)
S1	(S2, 0)	(S1, 0)
S2	(S0, 0)	(S3, 0)
S3	(S2, 0)	(S1, 1)

2. Detect sequence 110 (with overlap)

Present State	Next State,Output	
	Input = 0	**Input = 1**
S0	(S0, 0)	(S1, 0)
S1	(S0, 0)	(S2, 0)
S2	(S0, 1)	(S2, 0)

3. Detect alternating pattern 101010...

Present State	Next State,Output	
	Input = 0	**Input = 1**
S0	(S0, 0)	(S1, 0)
S1	(S0, 1)	(S1, 0)

4. Detect sequence ending with 1001

Present State	Next State,Output	
	Input = 0	**Input = 1**
S0	(S0, 0)	(S1, 0)
S1	(S2, 0)	(S1, 0)
S2	(S3, 0)	(S1, 0)
S3	(S0, 0)	(S0, 1)

5. Detect three consecutive 1's

Present State	Next State,Output	
	Input = 0	**Input = 1**
S0	(S0, 0)	(S1, 0)
S1	(S0, 0)	(S2, 0)
S2	(S0, 0)	(S0, 1)

6. Detect pattern 0110

Present State	Next State,Output	
	Input = 0	**Input = 1**
S0	(S1, 0)	(S0, 0)
S1	(S1, 0)	(S2, 0)
S2	(S1, 0)	(S3, 0)
S3	(S0, 1)	(S2, 0)

7. Detect sequence 001

Present State	Next State,Output	
	Input = 0	**Input = 1**
S0	(S1, 0)	(S0, 0)
S1	(S2, 0)	(S0, 0)
S2	(S2, 0)	(S0, 1)

8. Detect binary palindrome (e.g., 1001)

Present State	Next State,Output	
	Input = 0	**Input = 1**
S0	(S0, 0)	(S1, 0)
S1	(S2, 0)	(S1, 0)
S2	(S3, 0)	(S2, 0)
S3	(S0, 1)	(S3, 0)

Practical Application of FSM (Moore)

1. **Detect sequence 1011 in a serial bit stream.** *Hint: Use a 5-state FSM. Output = 1 is associated only with the final state.* **(overlap allowed)**

Present State	Output	Next State	
		Input = 0	Input = 1
S0	0	S0	S1
S1	0	S2	S1
S2	0	S0	S3
S3	0	S2	S4
S4	1	S2	S1

2. **Detect sequence 110 with overlap allowed.** *Hint: Use 4 states and set output=1 only in final state.*

Present State	Output	Next State	
		Input = 0	Input = 1
S0	0	S0	S1
S1	0	S0	S2
S2	0	S3	S2
S3	1	S0	S2

3. **Detect alternating pattern 101010...** *Hint: Output = 1 when expected bit is matched.*

Present State	Output	Next State	
		Input = 0	Input = 1
S0	0	S0	S1
S1	1	S2	S0
S2	1	S0	S1

4. **Detect any sequence ending with 1001.** *Hint: Use 5 states, output = 1 on final state.*

Present State	Output	Next State	
		Input = 0	Input = 1
S0	0	S0	S1
S1	0	S2	S1
S2	0	S0	S3
S3	0	S4	S1
S4	1	S0	S1

5. **Detect three consecutive 1's in a stream.** *Hint: Output 1 when three 1s seen continuously.*

Present State	Output	Next State	
		Input = 0	Input = 1
S0	0	S0	S1
S1	0	S0	S2
S2	0	S0	S3
S3	1	S0	S3

6. **Detect pattern 0110 used in bit-stuffing protocols.** *Hint: Output = 1 at final state.*

Present State	Output	Next State	
		Input = 0	Input = 1
S0	0	S1	S0
S1	0	S1	S2
S2	0	S3	S0
S3	1	S1	S2

7. **Detect two 0's followed by a 1 (i.e., 001).** *Hint: 3 states are sufficient.*

Present State	Output	Next State	
		Input = 0	Input = 1
S0	0	S1	S0
S1	0	S2	S0
S2	0	S2	S3
S3	1	S1	S0

8. **Detect binary palindrome pattern (e.g., 1001).** *Hint: FSM tracks mirrored states.*

Present State	Output	Next State	
		Input = 0	Input = 1
S0	0	S0	S1
S1	0	S2	S1
S2	0	S3	S2
S3	0	S0	S4
S4	1	S1	S1

Assignment

1. Design a Finite-State Machine (FSM) that generates a 3-bit Gray Code sequence on each clock cycle.

2. Design ann FSM based 3-bit even parity bit checker. Extend it to 4-bit after a successful implementation of 3 bit.

3. **Parking Payment System :** Allow entry when total coins inserted ≥ 20.
   Inputs: 5, 10, 20 coins
   States: Represent total amount inserted
   Output:

   - **0:** Gate closed
   - **1:** Gate opens (only in 20 state)

4. Design an FSM for a vending machine that sells an item for 10. The user can insert coins of Rs. 1, 2, or 5.
   States: 0, 1, 2, 5, 7, 10
   Inputs: Coin values: 1, 2, 5
   Output: Dispense item when 10 is reached.