# **Vending Machine FSM Code**

## **✅ Problem Statement**

Design a **finite state machine (FSM)** for a **vending machine** that sells a product costing **10 Taka**.

### **💵 Inputs:**

* Customers can insert money in 4 denominations:  
  + 0 Taka (00)
  + 5 Taka (01)
  + 10 Taka (10)
  + 20 Taka (11)

### **🎁 Outputs:**

* **purchase**: A signal that becomes 1 when the product is dispensed.
* **cash\_return**: The amount of change to be returned (0, 5, 10, or 15 Taka).

### **🧠 Objective:**

* Accumulate money inserted across states.
* Dispense product when at least **10 Taka** is collected.
* Return change if more than 10 Taka is inserted.
* Reset to initial state after a product is dispensed or canceled.

## **🧩 Solution Overview**

The solution is implemented as a **Mealy FSM** because:

* The **output depends on both the current state and input**.
* The FSM uses a clock to transition between states and generate output in response to money input.

## **🔄 FSM Design Process**

### **➕ States:**

| **State** | **Meaning** |
| --- | --- |
| S0 | 0 Taka inserted (initial state) |
| S1 | 5 Taka inserted |

Other possible states (10, 15 Taka) are not used in this simplified version.

## **🔁 State Transitions & Output Behavior**

### **From S0 (0 Taka inserted):**

| **cash\_in** | **Meaning** | **Next State** | **purchase** | **cash\_return** |
| --- | --- | --- | --- | --- |
| 00 | No money | S0 | 0 | 0 |
| 01 | 5 Taka | S1 | 0 | 0 |
| 10 | 10 Taka | S0 | 1 | 0 |
| 11 | 20 Taka | S0 | 1 | 10 |

### **From S1 (5 Taka inserted):**

| **cash\_in** | **Meaning** | **Next State** | **purchase** | **cash\_return** |
| --- | --- | --- | --- | --- |
| 00 | Cancel/refund | S0 | 0 | 5 |
| 01 | +5 = 10 Taka | S0 | 1 | 0 |
| 10 | +10 = 15 Taka | S0 | 1 | 5 |
| 11 | +20 = 25 Taka | S0 | 1 | 15 |

## **📈 FSM State Diagram**

cash\_in=01 (5 Taka)

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

| v

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

| S0 | | S1 |

| (0 Tk) | | (5 Tk) |

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

^ | ^ |cash\_in=10 |

| | | |(10 Taka) |

| | | v |

| | +---> [Dispense] |

| | z=1, return=0 |

| | |

| | cash\_in=11 (20 Taka) |

| +--> [Dispense] z=1, R=10 |

| |

| cash\_in=00 (cancel) |

+<-------------------------------+

[No product] z=0, R=5

## **💻 How the Verilog Code Solves the Problem**

### **1. State Definition**

parameter state0 = 2'b00, state1 = 2'b01;

Defines S0 and S1 as the two main states.

### **2. Reset Logic**

if (reset == 1)

Resets both present\_state and next\_state to S0.

### **3. Transition Logic**

case (present\_state)

state0: ...

state1: ...

Depending on current state and input, the next state and outputs (purchase, cash\_return) are calculated.

### **4. Output Logic**

* If enough money is received, purchase = 1
* If extra is received, change is returned via cash\_return
* If canceled, refund is given

## **🧠 Design Notes**

* This is a **Mealy FSM** because outputs are affected by both state and input.
* The FSM resets to state0 after every transaction, simulating a real vending machine cycle.
* You can **extend this FSM** to include more states (e.g., state2 for 10 Taka saved) for finer control.

## **✅ Conclusion**

This FSM effectively models a vending machine logic for a **10 Taka product**. It:

* Accepts multiple denominations
* Manages product dispensing and change
* Uses minimal logic with just two states (S0, S1)
* Demonstrates core FSM principles in digital design using Verilog