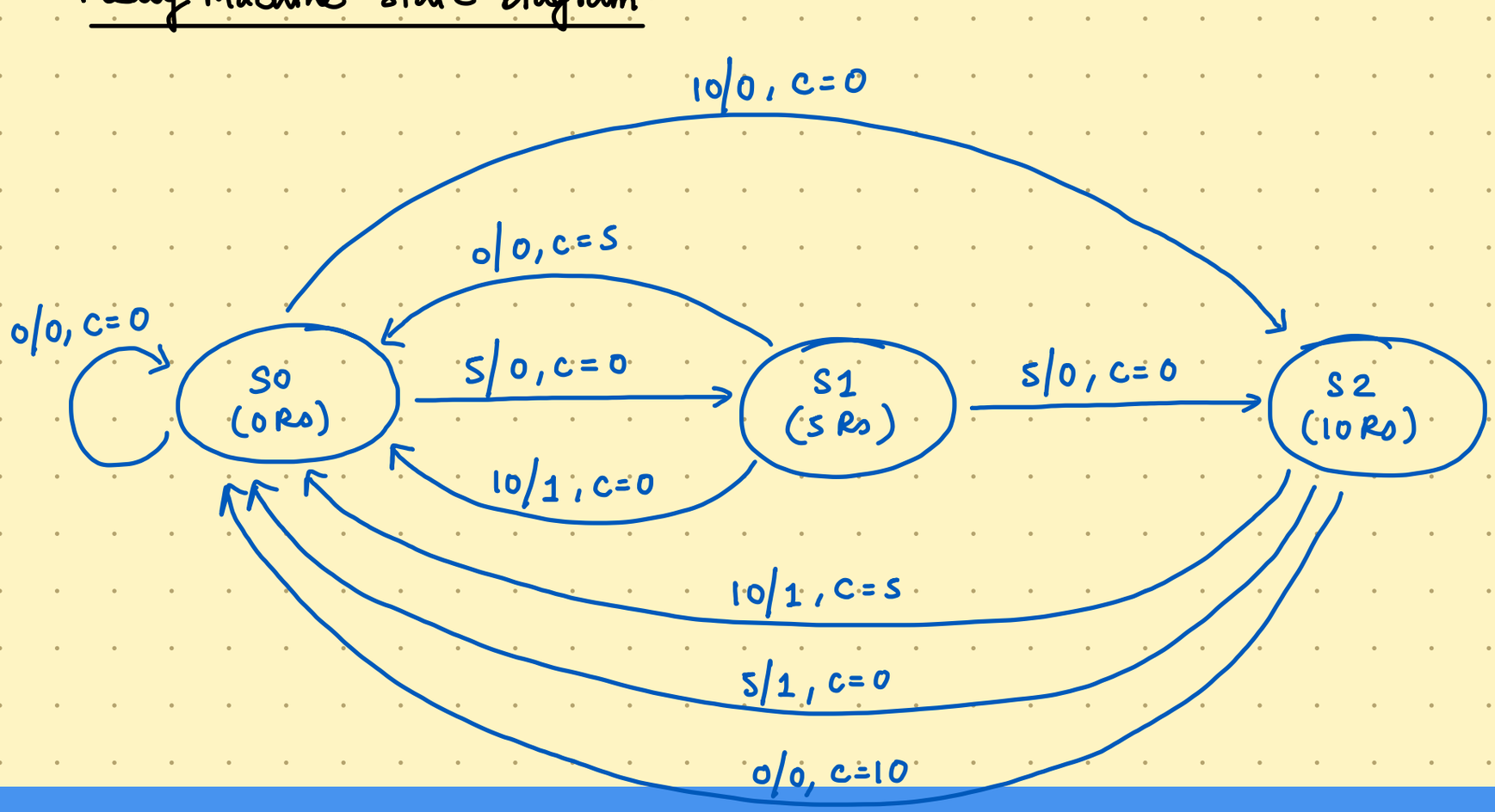
Vending Machine in Verilog (with Change Mechanism) –

⇒ Ideal Vending Machine (return back the change to customer if transaction is not complete after some time).

Some assumptions –

- Vending Machine holds only one Product
- Products cost Rs 15
- Currency denomination accepted are Rs 5 and Rs 10.

Mealy Machine State Diagram –



Verilog Code -

To create a Finite State Machine (FSM) based vending machine

- Accepts coins of 5 (input = 2'b01) 10 (input = 2'b10)
- Dispense a product when exact amount is 15.

① FSM Design theory -

An FSM (Finite State Machine) is a computation model used for designing **sequential logic circuits**. It consists of:

- A **finite number of states** (e.g., ₹0, ₹5, ₹10 collected)
- **Transitions** between states based on inputs
- **Outputs** that depend on the current state and/or input

FSMs are categorized into two types:

Type	Output Depends On
Moore	Only the current state
Mealy	Current state + input

This vending machine acts as a **Mealy machine**, because outputs like `out` (product dispensed) and `change` depend on both the **state** and **current input value**.

② Coin Encoding & Logic -

The machine accepts:

- ₹5 → represented as binary `2'b01`
- ₹10 → represented as binary `2'b10`
- No coin → `2'b00`

Why use 2 bits?

Because 1 bit can only represent two values (0 and 1). We need at least 3 values (₹0, ₹5, ₹10), hence a 2-bit representation.

This encoding allows compact representation of coin input and makes the design easily scalable.

③ State Transition logic theory -

The system has 3 states:

- `s0` : ₹0 collected
- `s1` : ₹5 collected
- `s2` : ₹10 collected

Transitions occur based on inserted coin value:

- From `s0` :
  - ₹5 → go to `s1`
  - ₹10 → go to `s2`
- From `s1` :
  - ₹5 → `s2`
  - ₹10 → `s0` and dispense product
- From `s2` :
  - ₹5 → `s0` and dispense product
  - ₹10 → `s0` and dispense product + return ₹5

This is a classic **accumulator-style FSM**, where the machine accumulates credit (money) in the states and triggers a **transaction** (dispense) when credit ≥ ₹10.

④ Reset Behaviour -

The `reset` signal:

```verilog
if(rst == 1)
```

Resets:

- Both current and next state to 0 (`s0`)
- Clears the `change` signal
- **Asynchronous reset** is implemented because it's checked directly, not via clock

Effectively, it brings the vending machine to its **initial state** regardless of current state or inputs.

⑤ Product Dispensing and Change Logic–

The product price is **₹10**. So when:

- ₹10 is inserted directly (state = `s0`, input = `2'b10`)
  → goes to `s2` (credit = ₹10)
- ₹10 inserted again (state = `s2`, input = `2'b10`)
  → total = ₹20 → product is dispensed (`out=1`), change = ₹5 (`change = 2'b01`)

The output logic is **embedded in the state transition code**:

verilog

```
else if(in == 2'b10)
begin
    n_state = s0;
    out = 1;
    change = 2'b01;
end
```

There is no separate output FSM. This makes the design **compact** but also **less modular**.