Finite State Machines

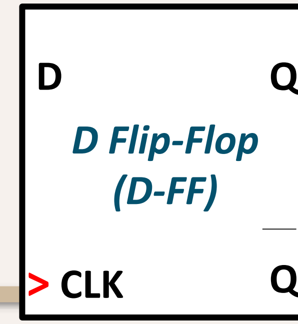
CS 64: Computer Organization and Design Logic
Lecture #16

Ziad Matni
Dept. of Computer Science, UCSB

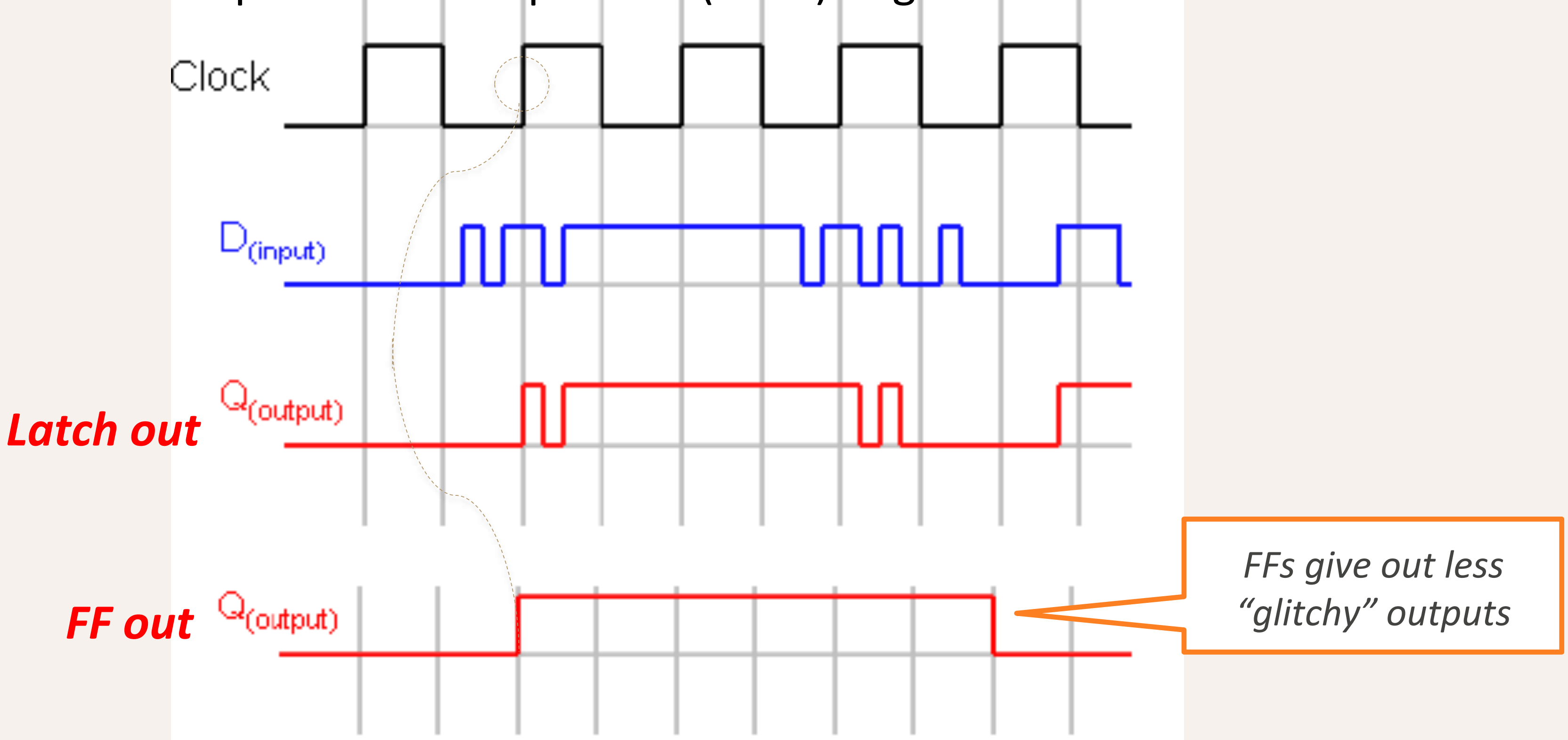
Lecture Outline

- Review of Latches vs. FFs
- Finite State Machines
  - Moore vs. Mealy types
  - State Diagrams
  - "One Hot" Method

Latches vs. FFs



- Latches capture data on an **entire 1 or 0 level of the clock**
- FFs capture data on the **edge of the clock**
  - This example shows the positive (0→1) edge used



If a combinational logic circuit is an implementation of a ***Boolean function***,

then a sequential logic circuit can be considered an implementation of a ***finite state machine***.

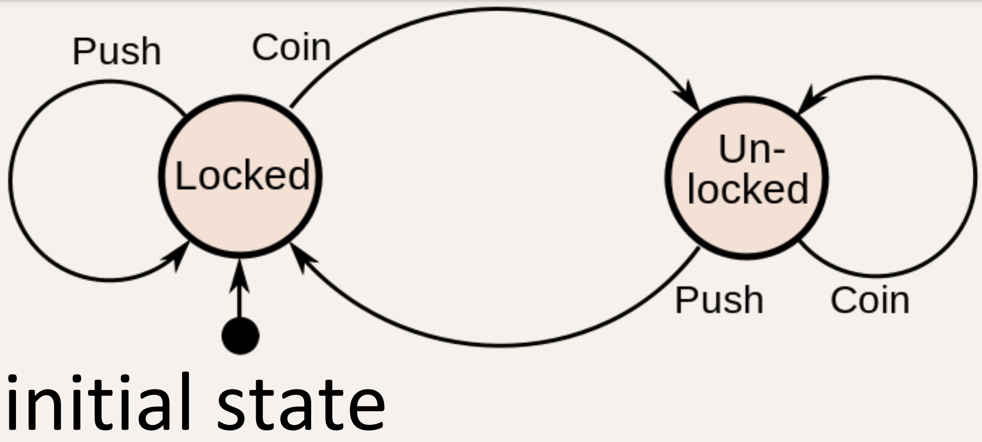
Finite State Machines (FSM)

- An **abstract machine** that can be in **exactly one of a finite number of states at any given time**
  - It's a very simple model of a computational machine, unlike Pushdown Automatons and Turing Machines
    - You'll discover these in other CS upper-div classes

- The FSM can change from one state to another in **response to some *<u>external inputs</u>***
- The change from one state to another is called a ***<u>transition</u>***.

- An FSM is defined by a list of its states, its initial state, and the conditions for each transition.

Example of a Simple FSM: The Turnstile

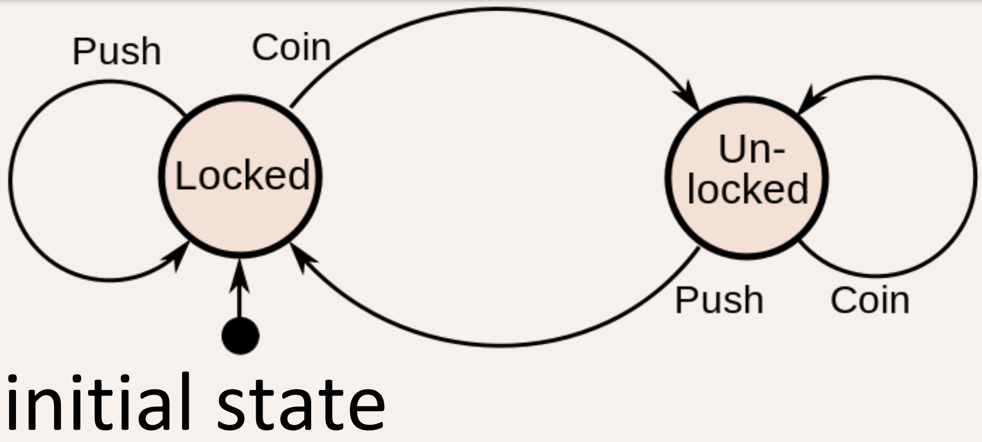


State Transition Table

Current State	Input	Next State	Output
Locked	Coin	Unlocked	Unlocks the turnstile so that the customer can push through.

Source: Wikipedia

Example of a Simple FSM: The Turnstile

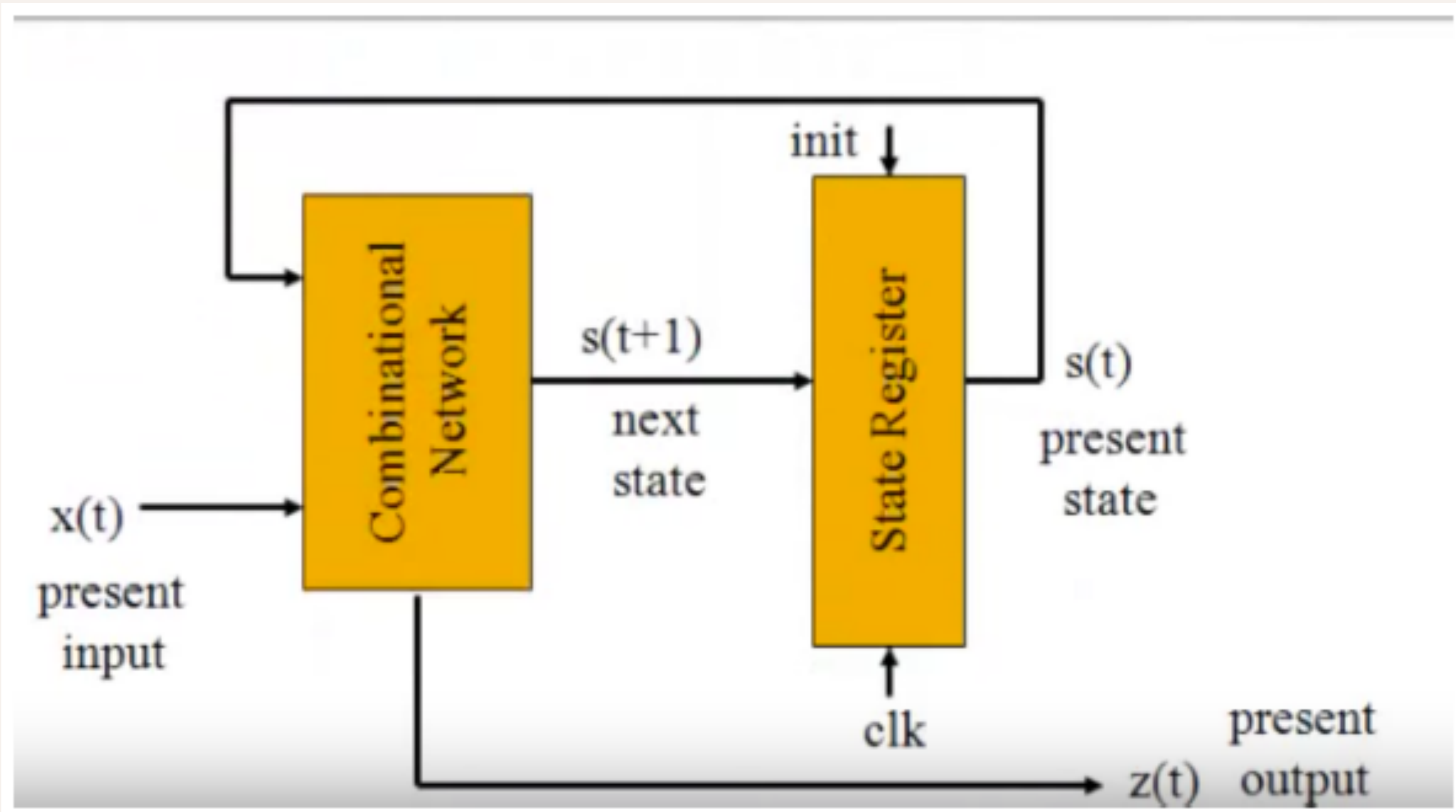


State Transition Table

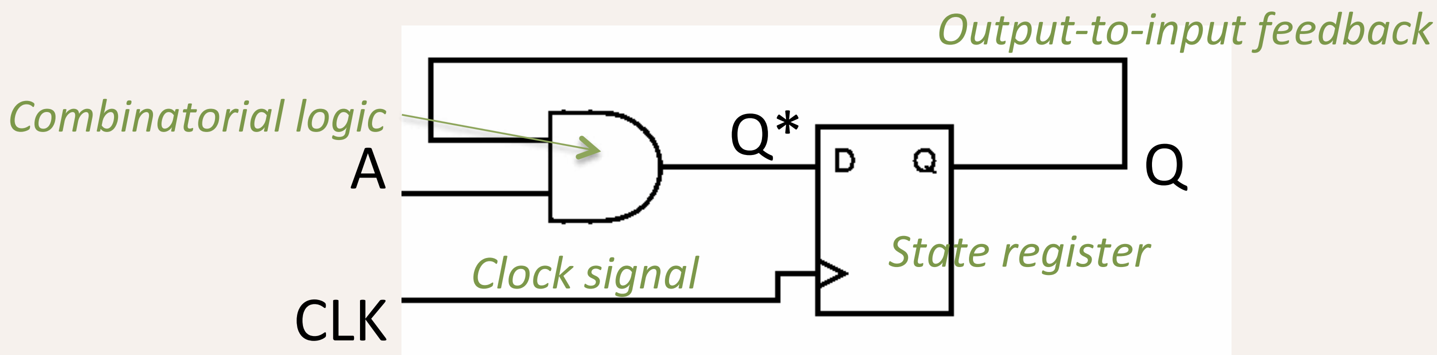
Current State	Input	Next State	Output
Locked	Coin	Unlocked	Unlocks the turnstile so that the customer can push through.
Locked	Push	Locked	Nothing – you're locked! ☺
Unlocked	Coin	Unlocked	Nothing – you just wasted a coin! ☺
Unlocked	Push	Locked	When the customer has pushed through, locks the turnstile.

Source: Wikipedia

General Form of FSMs



Example



$$Q^* = Q_O.A$$

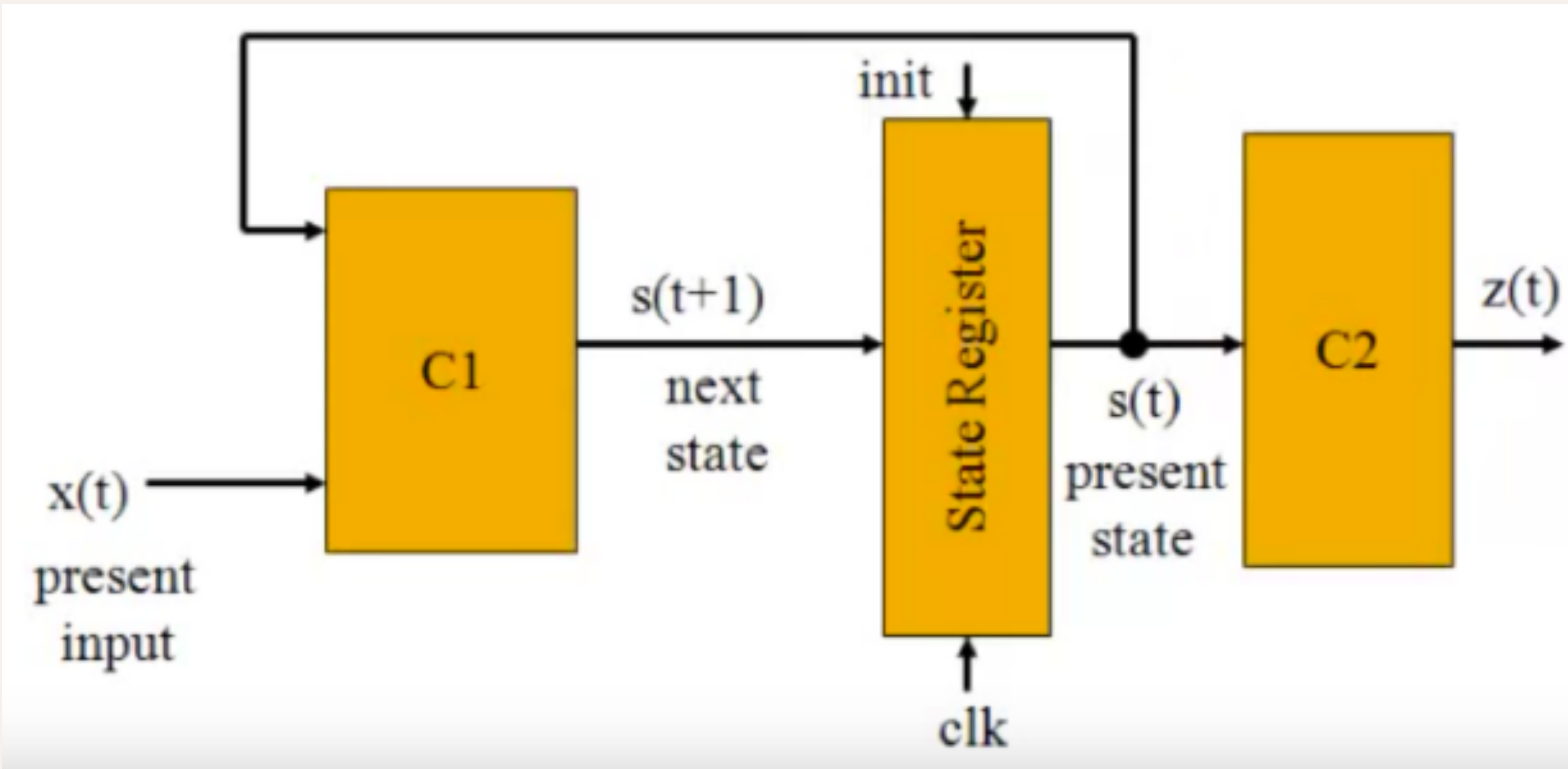
On the next rising edge of the clock, the output of the D-FF Q (Q^*) will become the previous value of Q (Q_O) **AND** the value of input A

FSM Types

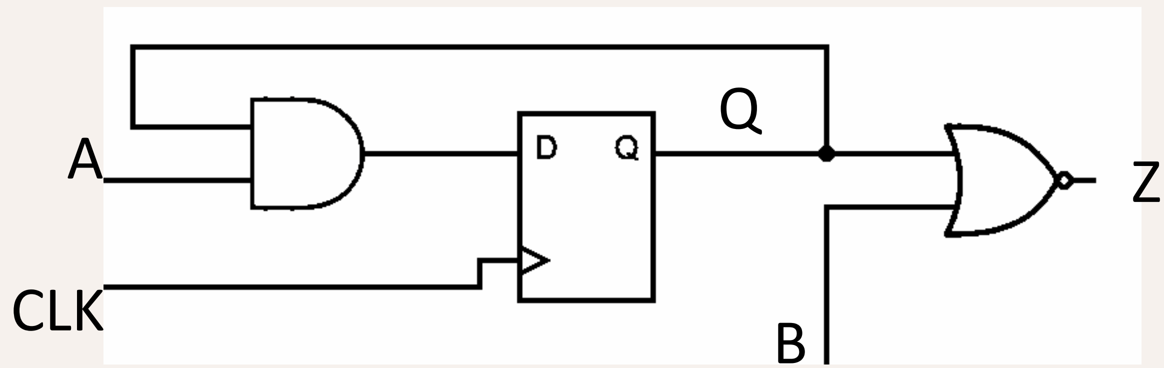
There are 2 types/models of FSMs:

- **Moore machine**
  – Output is function of present state only
- **Mealy machine**
  – Output is function of present state *and* present input

Moore Machine



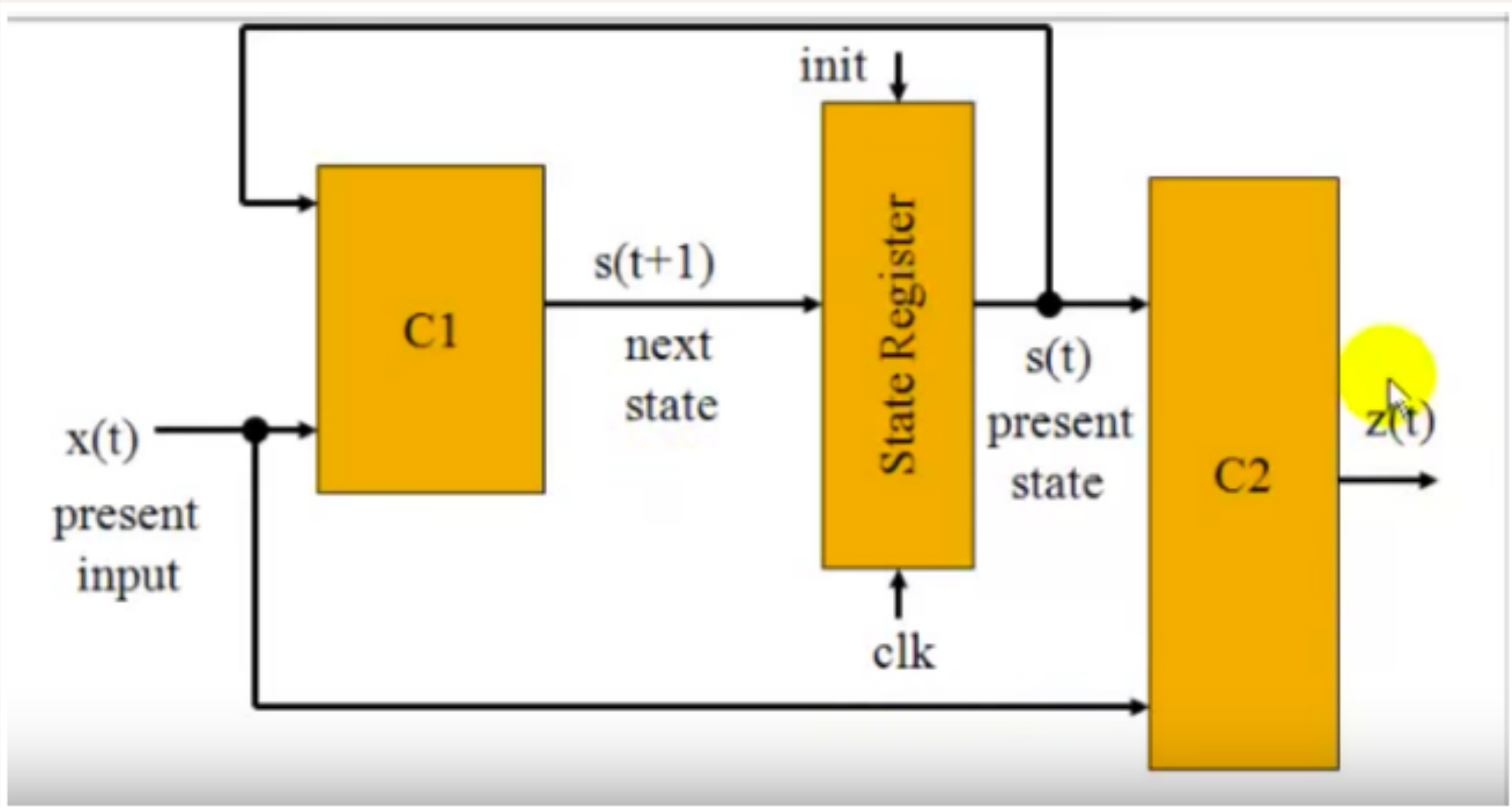
Example of a Moore Machine
(with 1 state)



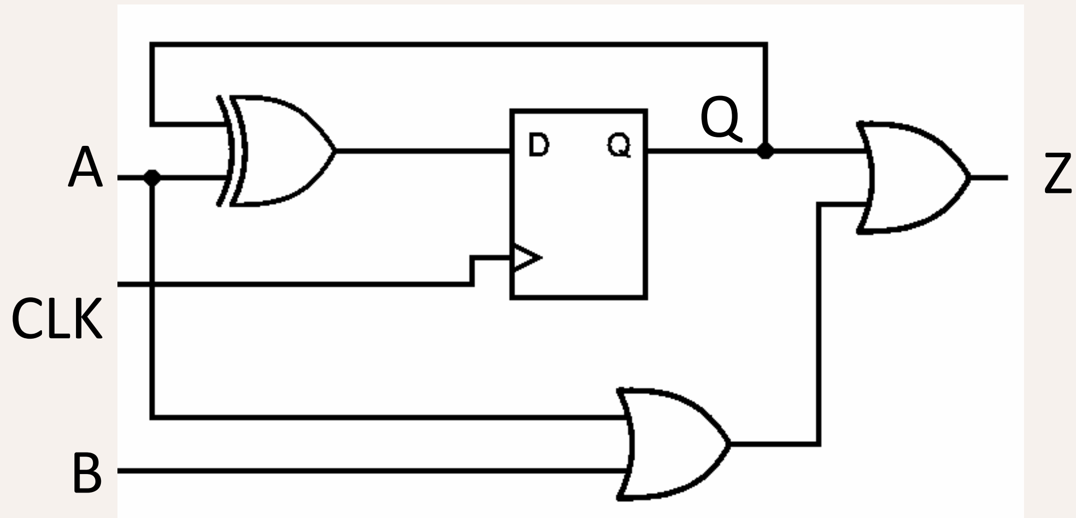
$$Z = \overline{(Q^* + B)} = \overline{(Q_O.A + B)}$$

On the next rising edge of the clock, the output of the entire circuit (Z) will become (the previous value of Q (Q_O) **AND** the value of input A) **NOR** B

Mealy Machine



Example of a Mealy Machine
(with 1 state)

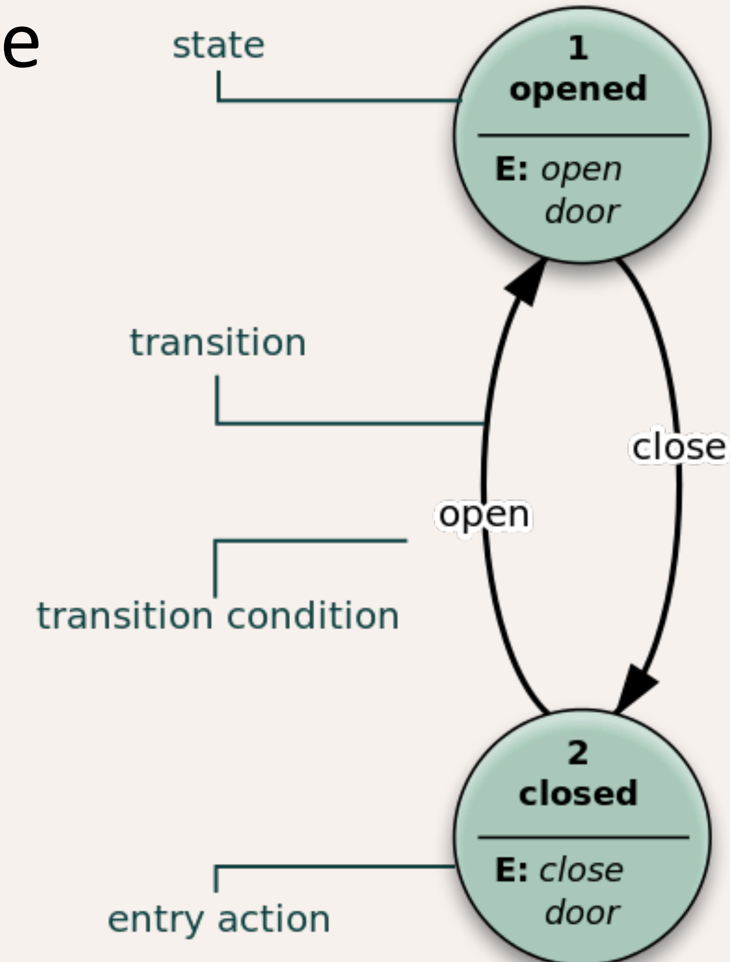


Z = (Q* + A + B) = (Q_O XOR A) + (A + B)

On the next rising edge of the clock, the output of the entire circuit (Z) will become …etc…

Diagraming State Machines

- A simple FSM example
- 2 states:
  - Door opened
  - Door closed
- This is called a **state diagram**



Example of a Moore Machine 1

WASHER_DRYER

- Let's "build" a sequential logic FSM that acts as a controller to a washer/dryer machine

SO:

- Before we begin, the machine is in an initial state that is waiting for you to insert a coin. We'll call that state the "Initial State" (inventive, no?)

- The machine will start a washer timer as soon as a coin is inserted. The timer is controlled by a signal (i.e. input var) called **TIMER_LT_30**, which is always initialized to be 1.

Example of a Moore Machine 1

WASHER_DRYER

- Upon inserting a coin in the machine, we will begin the ***wash cycle***. We'll call that state "Wash".
- This state will output a signal to fill the washer with water (**FILL_WATER**).
- As long as the timer is below 30 mins (**TIMER_LT_30 = 1**), the cycle continues.
- When the timer surpasses 30 mins (i.e. **TIMER_LT_30 = 0**), this state will end

Example of a Moore Machine 1

WASHER_DRYER

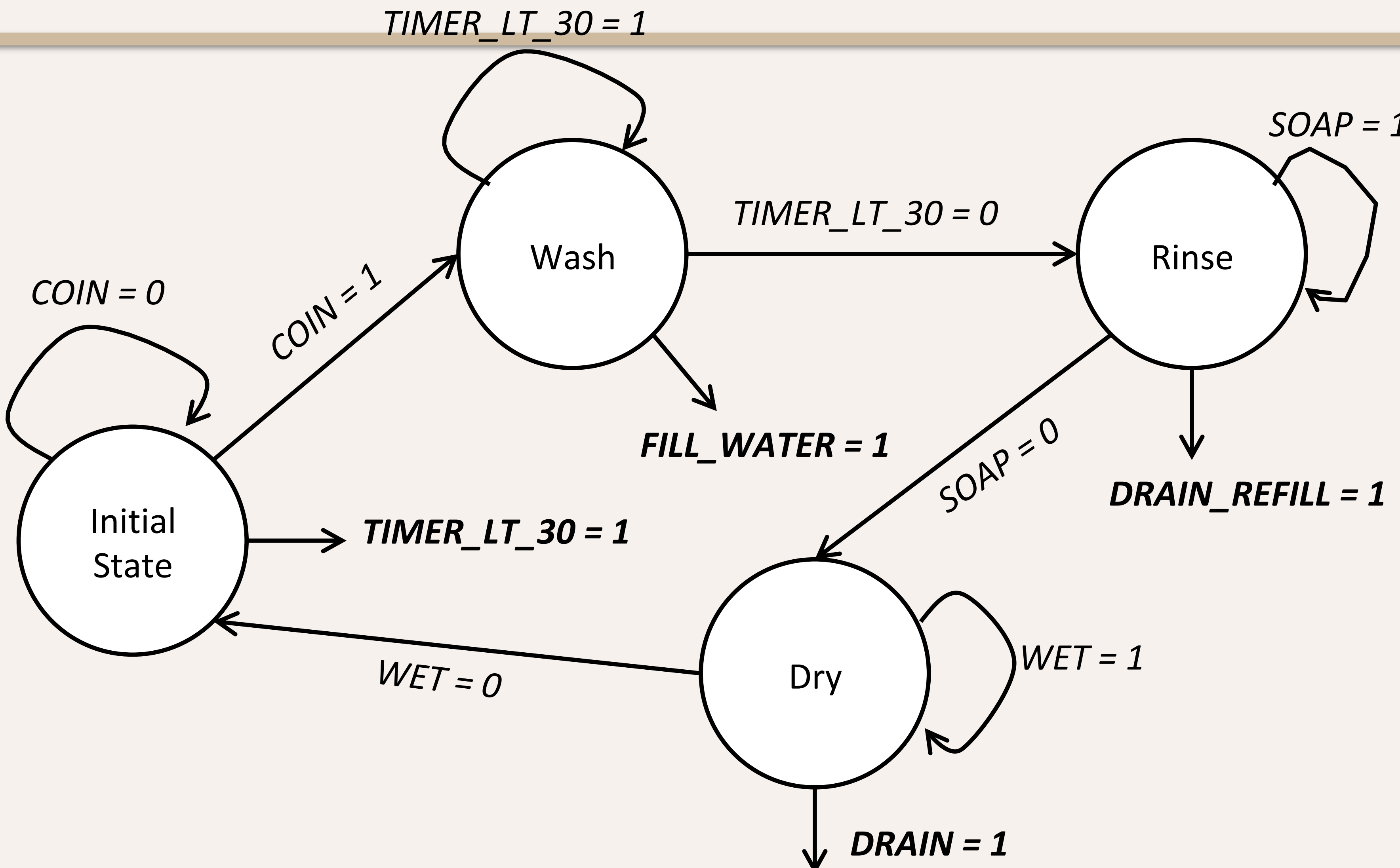
- When the timer hits 30 mins, we will begin the ***rinse cycle***. We'll call that "Rinse".
- This will output a signal to drain the water and refill with new water (**DRAIN_REFILL = 1**).
- As long as the soap sensor is on (**SOAP = 1**), the cycle continues.
- When the soap sensor turns off (i.e. **SOAP= 0**), this state will end

Example of a Moore Machine 1

WASHER_DRYER

- When the soap sensor goes off, we will begin the ***dry cycle***. We'll call that "Dry".
- This state will output a signal to drain the water and begin drying (**DRAIN = 1**).
- As long as the wet clothes sensor is on (**WET = 1**), the cycle continues.
- When the wet clothes sensor is off (**WET = 0**), we will stop!
- This means going back to the "**Initial State**"

State Diagram 1



Example of a Moore Machine 2

DETECT_1101

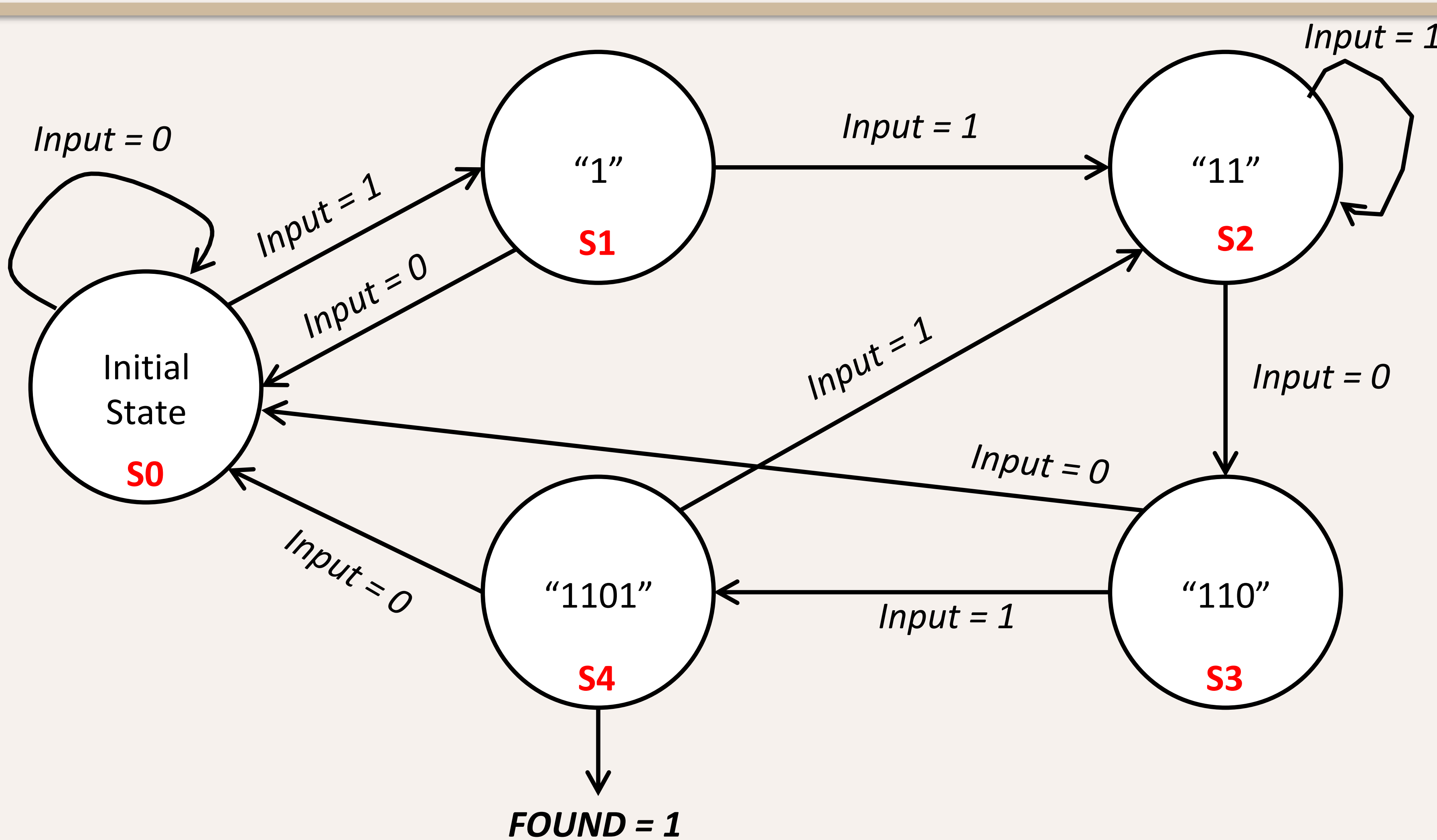
- Let's build a sequential logic FSM that always detects a specific serial sequence of bits: ***1101***

<u>*SO:*</u>

- We'll start at an "Initial" state (S0)
- We'll first look for a **1**. We'll call that "State 1" (S1)
  - Don't go to S1 if all we find is a **0**!
- We'll then keep looking for another **1**. We'll call that "State 11" (S2)
- Then… a **0**. We'll call that "State 110" (S3)
- Then another **1**.
  We'll call that "State 1101"(S4) – this will output a **FOUND** signal

- We will always be detecting "1101" (it doesn't end)

State Diagram 2



Going from State Diagram to Circuit

- There's more than 1 way to do this, but the most popular is the "**One Hot Method**"

- Give each state it's own D-FF output
  - # of FFs needed = # of states

- Inputs to the FFs are combinatorial logic that can simplified into a "sum-of-products" type of Boolean expression

- Current CAD software can do this automatically
- Implementation is usually done on a simulator (software), or prototype hardware Integrated Circuit (FPGA)

Encoding our States

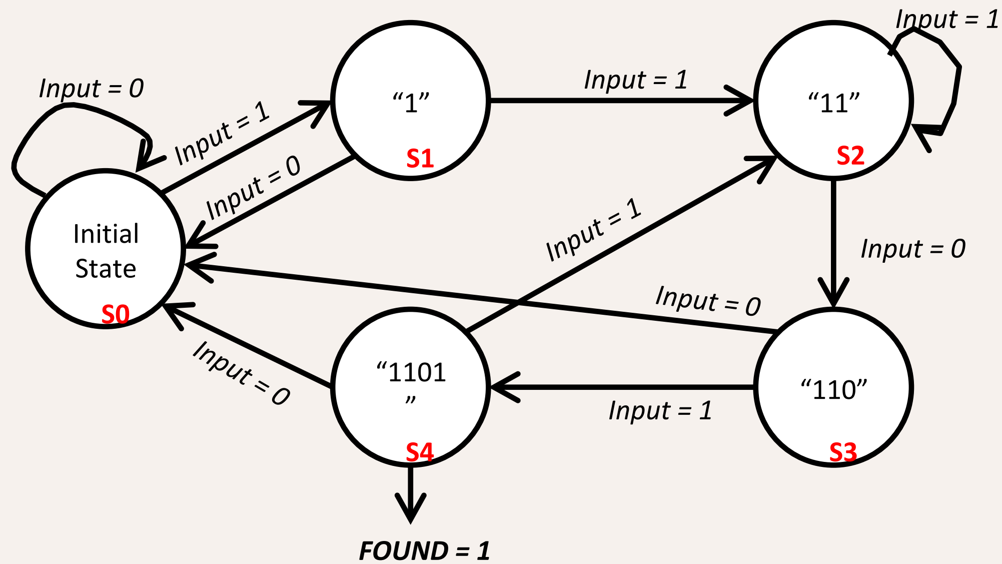
Per the last example: We had 5 separate states:

NAME		Binary Code	"One Hot" Code	OUTPUT
• Initial State	S0	000	00001	
• "1"	S1	001	00010	
• "11"	S2	010	00100	
• "110"	S3	011	01000	
• "1101"	S4	100	10000	FOUND

- Advantage of this "One Hot" approach?
  - When we implement the machine with circuits, we can use a D-FF for every state (so, in this example, we'd use 5 of them)

Using the “One Hot” Code to Determine the Circuit Design

- Every state has 1 D-FF
- We can see that (follow the arrows!!):



S1* = $S0.I$

S2* = $S1.I + S2.I + S4.I$

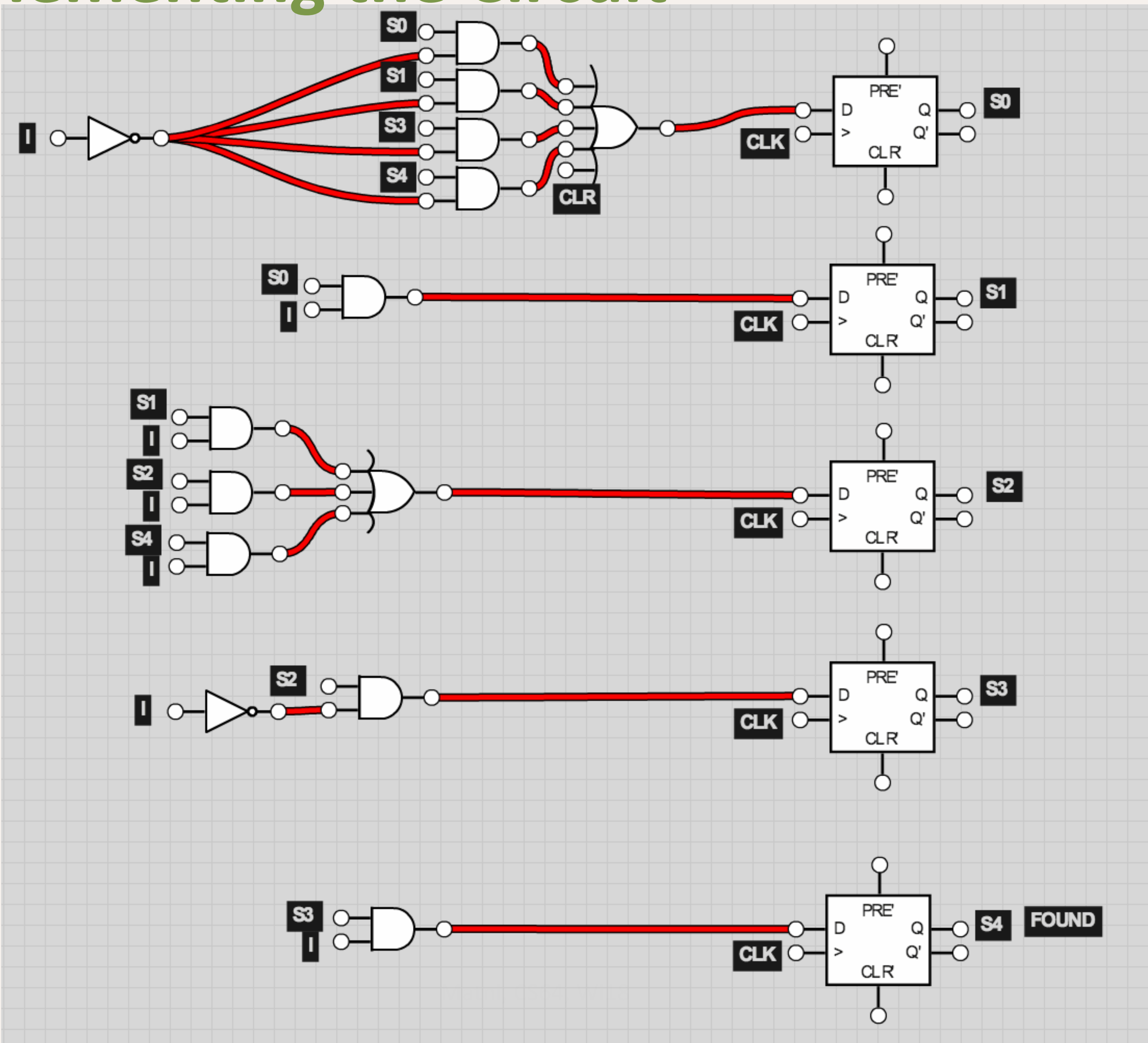
S3* = $S2.\overline{I}$

S4* = $S3.I$

S0* = $S0.I + S1.\overline{I} + S3.\overline{I} + S4.\overline{I}$

also, when S4 happens, FOUND = 1, i.e. **FOUND = S4**

Implementing the Circuit



Your To Dos

- Lab #8 is due end of day Friday

</LECTURE>