- **General Finite State Machine (FSM) design**
    - FSM = sequential logic circuit which can be implemented in a fixed number of states
    - basically extend design technique learned for counters

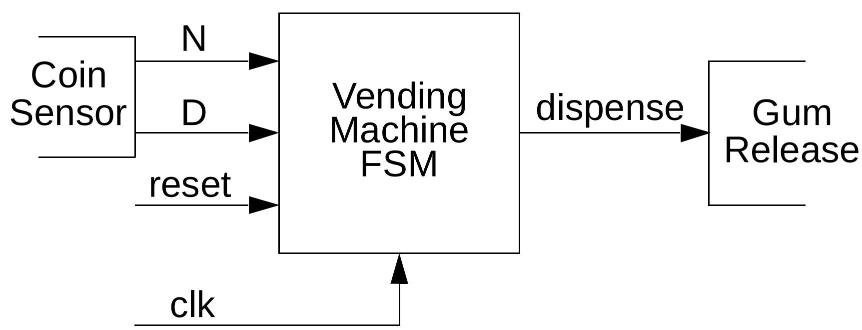
- **Basic design approach**

    1. understand the problem
        - make assumptions to complete design specification
        - identify inputs and outputs
        - draw a block diagram of FSM
        - enumerate possible inputs sequences and system states
    2. obtain abstract representation of FSM
        - draw state (transition) diagram
        - or alternative state machine representation
    3. perform state minimization (*new step*)
    4. perform state assignment
        - encode states
        - build state transition table
    5. choose FF type
        - remap next state into required FF inputs
    6. implement
        - simplify next state and output logic equations
        - wire the circuit together

- **Design example: simple vending machine**

  Step 1: understand the problem

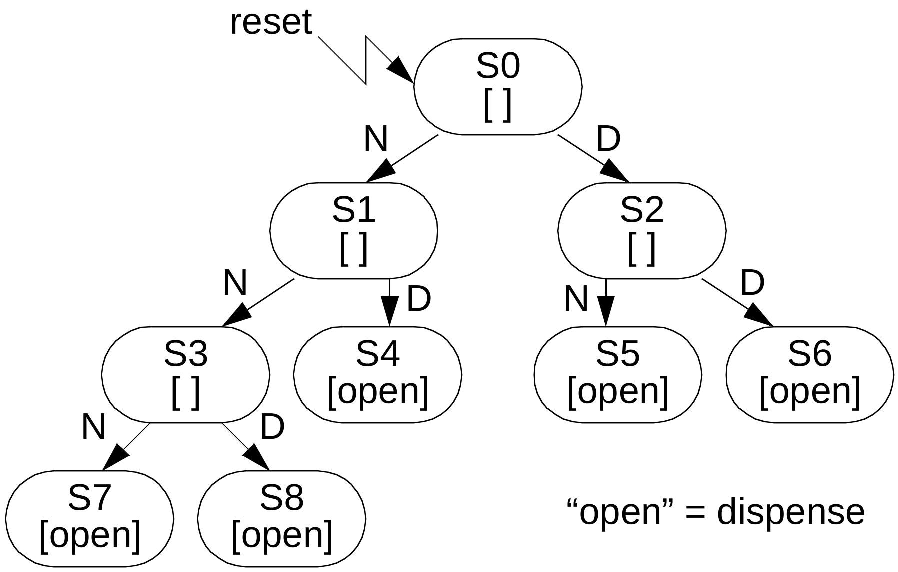
  - problem description:
    - vending machine dispenses gum given 15¢ or more
    - machine accepts nickels or dimes (sensor determines coin)
    - machine provides no change
  - assumptions:
    - other coins automatically returned
    - external reset applied after gum dispensed
  - inputs/outputs:
    - nickel inserted; dime inserted; reset; clock
    - dispense gum
  - block diagram:



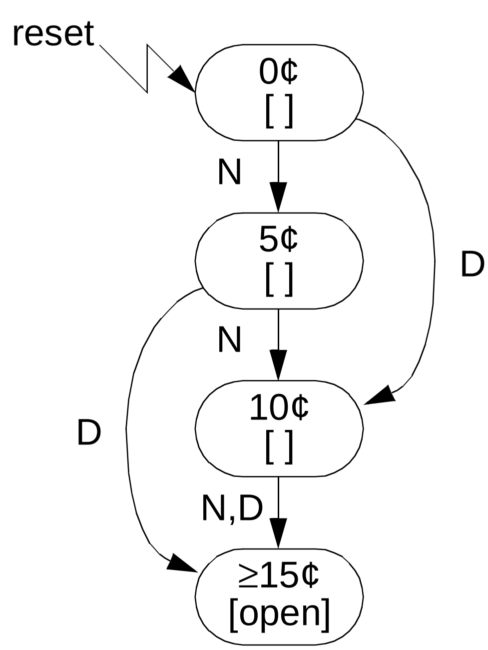

  - possible input sequences:
    - N, N, N
    - N, N, D
    - N, D
    - D, N
    - D, D
  - output: asserted only after reaching 15¢ or greater

Step 2: obtain abstract representation of FSM

- state diagram — based on tree of input options




- minimized state diagram (intuitive approach to Step 3)




- alternative state machine representations
  - algorithm state machines – like flowcharts w/ rigorous timing
  - hardware description languages – like HLL with explicit ||-ism

Step 4: perform state assignment

| Present State | Inputs D | N | Next State | Output Dispense |
|---|---|---|---|---|
| 0¢ | 0 | 0 | 0¢ | 0 |
| | 0 | 1 | 5¢ | 0 |
| | 1 | 0 | 10¢ | 0 |
| | 1 | 1 | X | X |
| 5¢ | 0 | 0 | 5¢ | 0 |
| | 0 | 1 | 10¢ | 0 |
| | 1 | 0 | ≥15¢ | 0 |
| | 1 | 1 | X | X |
| 10¢ | 0 | 0 | 10¢ | 0 |
| | 0 | 1 | ≥15¢ | 0 |
| | 1 | 0 | ≥15¢ | 0 |
| | 1 | 1 | X | X |
| ≥15¢ | X | X | ≥15¢ | 1 |

- 4 states requires 2 bits

  0¢ ⇒ 00

  5¢ ⇒ 01

  10¢ ⇒ 10

  ≥15¢ ⇒11

Step 5: choose FF type

- choose J-K FF ⇒ remap next state into FF inputs

| Present State | | Inputs | | Next State | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Q1 | Q0 | D | N | $Q1^+$ | $Q0^+$ | J1 | K1 | J0 | K0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | X | 0 | X |
| | | 0 | 1 | 0 | 1 | 0 | X | 1 | X |
| | | 1 | 0 | 1 | 0 | 1 | X | 0 | X |
| | | 1 | 1 | X | X | X | X | X | X |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | X | X | 1 |
| | | 0 | 1 | 1 | 0 | 1 | X | X | 1 |
| | | 1 | 0 | 1 | 1 | 1 | X | X | 0 |
| | | 1 | 1 | X | X | X | X | X | X |
| 1 | 0 | 0 | 0 | 1 | 0 | X | 0 | 0 | X |
| | | 0 | 1 | 1 | 1 | X | 0 | 1 | X |
| | | 1 | 0 | 1 | 1 | X | 0 | 1 | X |
| | | 1 | 1 | X | X | X | X | X | X |
| 1 | 1 | X | X | 1 | 1 | X | 0 | X | 0 |

Step 6: implement

- simplify logic equations for FF inputs and circuit output

| D N \ Q1Q0 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | X | X |
| 01 | 0 | 1 | X | X |
| 11 | X | X | X | X |
| 10 | 1 | 1 | X | X |

$J1 = D + Q0 \cdot N$

| D N \ Q1Q0 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | X | X | 0 | 0 |
| 01 | X | X | 0 | 0 |
| 11 | X | X | X | X |
| 10 | X | X | 0 | 0 |

$K1 = 0$

| D N \ Q1Q0 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | X | X | 0 |
| 01 | 1 | X | X | 1 |
| 11 | X | X | X | X |
| 10 | 0 | X | X | 1 |

$J0 = N + Q1 \cdot D$

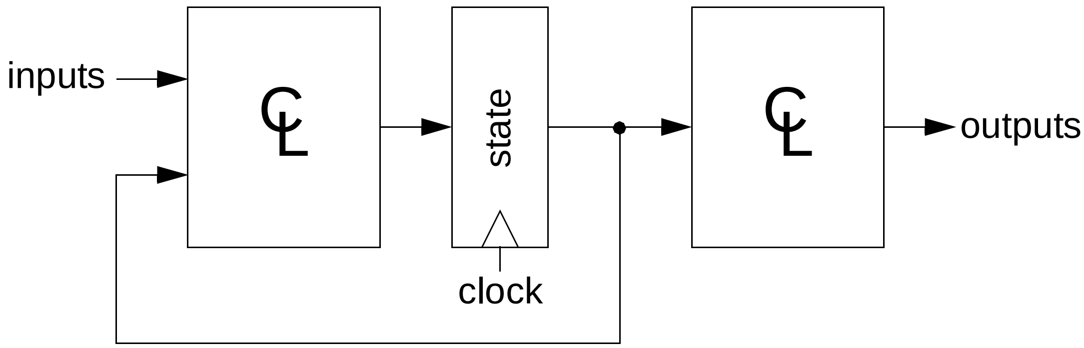
| D N \ Q1Q0 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | X | 0 | 0 | X |
| 01 | X | 1 | 0 | X |
| 11 | X | X | X | X |
| 10 | X | 0 | 0 | X |

$K0 = \overline{Q1} \cdot N$

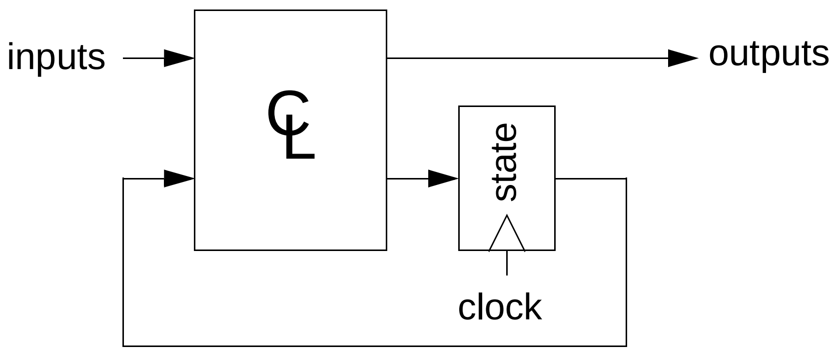
| D N \ Q1Q0 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 1 | 0 |
| 01 | 0 | 0 | 1 | 0 |
| 11 | X | X | X | X |
| 10 | 0 | 0 | 1 | 0 |

$\text{Dispense} = Q1 \cdot Q0$

- **Moore vs. Mealy machines**
  - block diagrams

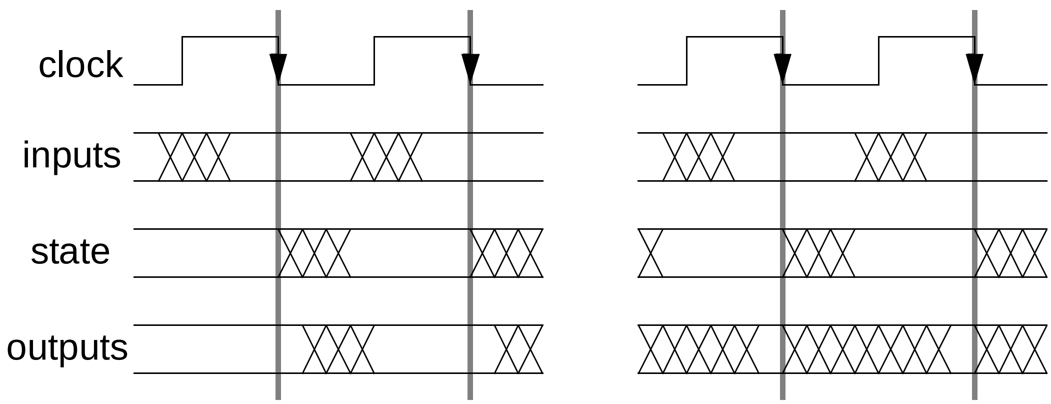



*Moore machine*



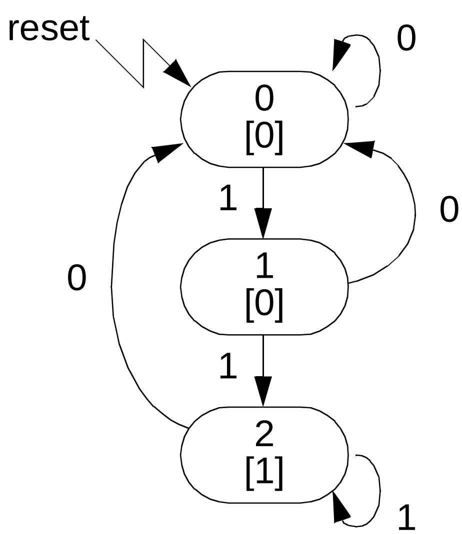

*Mealy machine*

  - timing of input, state, and output changes

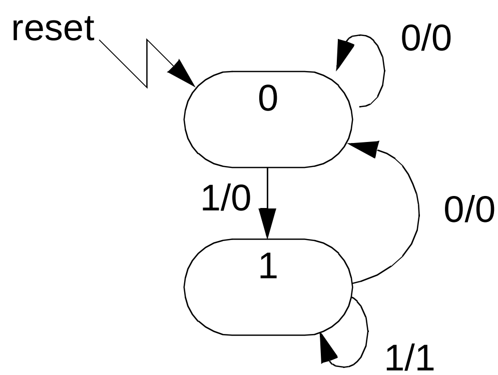



*Moore machine timing* *Mealy machine timing*

- design example:
    - function: assert output if 2 or more 1’s in a row
    - state diagram:




*Moore machine*




*Mealy machine*

- advantages/disadvantages
    - Mealy often has fewer states than Moore machine since it associates outputs with transitions
    - Mealy machine can fall victim to glitches since outputs are asynchronous