

Lecture 14:

Final Lecture

Announcements

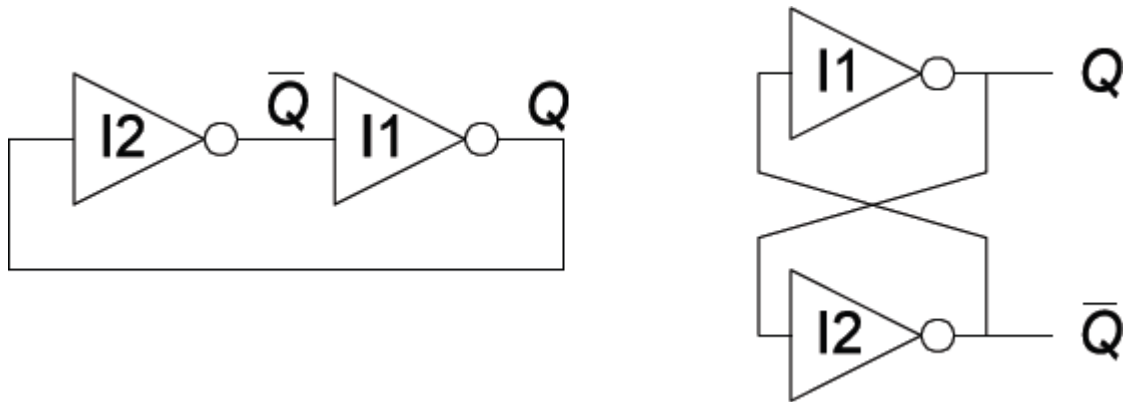
- Project 3 due on tomorrow
 - Due August 11th 11:55pm
 - Due day before Final, so plan accordingly
 - No Extension
 - Max point: 120, not 130
- Final is on Wednesday
 - 3-hour timed exam
 - Open for 24-hours
 - Few Sample Questions just released
- Remember to take the instructional survey
 - Currently ~70% of the class has taken it.
 - If at least 95% of the class takes it, one point will be added after final grade calculations
 - ~22 more students need to take it.
- Lecture today:
 - Finishing up FSM and Sequential Circuits
- Rest of Lecture/Recitation
 - Questions on any of the material from the whole course, questions on PA3

How are Sequential Circuits different from Combinational Circuits?

- How to make a circuit out of gates that is not combinational?
 - Feed-back: create loops in the diagram
 - Outputs of sequential logic depend on both current and prior values – **it has memory**
- Definitions:
 - **State**: all the information about a circuit to explain its future behavior
 - **Latches and flip-flops**: state elements that store one bit of state
 - **Synchronous sequential elements**: combinational logic followed by a bank of flip-flops

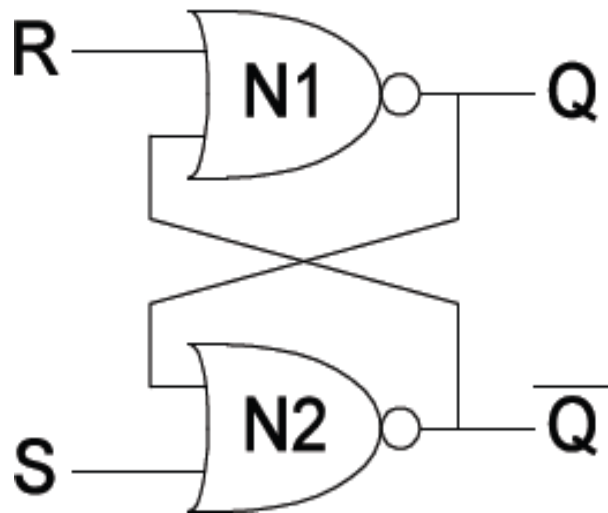
Bistable Circuits (Static latch)

- Fundamental building blocks of other elements
 - it can remember the state of the circuit indefinitely
- No inputs
- Two outputs (Q and Q')



Set/Reset Latch

A latch with NOR gates



- $S = 1, R = 0$

- $S = 0, R = 1$

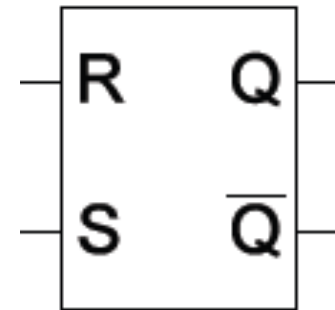
- $S = 0, R = 0$

- $S = 1, R = 1$

S/R Latch Symbol

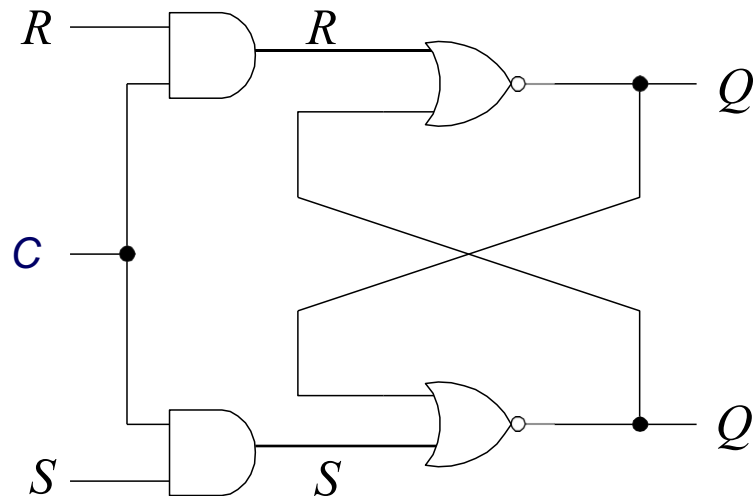
- Set operation
 - makes output 1 ($S = 1, R = 0, Q = 1$)
- Reset operation
 - makes output 0 ($S = 0, R = 1, Q = 0$)
- What about invalid state? ($S = 1, R = 1$)
makes $Q = \bar{Q} = 0$

SR Latch
Symbol



S/R Latch with Enable

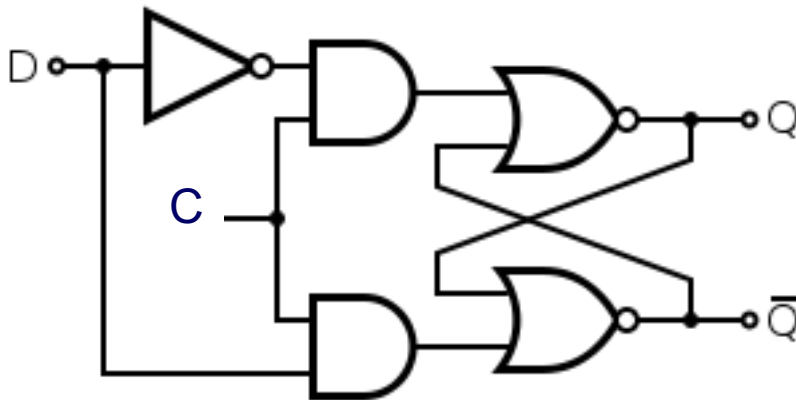
- The SR latch is sensitive to its inputs all the time. It is sometimes useful to be able to disable the inputs.
- The *SR latch with enable* accomplishes this by adding an enable input, C , to the original implementation of the latch that allows the latch to be enabled or disabled.



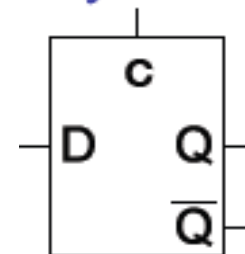
- The SR latch with enable is still sensitive to $S = R = 1$

D Latch

- Prevents $S = R = 1$ by adding an inverter
- The D latch has two inputs (C and D)
 - C (control input): controls when the output changes
 - D (data input): controls what the output changes to
- When $C = 1$, D passes through to Q (transparent latch)
- When $C = 0$, Q holds previous value (opaque latch)



D Latch
Symbol

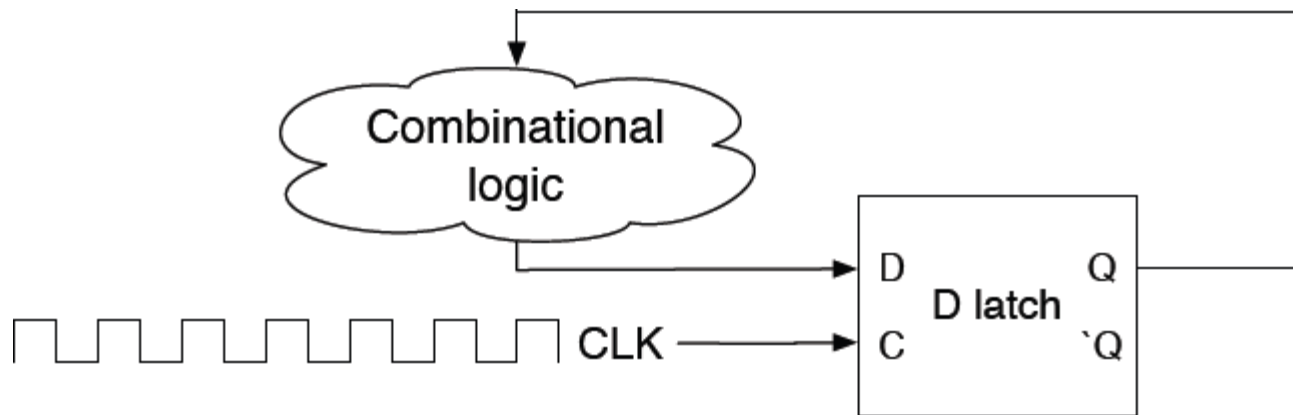


How to Coordinate Multiple Components of a Circuit?

- To do computations, we need more than just combinational circuits We need to use past circuit state to influence future output (latches)
- But how do we coordinate computations and the changing of state values across lots of different parts of a circuit?
 - How to synchronize latches to change values at the same time?
- We use CLOCKING (eg. 1 GHz clock on Intel processors)
- **On each clock pulse, combinational computations are performed, and results stored in latches**
- How to introduce clocks into latches?

Flip-flops: Latches on a Clock

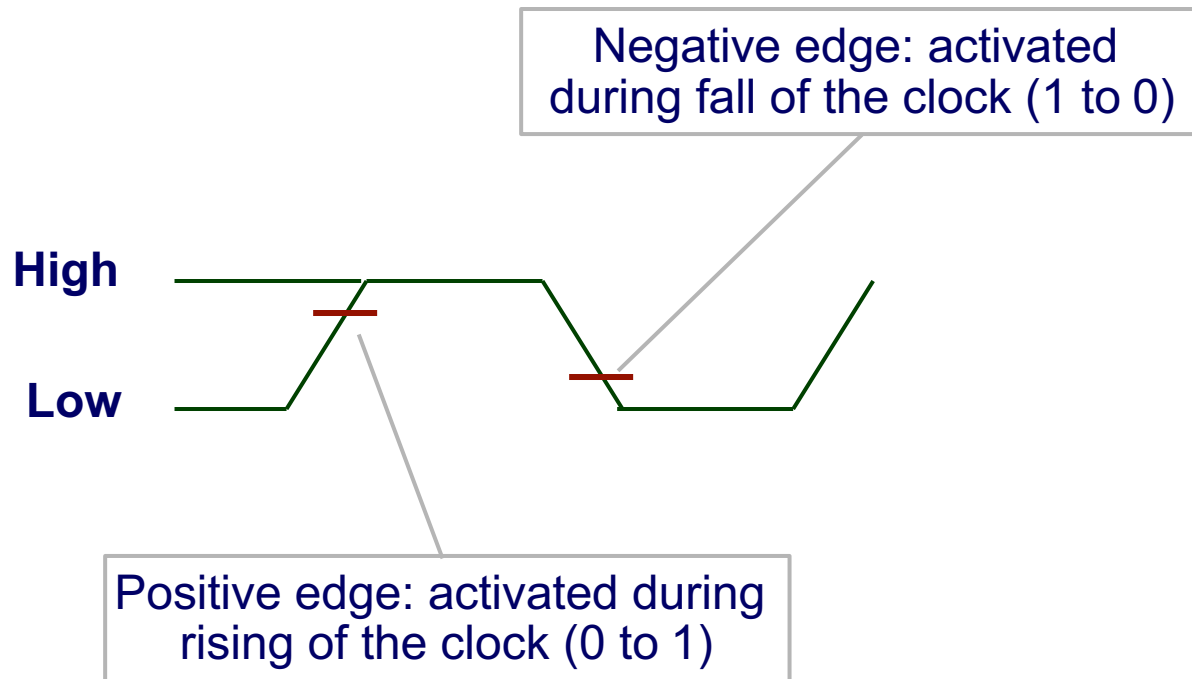
- A straightforward latch is not safely synchronous (or predictably synchronous)
 - Flip-flops can have its state changed only at a single, known instant of time.



- Flip-flops designed so that outputs will NOT change within a single clock pulse

Flip-Flop

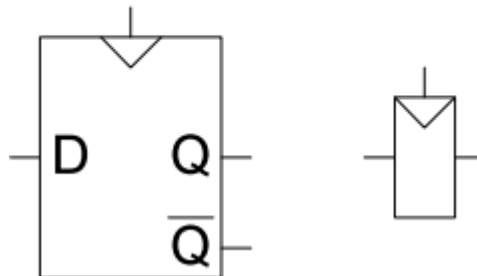
- Sensitive to the edge (transition) of the clock
 - rising or falling of the clock



Positive Edge-Triggered D Flip-Flop

- Two inputs:
 - Clk, D
- Function
 - The flip-flop samples D on rising edge of the Clk
 - When Clk goes from 0 to 1, D passes through Q
 - Otherwise, Q holds its value
 - Q only changes on rising edge of the Clk
- A flip-flop is called “edge-triggered” because it is activated only on the clock edge

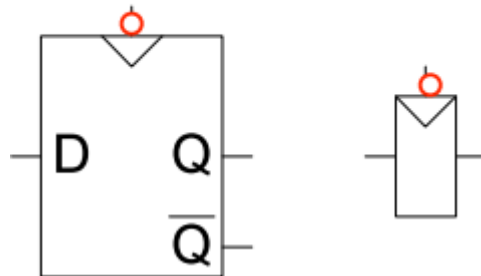
D Flip-Flop
Symbols



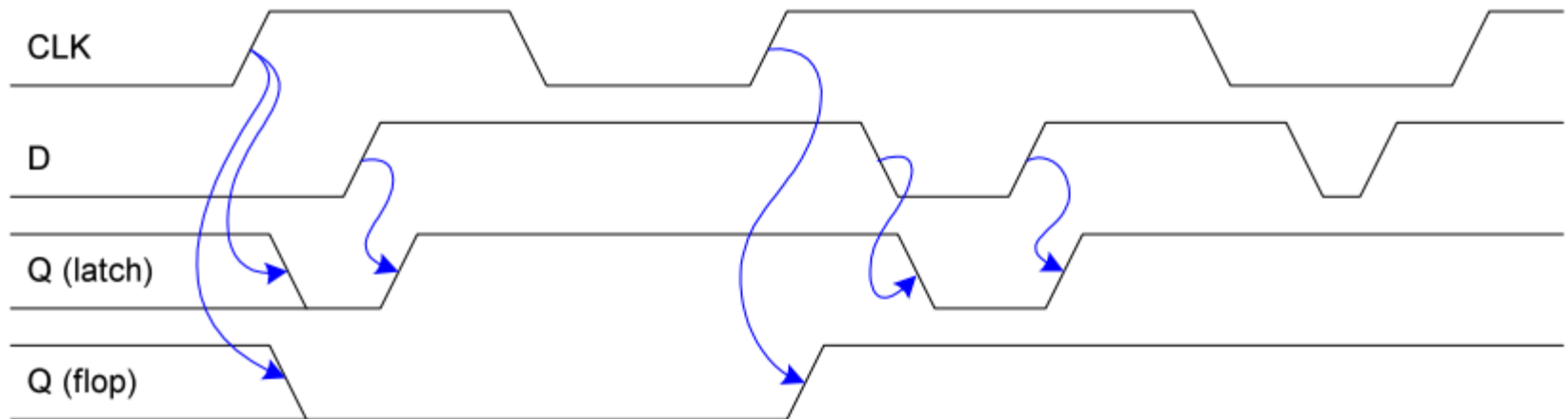
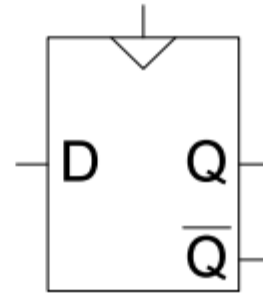
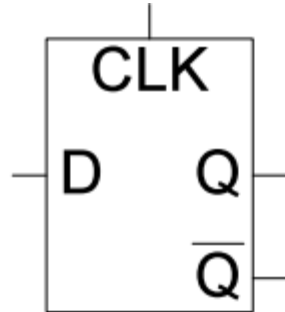
Negative Edge-Triggered D Flip-Flop

- Two inputs:
 - Clk, D
- Function
 - The flip-flop samples D on the falling edge of Clk
 - When Clk falls from 1 to 0, D passes through to Q
 - Otherwise, Q holds its previous values
 - Q changes only on the falling edge of Clk
- A flip-flop is called an edge-triggered device because it is activated on a clock cycle

D Flip-Flop
Symbols

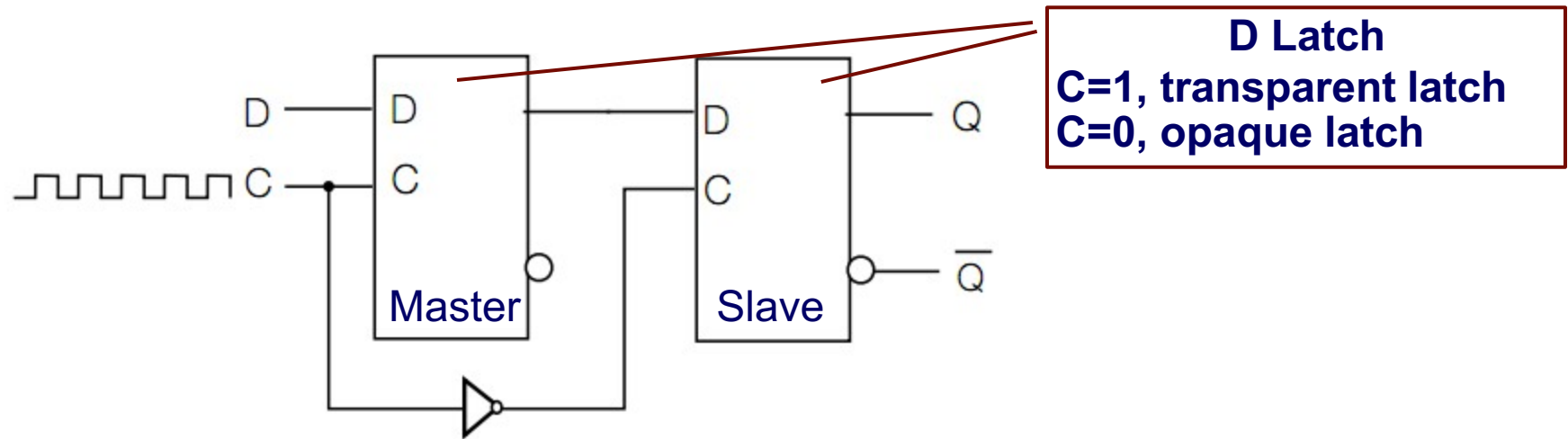


Flip-Flop versus Latch



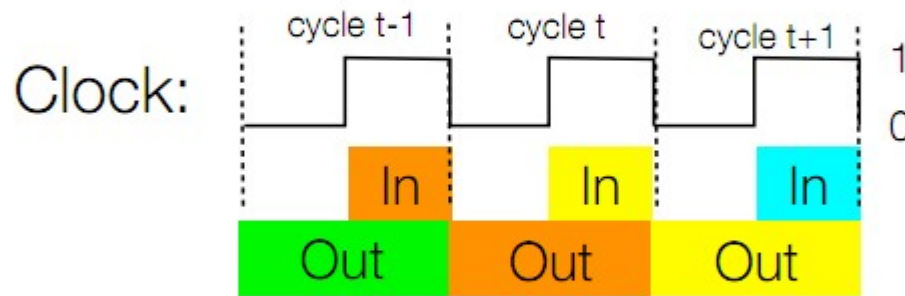
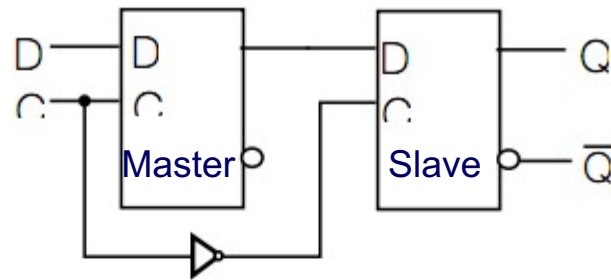
Latch outputs change at any time, flip-flops only during clock transitions

Master Slave D Flip-Flop



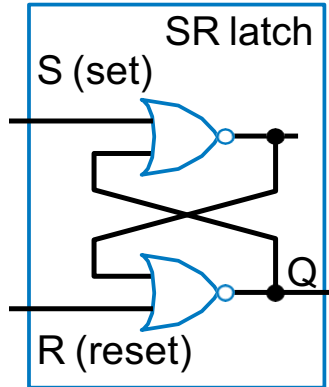
- C (Control) is fed a clock pulse (alternates between 0 and 1 with fixed period)
 - C=1: Master latch “on”, Slave latch “off”
 - New D input read into master
 - Previous Q values still emitted (not affected by new D inputs)
 - C=0: Master latch “off”, Slave latch “on”
 - Changing D inputs has no effect on Master (or Slave) latch
 - D inputs from last time C=1 stored safely in Master and transferred into Slave and reflected on output Q

Master Slave D Flip-Flop Activation Time



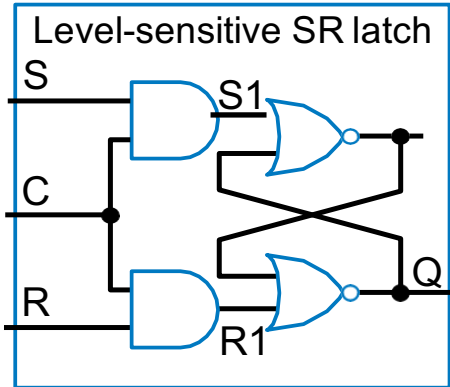
- $Q(t)$: value output by Flip-Flop during the t^{th} clock cycle (clock =0, then 1 during a full cycle)
- Depends on input during end of $t-1^{\text{st}}$ cycle

Bit Storage Summary



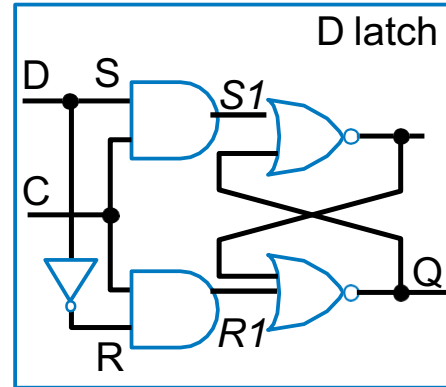
Feature: $S=1$ sets Q to 1, $R=1$ resets Q to 0.

Problem: $SR=11$ yields undefined Q , other glitches may set/reset inadvertently.



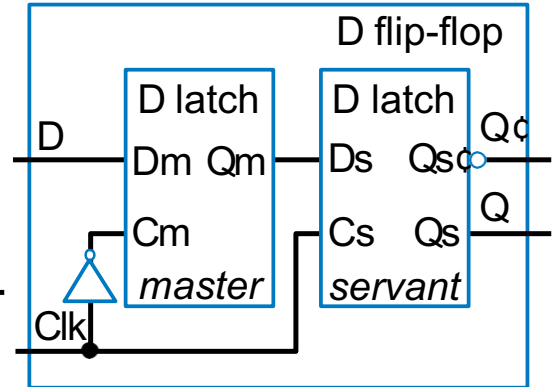
Feature: S and R only have effect when $C=1$. An external circuit can prevent $SR=11$ when $C=1$.

Problem: avoiding $SR=11$ can be a burden.



Feature: SR can't be 11.

Problem: $C=1$ for too long will propagate new values through too many latches; for too short may not result in the bit being stored.

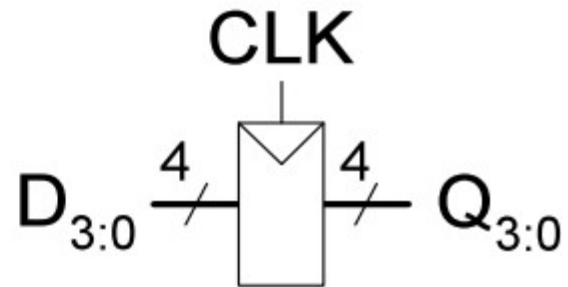
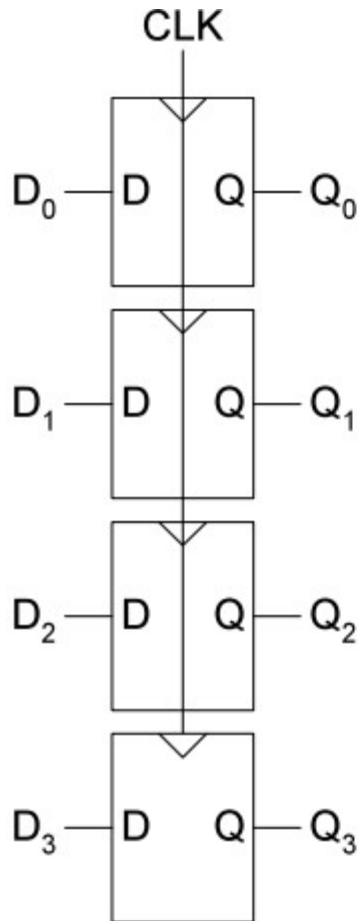


Feature: Only loads D value present at rising clock edge, so values can't propagate to other flip-flops during same clock cycle.

Tradeoff: uses more gates internally, and requires more external gates than SR — but transistors today are more plentiful and cheaper.

We considered increasingly better bit storage until we arrived at the robust D flip-flop bit storage

Registers



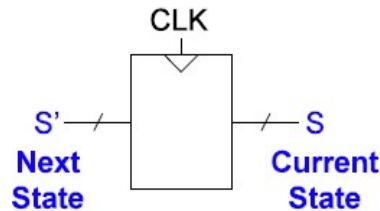
Synchronous Sequential Logic Design

- Structures that can process and store information
 - Registers contain the state of the system
 - Combinational circuits process information
- State changes at the clock edge, so the system is synchronized to the clock
- Rules of synchronous sequential circuit composition:
 - Every circuit is either a register or a combinational circuit
 - At least one circuit element is a register
 - All registers receive the same clock signal
 - Every cyclic path contains one register
- Two common synchronous sequential circuits
 - Finite state machines (FSMs)
 - Pipelines

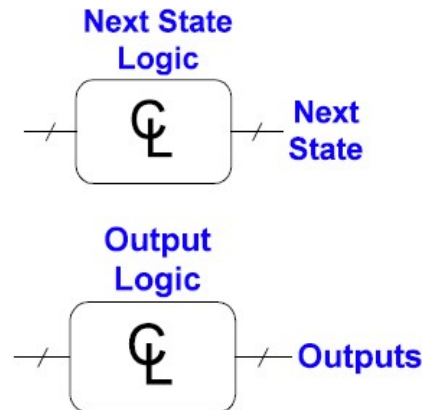
Finite State Machines

- A mathematical model of computation used to design:
 - Computer programs
 - Sequential logic circuits
- FSM = State register + combinational logic

Stores the next state and loads the next state at clock edge

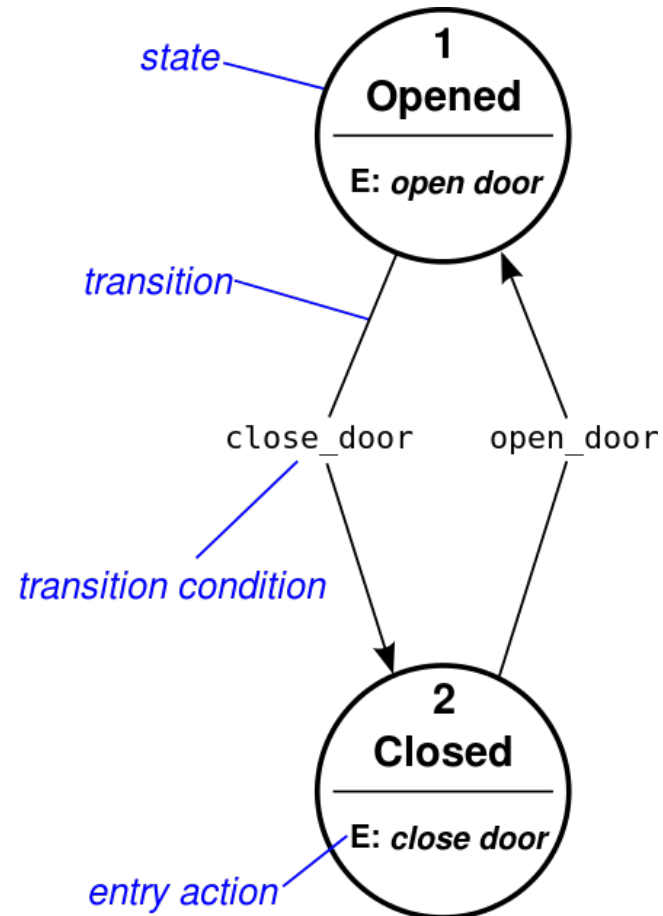


Computes the next state and computes the outputs



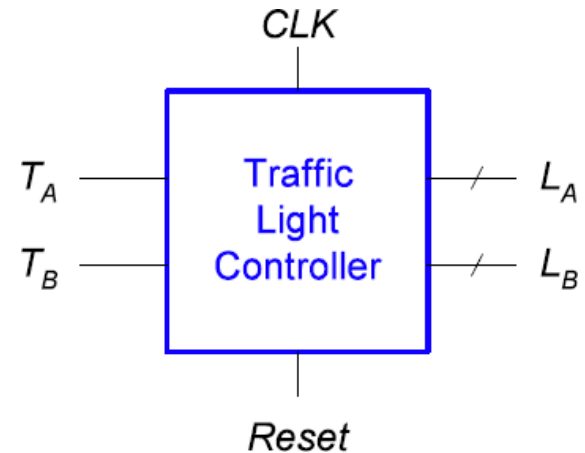
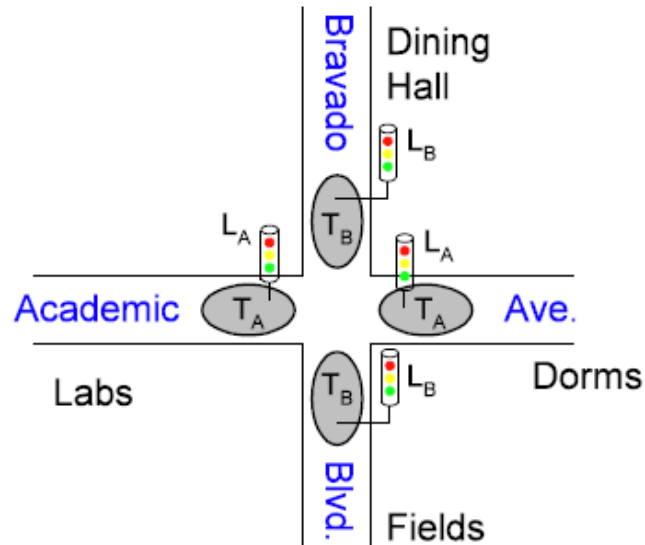
Finite State Machines

- Can be represented using a state diagram
 - Finite number of states
 - Transitions



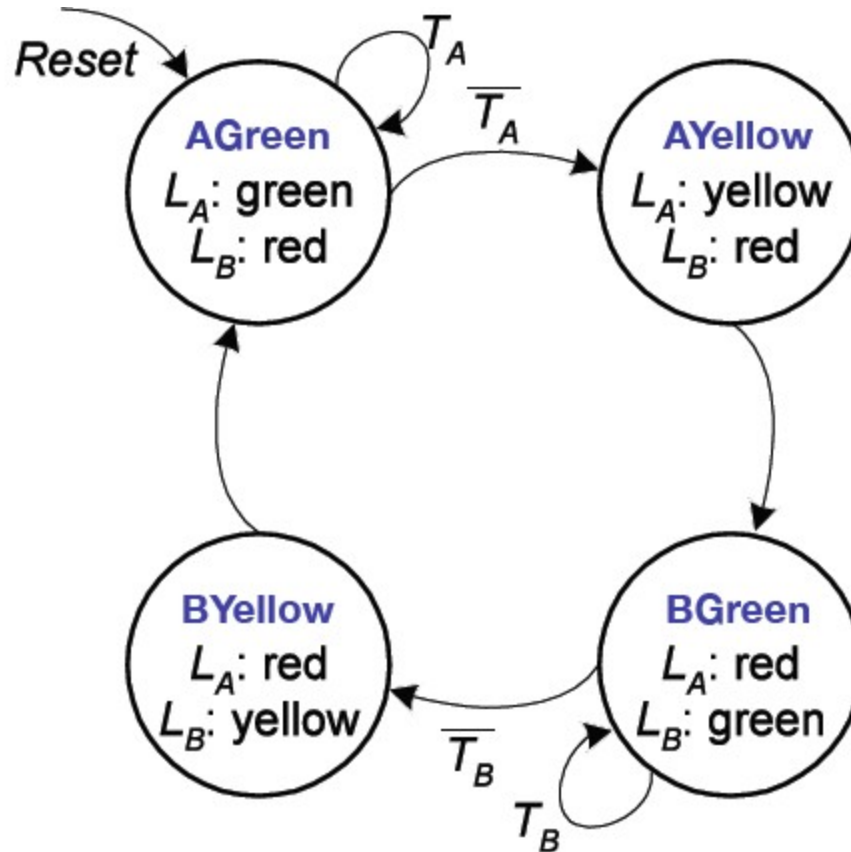
FSM: Traffic Light Controller Example

- Traffic sensors: T_A , T_B (TRUE when there is traffic)
- Lights: L_A , L_B



FSM State Transition Diagram

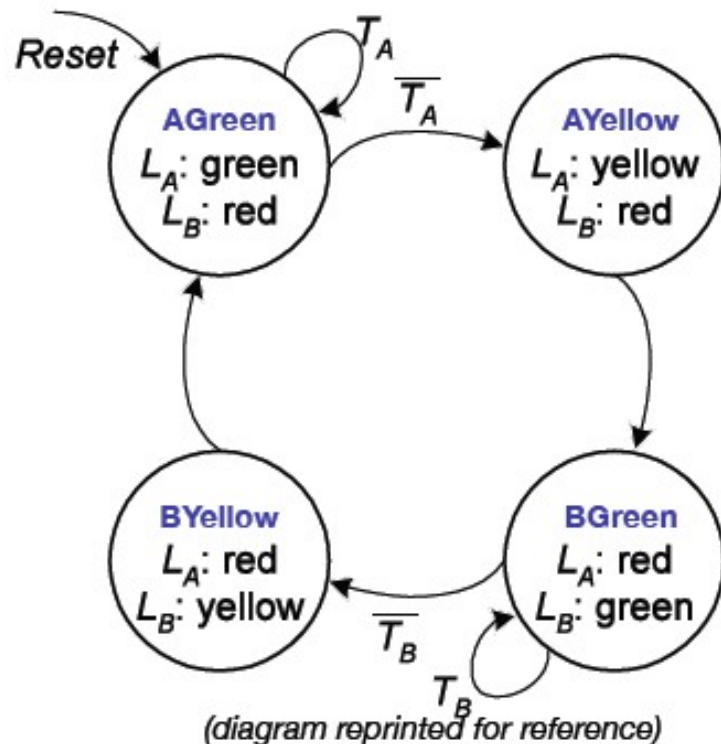
- States: Circles
- Transitions: Arcs



FSM State Transition Table

- State transitions from diagram can be rewritten in a state transition table

(S = current state, S' = next state)



Current State	Inputs		Next State
S	TA	TB	S'
AGreen	0	X	AYellow
AGreen	1	X	AGreen
AYellow	X	X	BGreen
BGreen	X	0	BYellow
BGreen	X	1	BGreen
BYellow	X	X	AGreen

Encoded State Transition Table

- After selecting a state encoding, the symbolic states in the transition table can be realized with current state/next state bits

State	Encoding	
	S1	S0
AGreen	0	0
AYellow	0	1
BGreen	1	0
BYellow	1	1

Current State	Encoded Current State		Inputs		Next State	Encoded Next State	
S	S1	S0	TA	TB	S'	S1'	S0'
AGreen	0	0	0	X	AYellow	0	1
AGreen	0	0	1	X	AGreen	0	0
AYellow	0	1	X	X	BGreen	1	0
BGreen	1	0	X	0	BYellow	1	1
BGreen	1	0	X	1	BGreen	1	0
BYellow	1	1	X	X	AGreen	0	0

Computing Next State Logic

Current State	Encoded Current State		Inputs		Next State	Encoded Next State	
S	S1	S0	TA	TB	S'	S1'	S0'
AGreen	0	0	0	X	AYellow	0	1
AGreen	0	0	1	X	AGreen	0	0
AYellow	0	1	X	X	BGreen	1	0
BGreen	1	0	X	0	BYellow	1	1
BGreen	1	0	X	1	BGreen	1	0
BYellow	1	1	X	X	AGreen	0	0

- From K-maps, figure out expressions for the next state:
 - $S1(t+1) = S1(t) \text{ xor } S0(t)$
 - $S0(t+1) = \overline{S1}(t) \overline{S0}(t) \overline{TA} + S1(t) \overline{S0}(t) \overline{TB}$
- Another way of writing the same thing (just a change of notation):
 - $S1' = S1 \text{ XOR } S0$
 - $S0' = S1 \wedge S0 \wedge TA \wedge + S1 S0 \wedge TB \wedge$

FSM Output Table

- FSM output logic is computed in similar manner as next state logic
- In this system, output is a function of current state (Moore machine)
- Alternative – Mealy machine (output function of both current state and inputs, though we won't cover this in class)

output encoding

Output	Encoding	
Green	0	0
Yellow	0	1
Red	1	0

output truth table

State	State		LA		LB	
	S1	S0	LA1	LA0	LB1	LB0
AGreen	0	0	0	0	1	0
AYellow	0	1	0	1	1	0
BGreen	1	0	1	0	0	0
BYellow	1	1	1	0	0	1

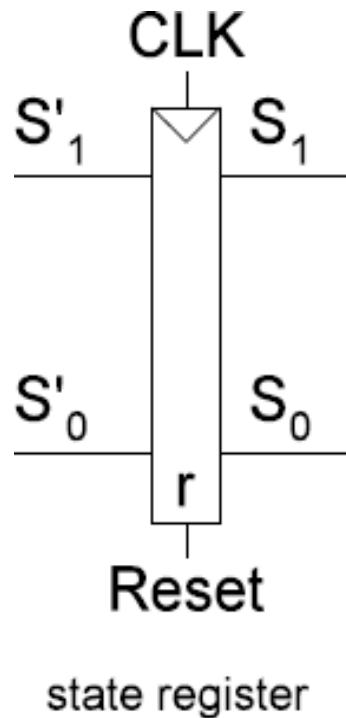
- Compute output bits as function of state bits

$$LA1 = S1; LA0 = S1'S0$$

$$LB1 = 'S1; LB0 = S1S0$$

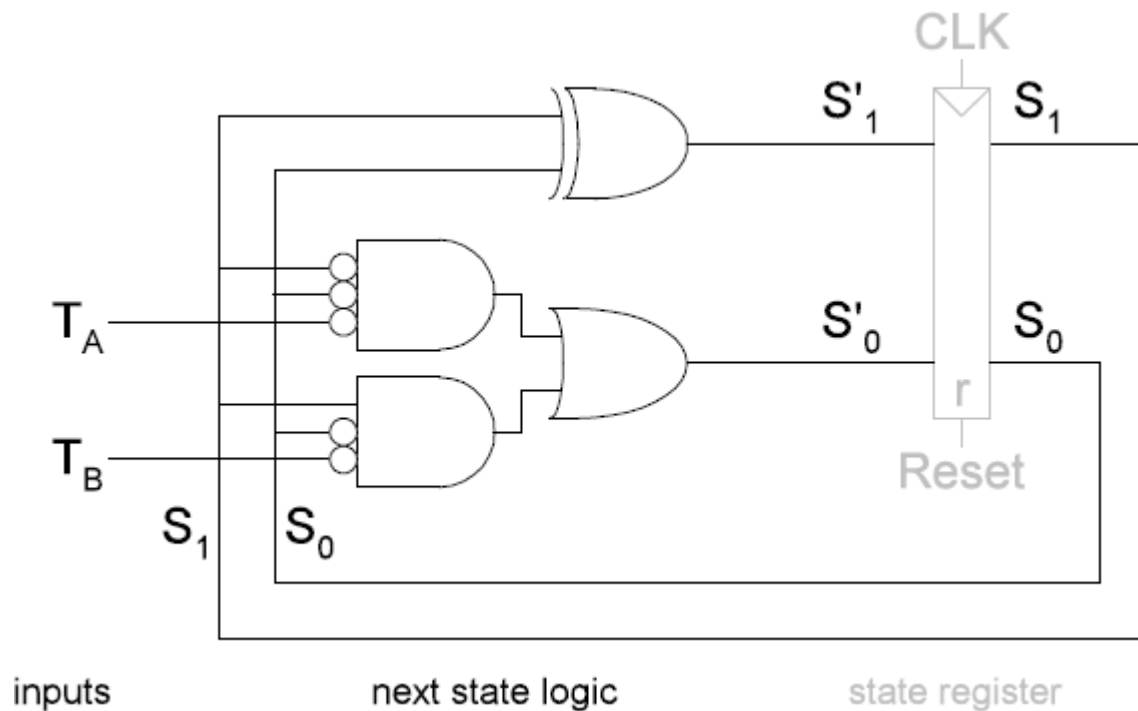
red light

State Register: Assume D Flip-Flop



FSM: Figure out Next State Logic

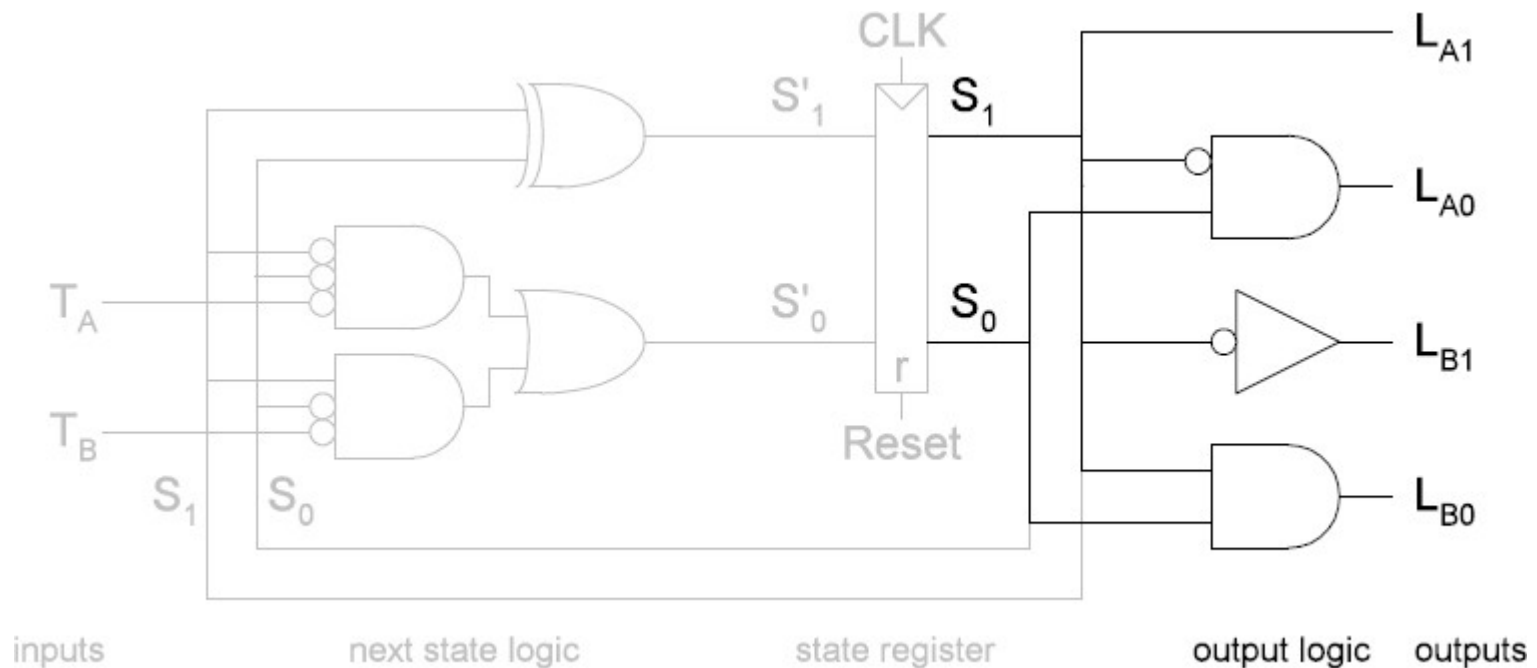
- $S_1' = S_1 \text{ XOR } S_0$
- $S_0' = S_1 \wedge S_0 \wedge T_A \vee S_1 S_0 \wedge T_B$



FSM: Figure out Output Logic

$$LA1 = S1; LA0 = S1'S0$$

$$LB1 = \neg S1; LB0 = S1S0$$



FSM: divisible by 3

- Construct a "divisible by 3" FSM that accepts a binary number entered 1 bit at a time, MSB (most significant bit) first, and indicate with a light if the number entered so far is divisible by 3.

1) $N \bmod 3$

$$0 \rightarrow N = 3p + 0$$

$$1 \rightarrow N = 3p + 1$$

$$2 \rightarrow N = 3p + 2$$

2) If the number so far is N , after a digit b is entered, the new number is $N' = 2N + b$

$N \bmod 3$

$$0 \rightarrow N = 3p + 0, \text{ after digit } b \text{ is entered, } N' = 6p + b.$$

$$N' = b \bmod 3$$

$$1 \rightarrow N = 3p + 1, \text{ after digit } b \text{ is entered, } N' = 6p + 2 + b.$$

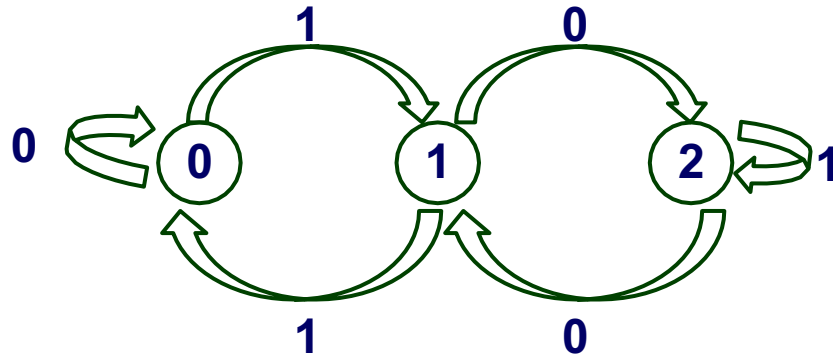
$$N' = b + 2 \bmod 3$$

$$2 \rightarrow N = 3p + 2, \text{ after digit } b \text{ is entered, } N' = 6p + 4 + b.$$

$$N' = b + 1 \bmod 3$$

FSM: divisible by 3

1. State Transition Diagram



2. State Transition Table

S1	S0	b	S1'	S0'	light
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	0	0	1
1	0	0	0	1	0
1	0	1	1	0	0

2. State Encoding

State	S1	S0
0	0	0
1	0	1
2	1	0

3. Next State Logic

$$S1' = \overline{S1} \overline{S0} \overline{b} + S1 \overline{S0} b$$

$$S0' = \overline{S1} \overline{S0} b + S1 \overline{S0} \overline{b}$$

4. Output Logic

$$\text{light} = \overline{S1} \overline{S0}$$