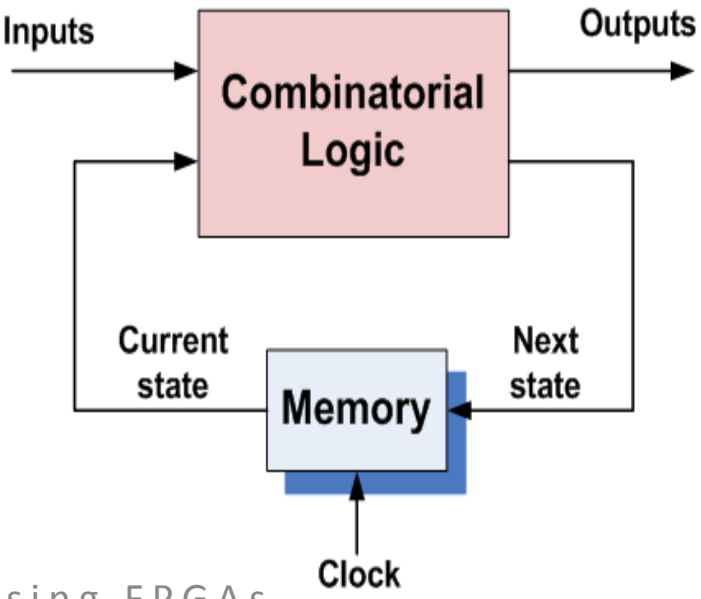
Programmable Electronics using FPGAs

This week

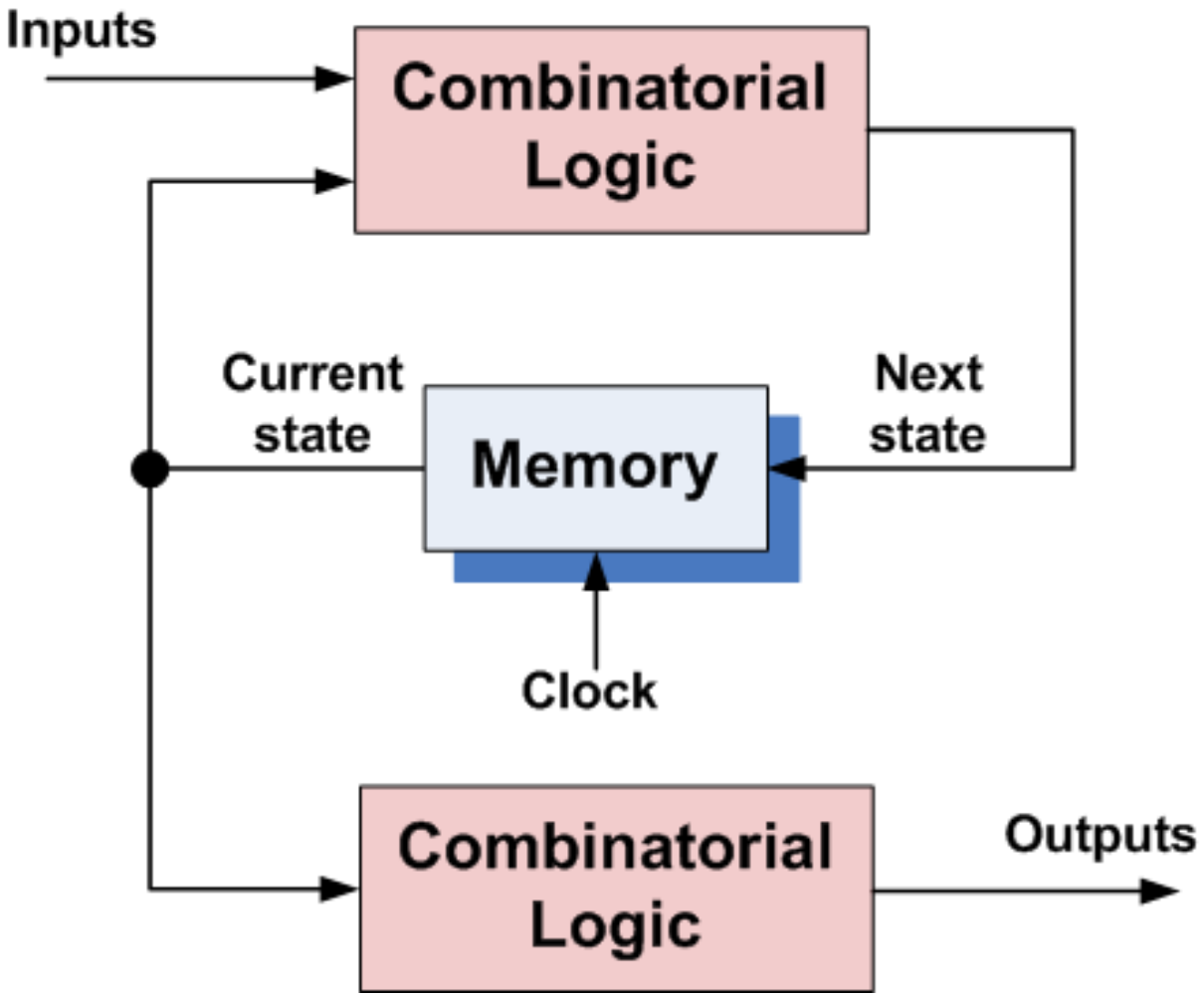
- Recap: Mealy state machine
- Moore state machine
- Vending machine controller
- Assessment 2: Traffic light controller

Recap: Mealy state machine

- Within a Mealy machine:
  - Outputs depend on both current state and inputs
  - May be fewer states
  - It has asynchronous outputs
    - If input glitches, so does output
    - Output available immediately
    - May be unstable output

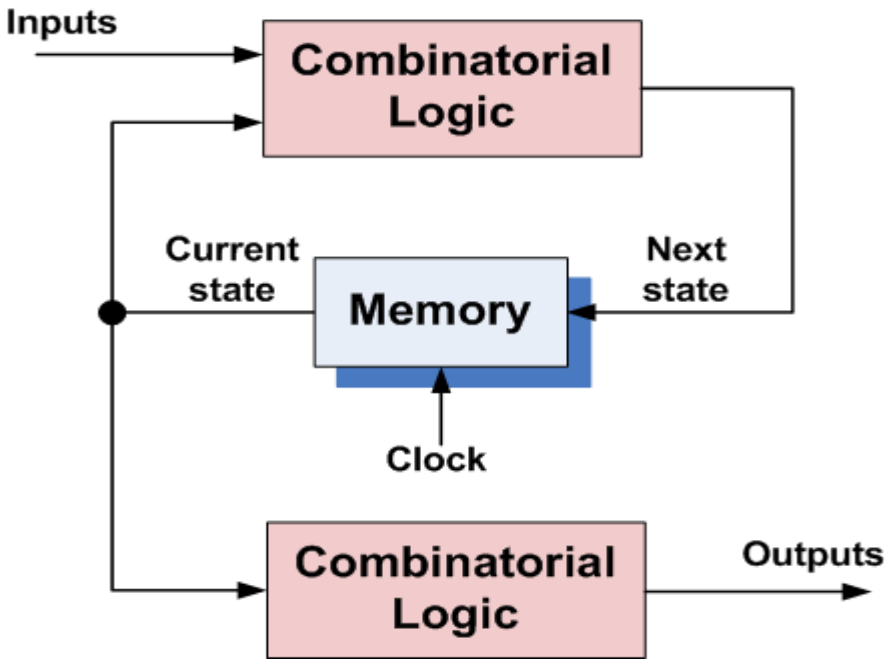


Moore state machine model



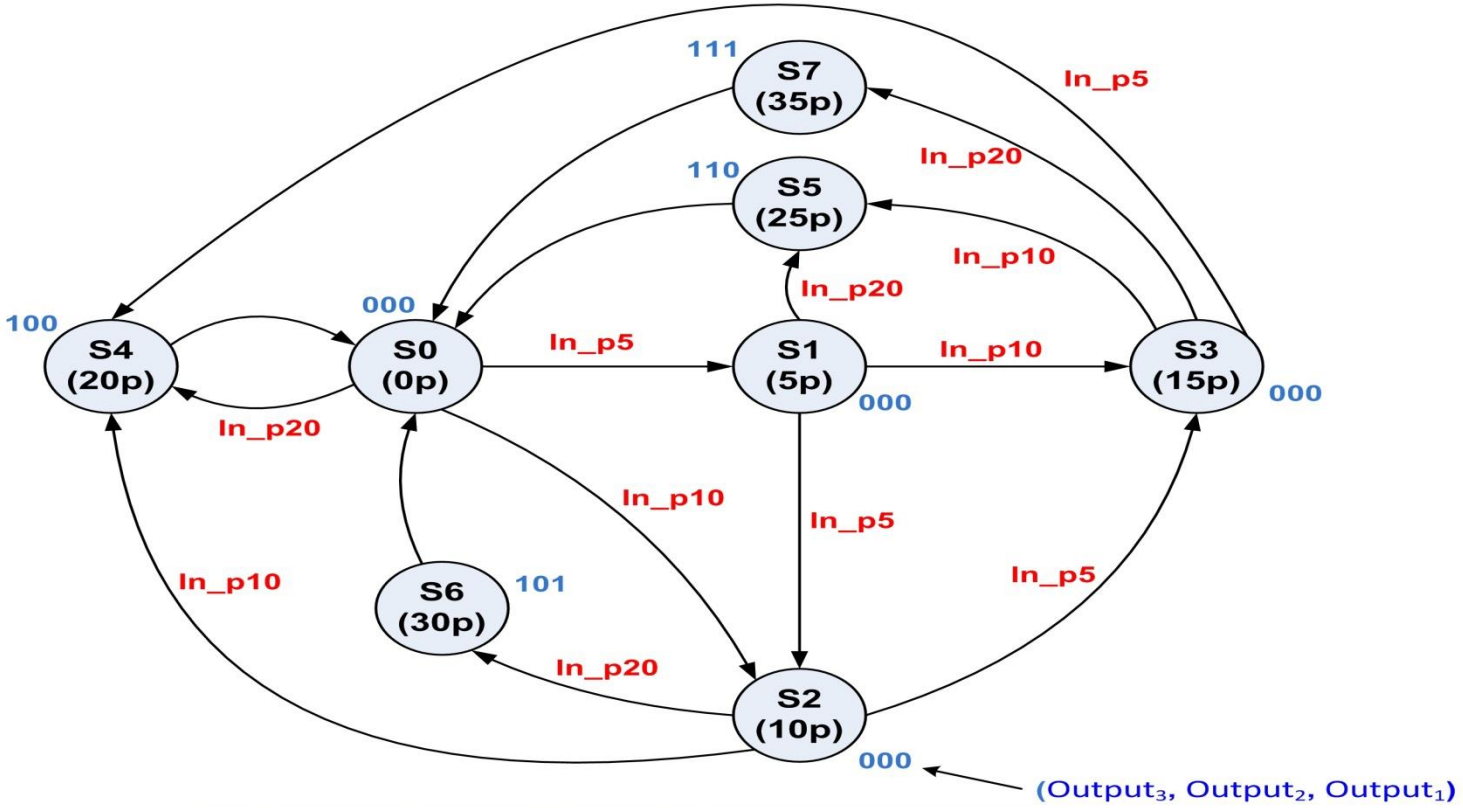
Moore state machine

- In a Moore state machine:
  - Outputs depend on the current state only
  - May be more states
  - It has synchronous outputs
    - No glitch in design
    - Delay of one clock cycle
    - Stable output



Example: Vending machine controller using a Moore machine

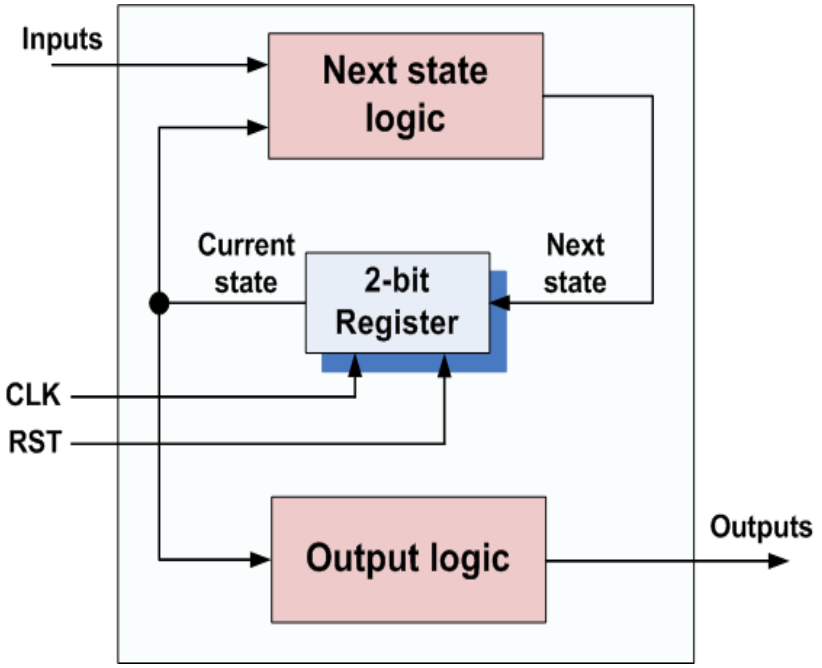
- The state diagram is more complex:

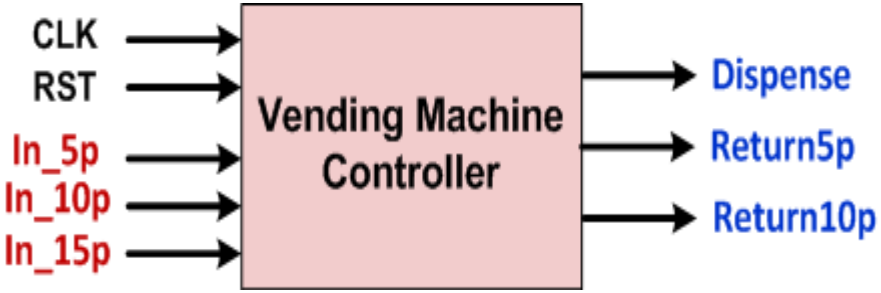


Input = In_p5 | In_p10 | In_p20
$Output_3$ = **1** | **0** = Dispense / Do not dispense product
$Output_2$ = **1** | **0** = Return / Do not return a 5p in change
$Output_1$ = **1** | **0** = Return / Do not return a 10p in change

Example: Vending machine controller using Moore FSM model

- Moore machine consists of the following components:
  - Entity declaration
  - Define state encoding
  - State register using a process
  - Next state logic using a process
  - Output logic using a process





Example: Vending machine controller

- Entity declaration and define state encodings:

```
Entity  VFSM  is
        port  (  CLK, RST                          :  IN     std_logic;
                 In_5p, In_10p, In_20p             :  IN     std_logic;
                 Dispense, Return5p, Return10p  :  OUT  std_logic  );
end VFSM ;

architecture   My_FSM  of   VFSM   is
        type FSM_type  is  ( S0 ,  S1 ,  S2 ,  S3 ,  S4 ,  S5 ,  S6 ,  S7 ) ;
        signal Current_State,  Next_State  :  FSM_type;
begin
```

Example: Vending machine controller

- State register using a process block

```
process (CLK , RST) begin
    if RST = '1' then
        Current_State <= S0;
    elsif rising_edge (CLK) then
        Current_State <= Next_State;
    else
        Current_State <= Current_State;
    end if;
end process;
```

Example: Vending machine controller

- Next state logic using a process block

```
process ( Current_State,  In_5p,  In_10p,  In_20p )  begin
    case   Current_State   is
          when S0   =>   if  In_5p  =  '1'  then  Next_State <=  S1;
                         elsif   In_10p = '1'   then  Next_State <=  S2;
                         elsif   In_20p = '1'   then Next_State <=  S4;
                         else    Next_State <= S0;    end if;
          when S1   => …
          when  S2   => …
          when S3   => …
          when  S4   => Next_State  <=  S0;
          when  S5   => Next_State  <=  S0;
          when  S6   => Next_State  <=  S0;
          when  S7   => Next_State  <=  S0;
          when others => Next_State <= S0;
    end case;
end process;
```

Example: Vending machine controller

- Output logic using a process block

```
process ( Current_State)  begin
    case  Current_State  is
            when S0  =>   Dispense <= '0';  Return5p <= '0';   Return10p <= '0';
            when S1  =>   Dispense <= '0';  Return5p <= '0';   Return10p <= '0';
            when S2  =>  Dispense <= '0';  Return5p <= '0';   Return10p <= '0';
            when S3  =>  Dispense <= '0';  Return5p <= '0';   Return10p <= '0';
            when S4  =>  Dispense <= '1';  Return5p <= '0';   Return10p <= '0';
            when S5  =>  Dispense <= '1';  Return5p <= '1';   Return10p <= '0';
            when S6  =>  Dispense <= '1';  Return5p <= '0';   Return10p <= '1';
            when S7  =>  Dispense <= '1';  Return5p <= '1';   Return10p <= '1';
            when others => Dispense <= '0';  Return5p <= '0';   Return10p <= '0';
    end case;
end process;
end My_FSM ;
```