# **Mealy Finite State Machine**

## **✅ Problem Statement**

Design a **Mealy FSM** that meets the following requirements:

* One input: w
* One output: z
* The output z = 1 **only if** the **last two inputs** were 1
* Transitions and output evaluations occur on the **positive edge of the clock**
* Implemented using a **Mealy-style FSM** where the output depends on the **state and input**

## **🤔 Why Use a Mealy FSM?**

In a **Mealy FSM**, the output is generated based on:

* The **current state**
* The **current input**

### **Benefits:**

* Requires **fewer states** than Moore FSM
* **Faster response** to input changes (output can change in the same clock cycle)
* More compact design in some applications

## **🔁 FSM Design Strategy**

### **Goal:**

Detect when two consecutive inputs w are 1. That means: z = 1 at the **second 1** in a row.

### **📊 State Description**

| **State** | **Meaning** | **Output z (if w = 1)** |
| --- | --- | --- |
| A | Initial or w = 0 | 0 |
| B | One 1 seen | 1 |

### **⭕ State Diagram**

This FSM needs only **two states**:

+---------+ w=1,z=0 +---------+

| A | -------------> | B |

| z=0 | | z=1 if w=1

+---------+ +---------+

^ | |

| | w=0,z=0 | w=0,z=0

+---+----------------------+

* **A → B**: When w = 1 (first 1 detected), go to state B, but output is still z = 0
* **B → B**: If another w = 1 arrives, stay in B and set z = 1
* **Any w = 0**: Return to A, and output is z = 0

## **💻 Verilog Implementation Breakdown**

module Mealy11(clk, reset, w, z, y, Y);

* clk: Clock input
* reset: Asynchronous reset (active-low)
* w: Input signal
* z: Output signal
* y: Current state
* Y: Next state

### **🔢 State Encoding**

parameter A = 0, B = 1;

* A: Starting state
* B: After detecting one 1

### **⏱ State Transition Logic**

always@(posedge clk, negedge reset)

* Triggered on **positive edge of clock** or **falling edge of reset**

if(reset==0)

y = A; // Reset to initial state A

else begin

y = Y; // Update current state to next state

casex(y)

A: if(w==1) Y = B; else Y = A;

B: if(w==1) Y = B; else Y = A;

endcase

end

* From A, go to B if w = 1, else stay
* From B, stay if w = 1, else go back to A

### **🧾 Output Logic (Mealy Style)**

assign z = ((y==B) & (w==1));

* Output z = 1 **only** when:  
  + Current state is B
  + AND input w = 1
* This reflects that we’ve seen **two consecutive 1s**

## **🧪 Example Input/Output Behavior**

| **Clock Cycle** | **w** | **State** | **Output z** |
| --- | --- | --- | --- |
| 1 | 1 | A → B | 0 |
| 2 | 1 | B | 1 ✅ |
| 3 | 0 | A | 0 |
| 4 | 1 | A → B | 0 |
| 5 | 1 | B | 1 ✅ |

## **✅ Summary**

| **Category** | **Mealy FSM Description** |
| --- | --- |
| Input | w |
| Output | z = 1 when last two ws were 1 |
| States | A and B |
| Output Dependency | Depends on both **state** and **input** |
| Number of States | Fewer (only 2 states) |
| Output Timing | Faster (can change in same cycle) |
| FSM Type | Mealy (output outside state block) |