Synthesis of Finite-State-Machines

- Model a description of a sequential system
- Finite-State Transition System: FSM
- The system moves from state-to-state
- State transition are defined by the inputs
- The machine outputs some signals
- Outputs depend upon the present state, or on present states and current inputs
- Examples: Vending machine controller, traffic light controllers, serial adders, string matching circuits, etc.
Vending machine controller

- Takes in Nickels (N) and Dimes (D)
- Cost of item: (say) 15 cents
- If 15 cents deposited, output coke
- State depends upon the amount of money deposited
  - Think in terms of “state computation”
  - State transitions correspond to “next computation”
  - Output = 1 if enough money deposited.
What is FSM Synthesis?

- Design a FSM based on the given spec.
- Is your FSM complete? Minimal?
- Does your FSM have redundancies? Two or more states performing the “same computation”?
- FSM minimization - exact minimization algorithms
- How do you realize the states? Via State Registers (DFFs)!
- The State Encoding Problem!
- Implement the Logic! But how?
- How fast can you clock your circuit?
## State Table

### Table 1: State Transition Table

<table>
<thead>
<tr>
<th>P.S. ((y_1, y_0))</th>
<th>(\overline{ND})</th>
<th>(\overline{ND})</th>
<th>(\overline{ND})</th>
<th>(\overline{ND})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A = 00)</td>
<td>(A = 00)</td>
<td>(C = 11)</td>
<td>(B = 01)</td>
<td>(A = 00)</td>
</tr>
<tr>
<td>(B = 01)</td>
<td>(B = 01)</td>
<td>(D = 10)</td>
<td>(C = 11)</td>
<td>(B = 01)</td>
</tr>
<tr>
<td>(C = 11)</td>
<td>(C = 11)</td>
<td>(D = 10)</td>
<td>(D = 10)</td>
<td>(C = 11)</td>
</tr>
<tr>
<td>(D = 10)</td>
<td>(A = 00)</td>
<td>(A = 00)</td>
<td>(A = 00)</td>
<td>(A = 00)</td>
</tr>
</tbody>
</table>