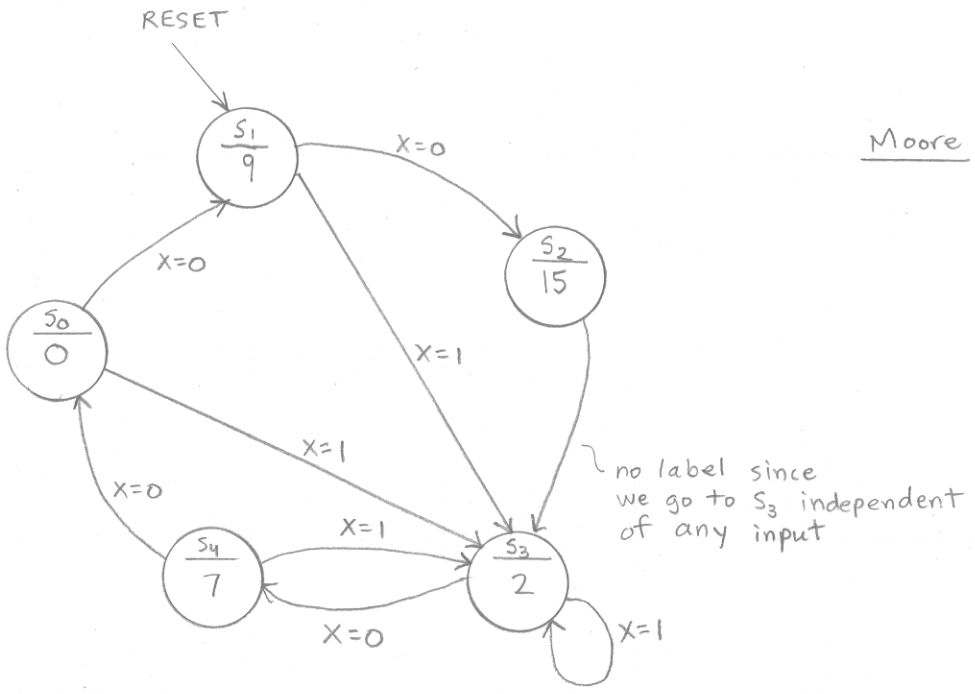
Finite State Machine Design

Design Process

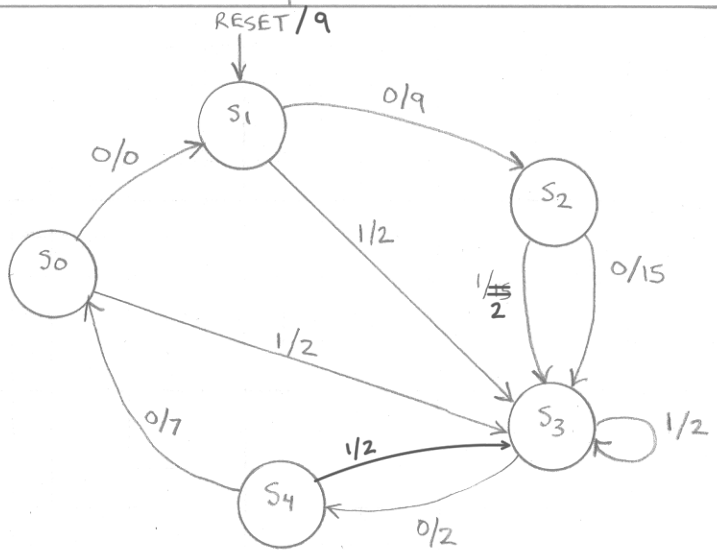
1) Figure out necessary states from problem specification (e.g., a counter from 0 to 10 needs 11 states)
2) Draw state graph
   - Enumerate states
   - capture transitions between states ~~and inputs~~
3) Add outputs
   - Moore machine: outputs function of present state only
   - Mealy machine: outputs are function of present state and inputs
4) Figure out number of flip-flops required for implementation (minimum # is $\lceil \log_2 (\text{\# of states}) \rceil$)
5) Assign states (assignment may affect the ~~number of flip-flops~~ amount of logic needed)
6) Construct a state table
   - Develop equations for next states
7) Figure out outputs
   - Develop equations for outputs (different for Moore vs. Mealy)

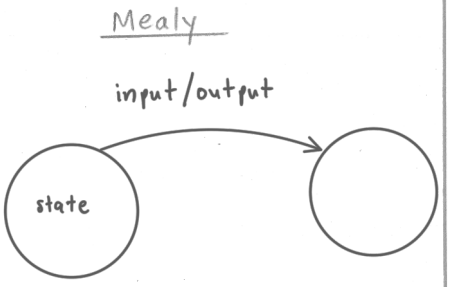
Ex: Count sequence 0, 9, 15, 2, 7 <repeat>, when X=0
If input X=1, output 2 and start count from there when X=0.
Reset input: go to 9 in the sequence

5 values in sequence ⇒ 5 states (S_0 - S_4)



Moore





Present State	Next State X=0	Next State X=1	Moore Output	Mealy Output X=0	Mealy Output X=1
S_0	S_1	S_3	0	0	2
S_1	S_2	S_3	9	9	2
S_2	S_3	S_3	15	15	2
S_3	S_4	S_3	2	2	2
S_4	S_0	S_3	7	7	2

State Assignments

1.) Assign in order of normal operating sequence (easy to think about)

or

2.) Assign at random

or

3.) Assign to minimize logic/circuit implementation

or

4.) One-hot encoding

- Use one flip-flop for each state, so an N-state FSM requires N flip-flops (possibly much larger than $\log_2 N$)
- Reduces complexity of logic for computing next states and outputs
- Use when flip-flops are relatively cheap (e.g., FPGAs, CPLDs, etc.)

or 5.) Gray code (reduces power by reducing number of bit transitions)

Ex: Continue previous example; sequential state assignment

	Present State $Q_2 Q_1 Q_0$	Next State X = 0 $Q_2^+ Q_1^+ Q_0^+$	Next State X = 1 $Q_2^+ Q_1^+ Q_0^+$
S_0	0 0 0	0 0 1	0 1 1
S_1	0 0 1	0 1 0	0 1 1
S_2	0 1 0	0 1 1	0 1 1
S_3	0 1 1	1 0 0	0 1 1
S_4	1 0 0	0 0 0	0 1 1

Equations:

$$Q_0^+ = X + X' Q_2' Q_0'$$

$$Q_2^+ = X' Q_1 Q_0$$

$$Q_1^+ = X + X' Q_2' Q_1' Q_0 + X' Q_2' Q_1 Q_0'$$

Now try one-hot state encoding → need 5 flip-flops

	Present State $Q_4 Q_3 Q_2 Q_1 Q_0$	Next State X = 0	Next State X = 1
S_0	0 0 0 0 1	00010	01000
S_1	0 0 0 1 0	00100	01000
S_2	0 0 1 0 0	01000	01000
S_3	0 1 0 0 0	10000	01000
S_4	1 0 0 0 0	00001	01000

Equations:

$$Q_0^+ = Q_4 \cdot X'$$

$$Q_1^+ = Q_0 \cdot X'$$

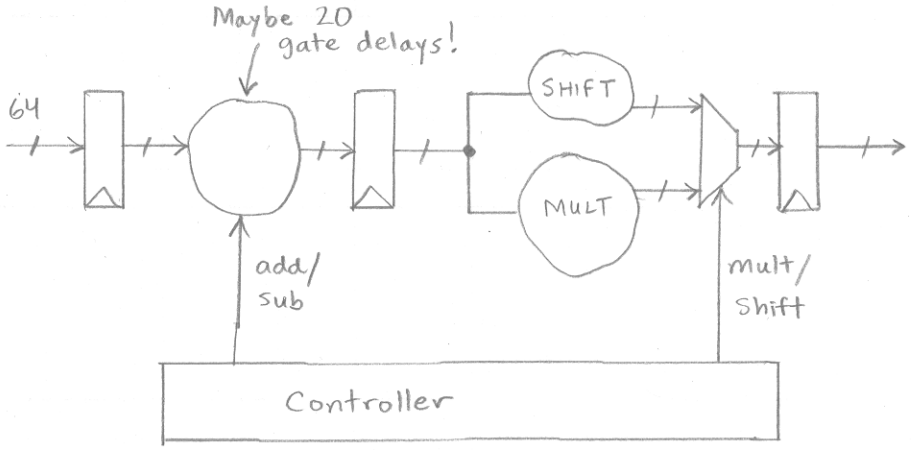
$$Q_2^+ = Q_1 \cdot X'$$

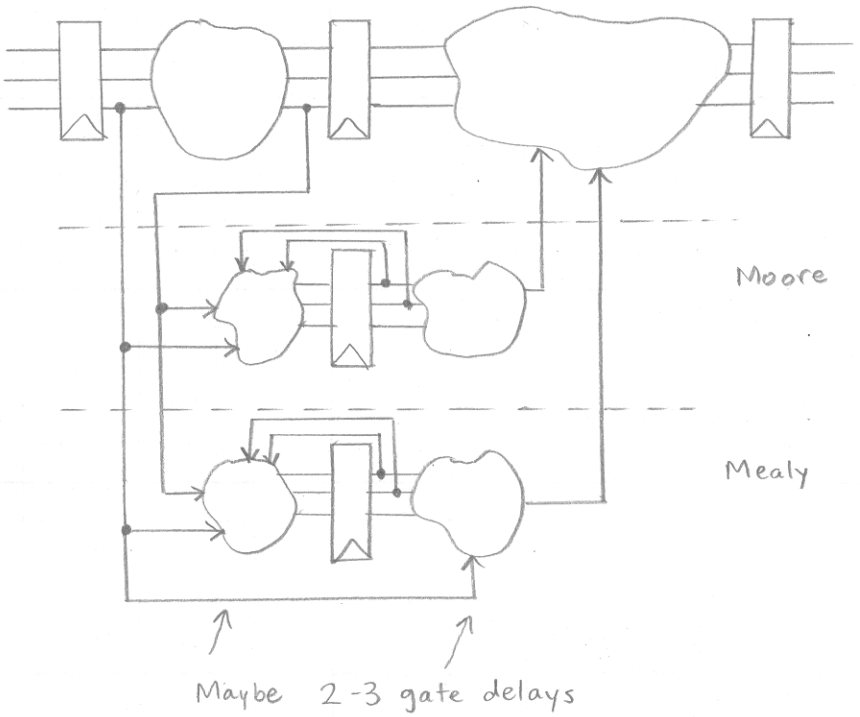
$$Q_3^+ = Q_2 \cdot X' + X$$

$$Q_4^+ = Q_3 \cdot X'$$

Speed of Moore vs. Speed of Mealy

Consider a microprocessor data pipeline → the controller can be an FSM





	Longest Path
Older computers	~50 gate delays
fastest current computers	~15
Special purpose processors	~10

- Easier to design a long path with a Mealy machine
- Purely combinational path from input to output yields possibly faster decisions, but potential for hazards

Comparing Moore vs. Mealy outputs: