CS/ECE 250: Computer Architecture

Logic Design: Tristate Buffers, Finite State Machines

Benjamin Lee

Slides are derived from work by
Alvin Lebeck, Drew Hilton, Amir Roth, Dan Sorin

Admin

- **Homework #3 assigned**

- **Readings**
  - **Pragmatic Logic**
  - **Combinational Circuits Ch 4.1-4.2, Ch 5.3**
  - **Sequential Circuits Ch 6**
  - **Also if you want appendix C of H&P**

Finite State Machine

- **S ={** $s_0, s_1, . . . s_{n-1}$} is a finite set of states.
- **I =** { $i_0, i_1, . . . i_{k-, 1}$} is a finite set of input values.
- **O=** { $o_0, o_1, . . . o_{m-1}$} is a finite set output values.

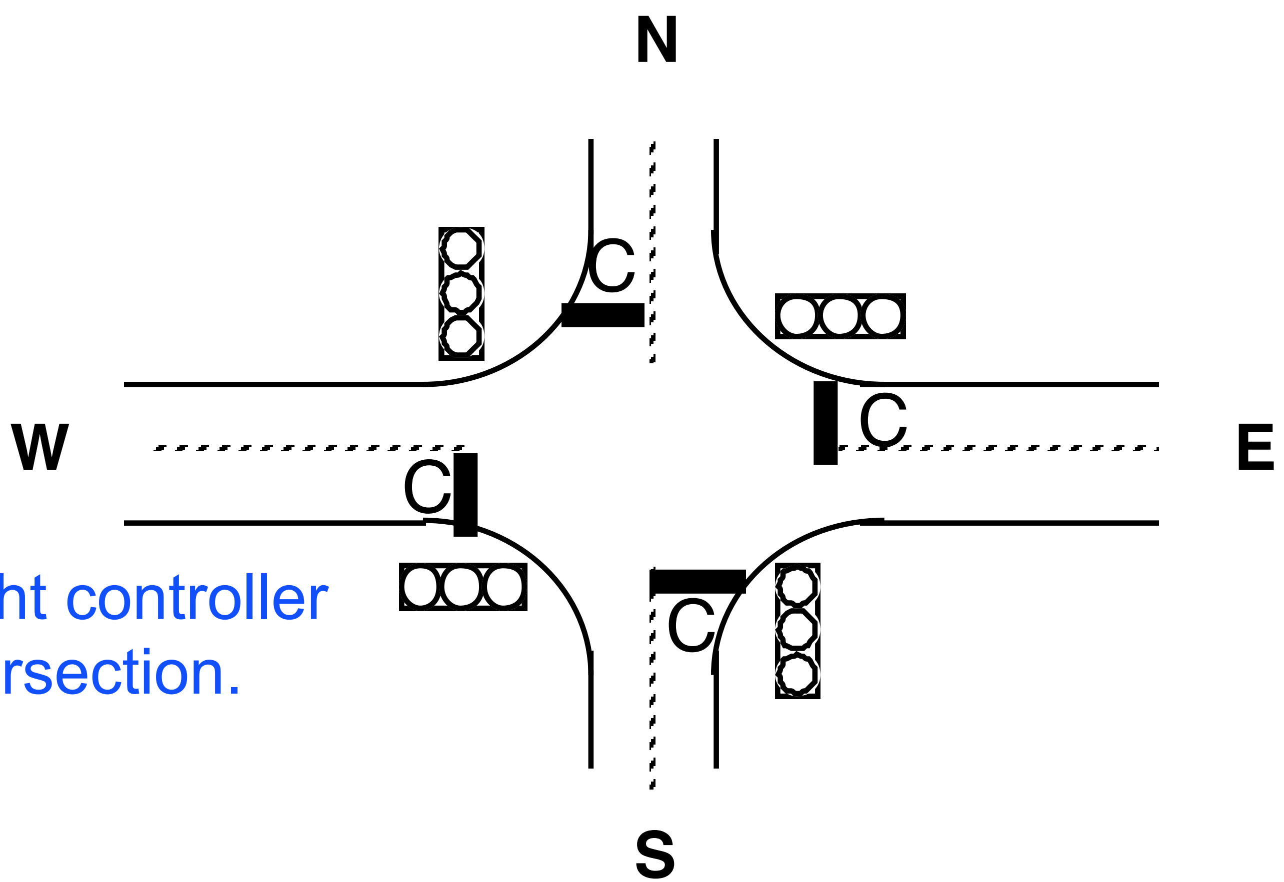
Definition: A finite state machine is a function F:(**S** x **I**) -> (**S** x **O**) that gets a sequence of input values $I_k \in I$, k = 0,1,2 , • • • and it produces a sequence of output values $O_k \in O$, k= 1,2, • • • such that:

$$\mathbf{F}(s_k, i_k) = (s_{k+1}, o_{k+1}) \quad K=0, 1, 2, \cdots$$

Finite State Machine

- **Finite State Machine is:**
  - **A machine with a finite number of possible states.**
  - **A machine with a finite number of possible Inputs.**
  - **A machine with a finite number of possible different outputs.**
  - **At each period (clock cycle) the machine receives an input and it produces an output.**
  - **The output is a function of the input and current state.**
  - **After each period the machine changes state.**
  - **The new state is a function of the input and current state.**

Example: Traffic Light Controller



Traffic light controller
at an intersection.

Finite State Machine (cont.)

- Example: Traffic lights controller:
  - There are four states:
    - NG: Green light in the north-south direction.
    - NY: Yellow light in the north-south direction.
    - EG: Green light at the East-West direction.
    - EY: Yellow light at the East-West direction.
  - There are four outputs:
    - (G;R): North-South green light, East-West red light
    - (Y;R): North-South yellow light, East West red light
    - (R;Y): North-South red light, East-West yellow light
    - (R;G): North-South red light, East-West green light
  - There are four input values:
    - (c, c): Car at the North-South, Car at East-West
    - (c, nc) Car at North-South, No-car at East-West
    - (nc, c): No-car at North-South, Car at East-West
    - (nc, nc): No-car at North-South, No-car at East-West

FSM Example: Traffic Light

State Transitions:

State	Input	Next-State	Output
NG	(-;NC)	NG	(G;R)
NG	(-;C)	NY	(G;R)
NY	-	EG	(Y;R)
EG	(NC;-)	EG	(R;G)
EG	(C;-)	EY	(R;G)
EY	-	NG	(R;Y)

– means don't care

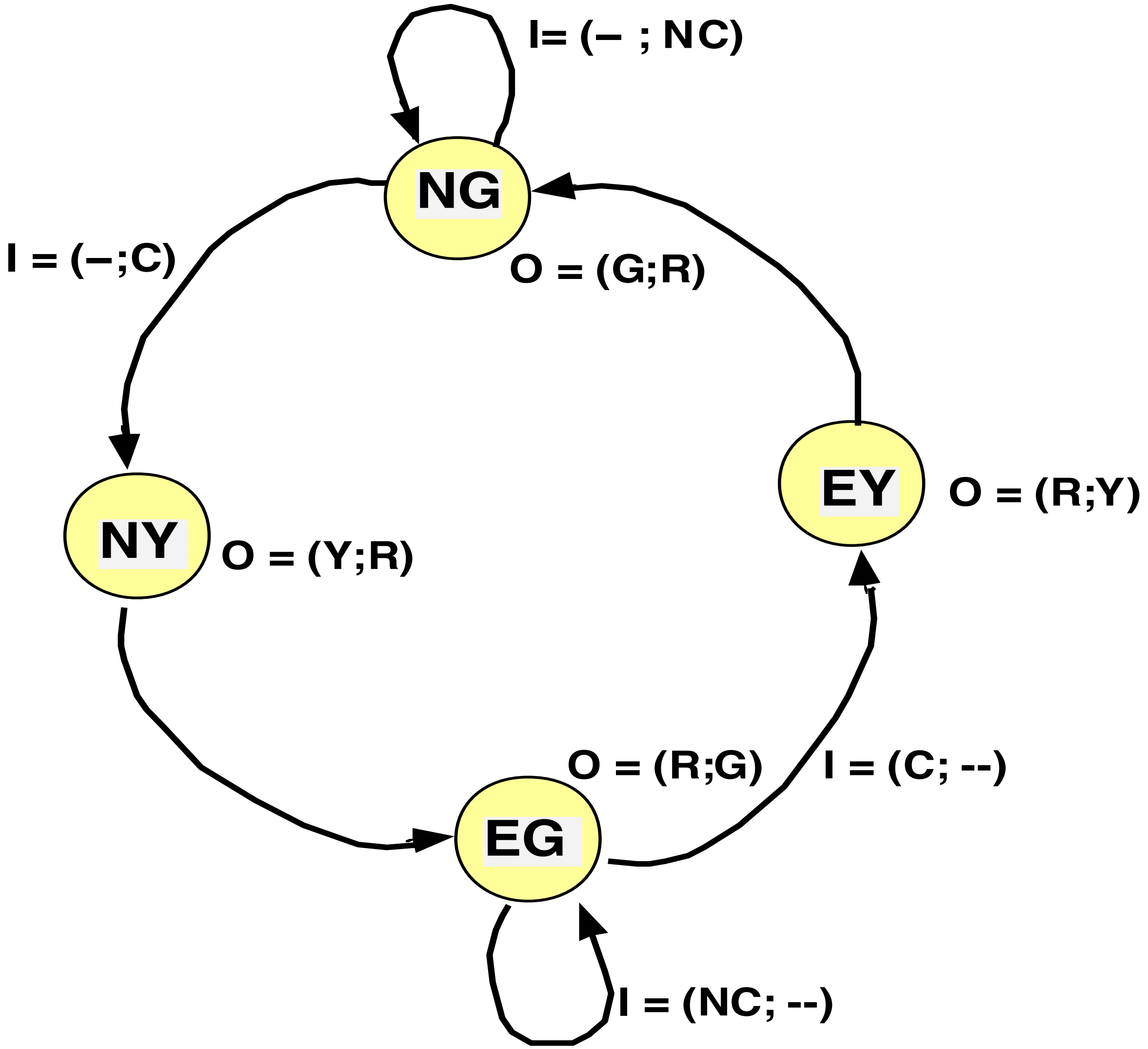
Format
(North/South; East/West)

Finite State Machine (cont.)

- Finite State Machines can be represented by a graph.
- The graph is called a state diagram.
- The states are the nodes in the graph.

- The directed edges in the graph represent state transitions.
- Each directed edge is labeled with the inputs that cause the transition
- Nodes are labeled with the outputs.

FSM State Diagram
Example: Traffic light Controller



State Coding

State	Code
NG	00
NY	01
EG	10
EY	11

Enumerate States

Input	Code
(C;C)	11
(C;NC)	10
(NC;C)	01
(NC;NC)	00

One bit for each Input

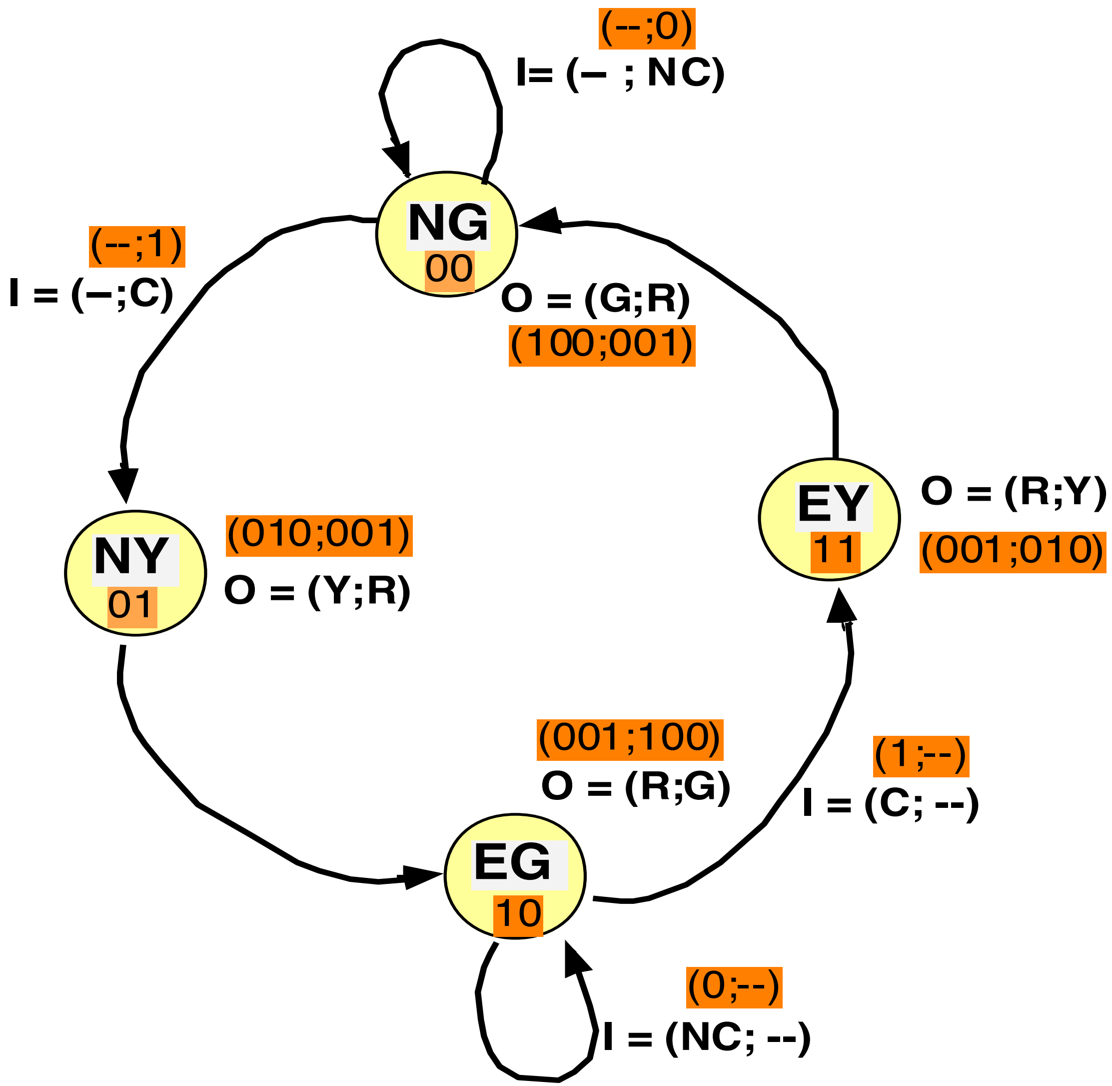
Input is either true or false

Output	Code
(R;G)	001100
(G;R)	100001
(Y;R)	010001
(R;Y)	001010

One bit per color for each light GYRGYR

(North; East)

Coded State Diagram



Example: Traffic Light Controller

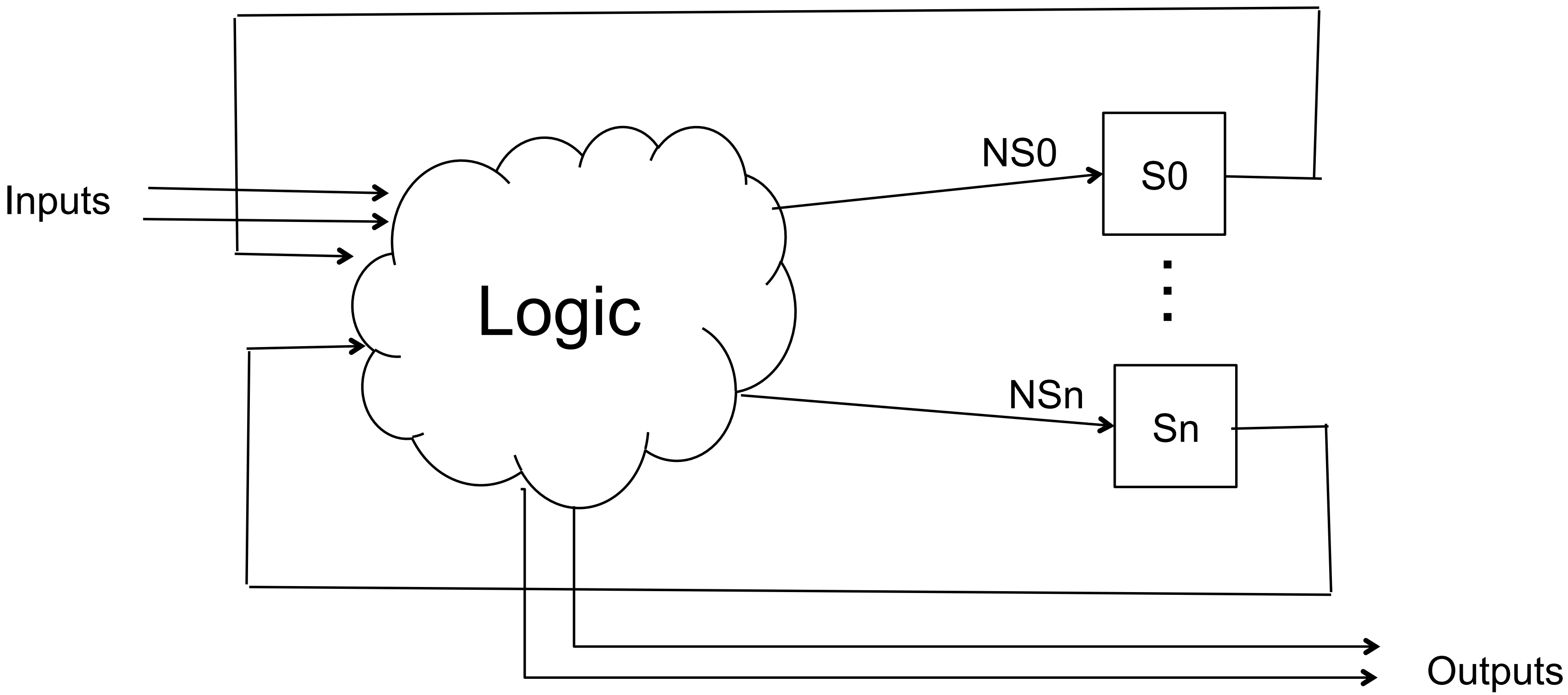
S = State, bits are S0 and S1
NS = Next State, bits are NS0 and NS1

IN 01	S 01	NS 01	OUT 012345
0-	00	00	100001
1-	00	01	100001
--	01	10	010001
-0	10	10	001100
-1	10	11	001100
--	11	00	001010

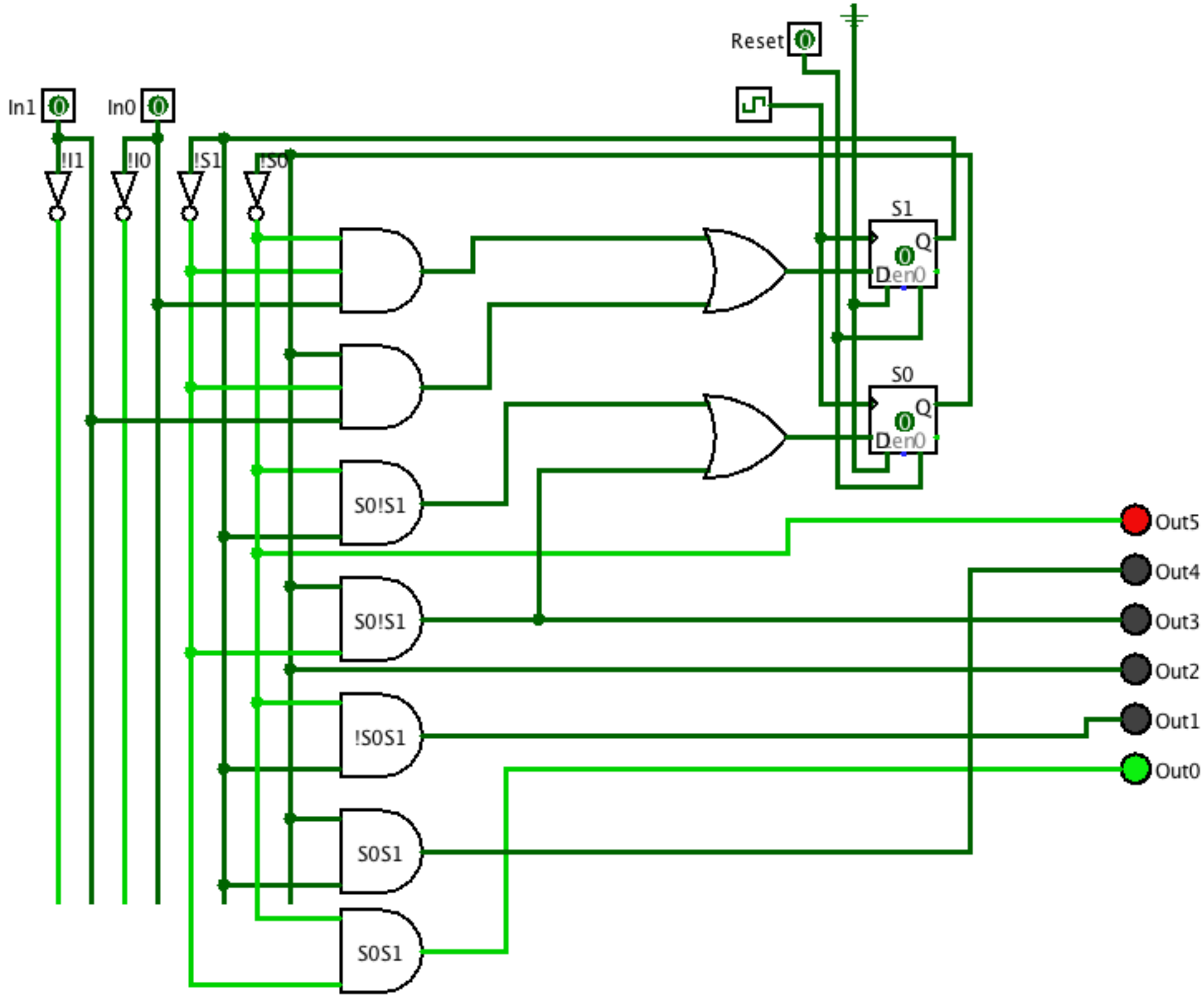
```
NS1  = S0'*S1'*I0+S0*S1'*I1
     = S1'*(S0'I0+S0*I1)
NS0  = S0'*S1+S0*S1'*I1'+S0*S1'*I1
     = S0'*S1+S0*S1'
OUT0 = S0'*S1'
OUT1 = S0'*S1
OUT2 = S0*S1'+S0*S1= S0
OUT3 = S0*S1'
OUT4 = S0*S1
OUT5 = S0'*S1'+S0'*S1= S0'
```

FSM Implementation

- State is stored in D-Flip Flop
- Next State and Output are computed using combinational logic



Traffic Controller FSM implementation



General Method for FSM design

- **Determine the problem:**
  1. **Draw the state diagram,**
  2. **Write the truth table,**
  3. **Write sum-of-products equations**
  4. **Implement in Logic**

A Simple Arrow FSM

- **Consider those flashing arrow signs**
- **No light, one arrow, two arrows, three arrows**
  - **> >> >>>**
- **Let’s design the FSM to control this sign**

Pattern Recognizer

- **A pattern recognizer examines a sequence of inputs to detect when it sees the pattern 101. When it sees this pattern its output is 1 forever.**

- **Let's design the FSM**

Summary

- Can layout logic to compute things
  - Add, subtract,…
- Now can store things
  - D flip-flops
  - Registers
- Also understand clocks
- Can build a finite state machine to control things.

Just about ready to make a processor datapath!