

CMSC 313 Lecture 22

- J-K flip-flops
- Edge-triggered flip-flops
- Finite state machines
- DigSim Assignment 2

Recap: Latches vs Flip-Flops

• Latch

- ◇ Output changes right after the input changes
- ◇ No reference to clocking event
- ◇ M&H's "SR flip-flop" is often called an SR latch in other texts.

• Level-Sensitive Latch

- ◇ A latch that operates only when the clock is high or only when low
- ◇ M&H's "clocked SR flip-flop" is a level sensitive latch

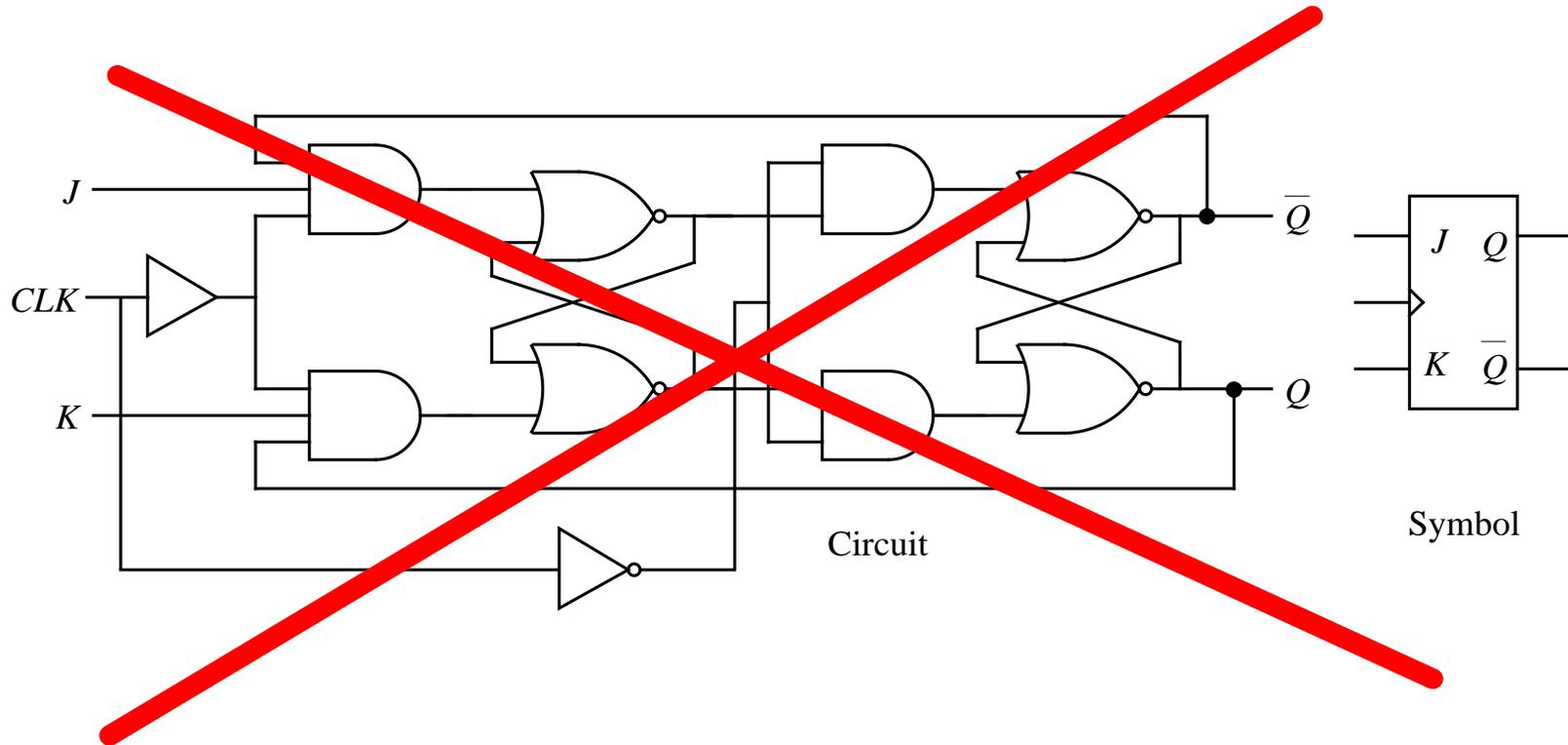
• Flip-Flop

- ◇ Reserved for circuits that record the input only during clocking events
- ◇ The output of the flip-flop does not change during this clocking event
- ◇ M&H's "master-slave flip-flop" fits this definition

J-K Flip-Flops

- Allows both "set" and "reset" to be 1
- When both J and K are 1, the output toggles
- If the clock is high, "endless toggle" occurs
- Master-slave J-K flip-flops solve the endless toggle problem, but has the ones-catching problem
- Use edge-triggered flip-flops to eliminate the ones-catching problem.

Master-Slave J-K Flip-Flop

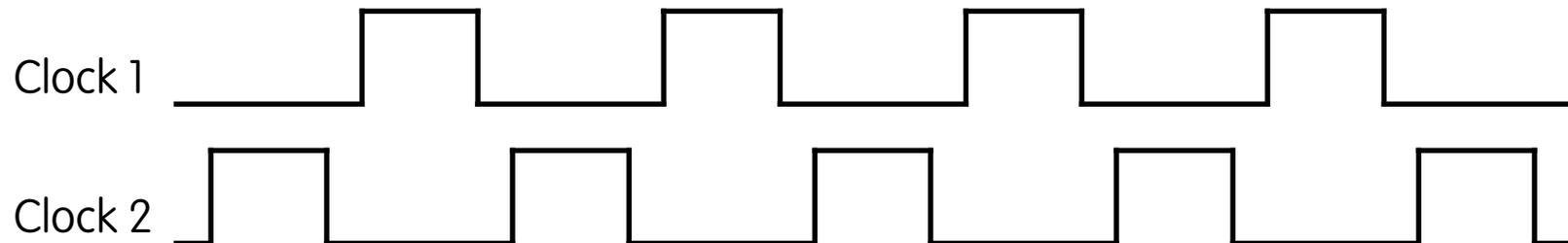


Master-Slave Flip-Flops Revisited

- **Master-slave J-K flip-flops**

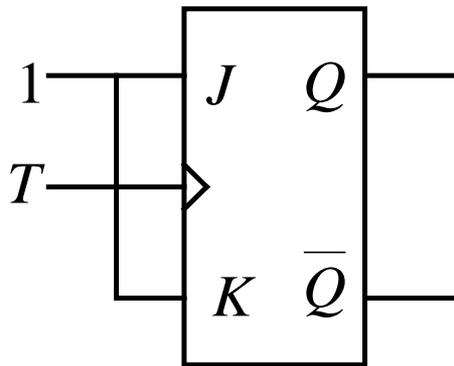
- ◇ Should get rid of the endless toggle condition when $J=1$ & $K=1$
- ◇ Clocking Problem 1: if master turns on before slave turns off, slave latches the input as soon as the clock goes high.
- ◇ Clocking Problem 2: if slave turns on before master turns off, feedback of Q and $\neg Q$ propagates through the master.

- **Two-Phase Clocks: always turn off first!**

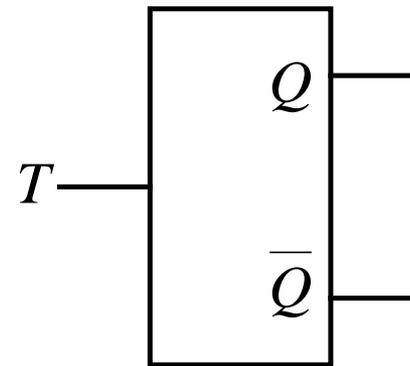


Clocked T Flip-Flop

- The presence of a constant 1 at J and K means that the flip-flop will change its state from 0 to 1 or 1 to 0 each time it is clocked by the T (Toggle) input.



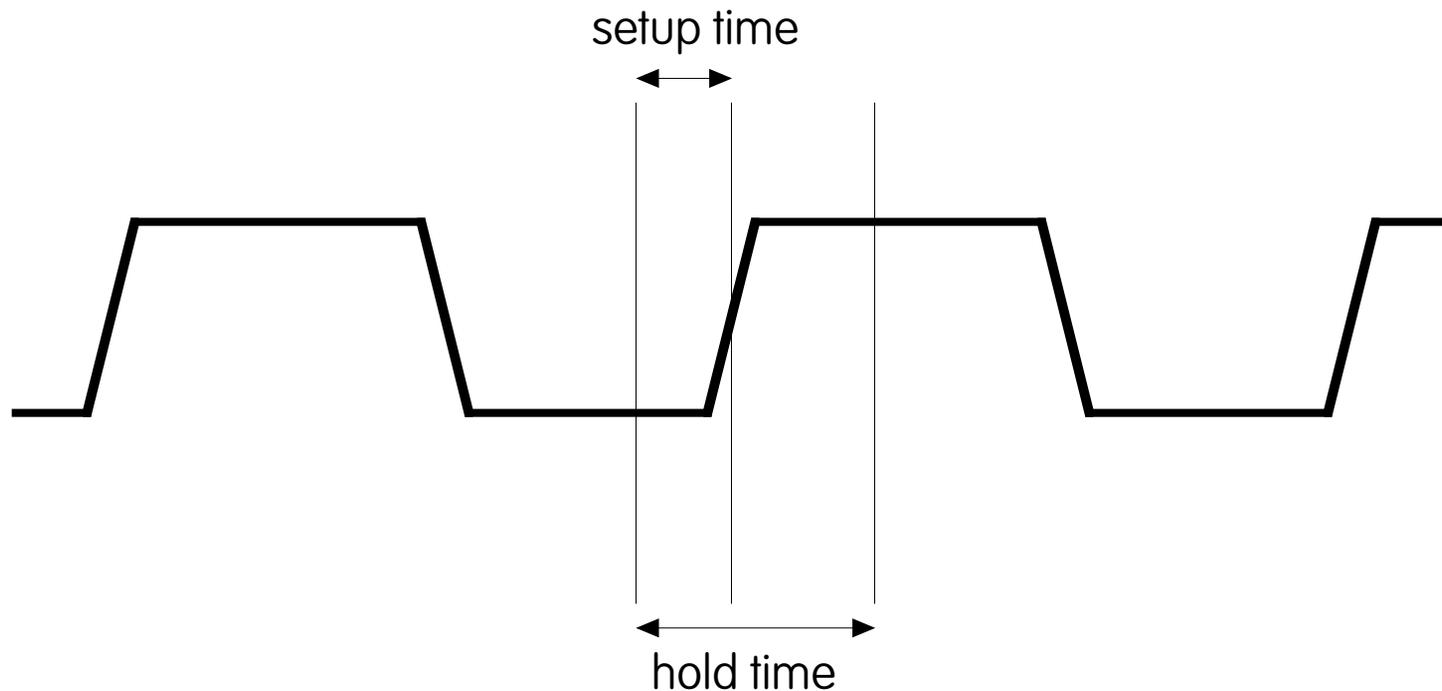
Circuit



Symbol

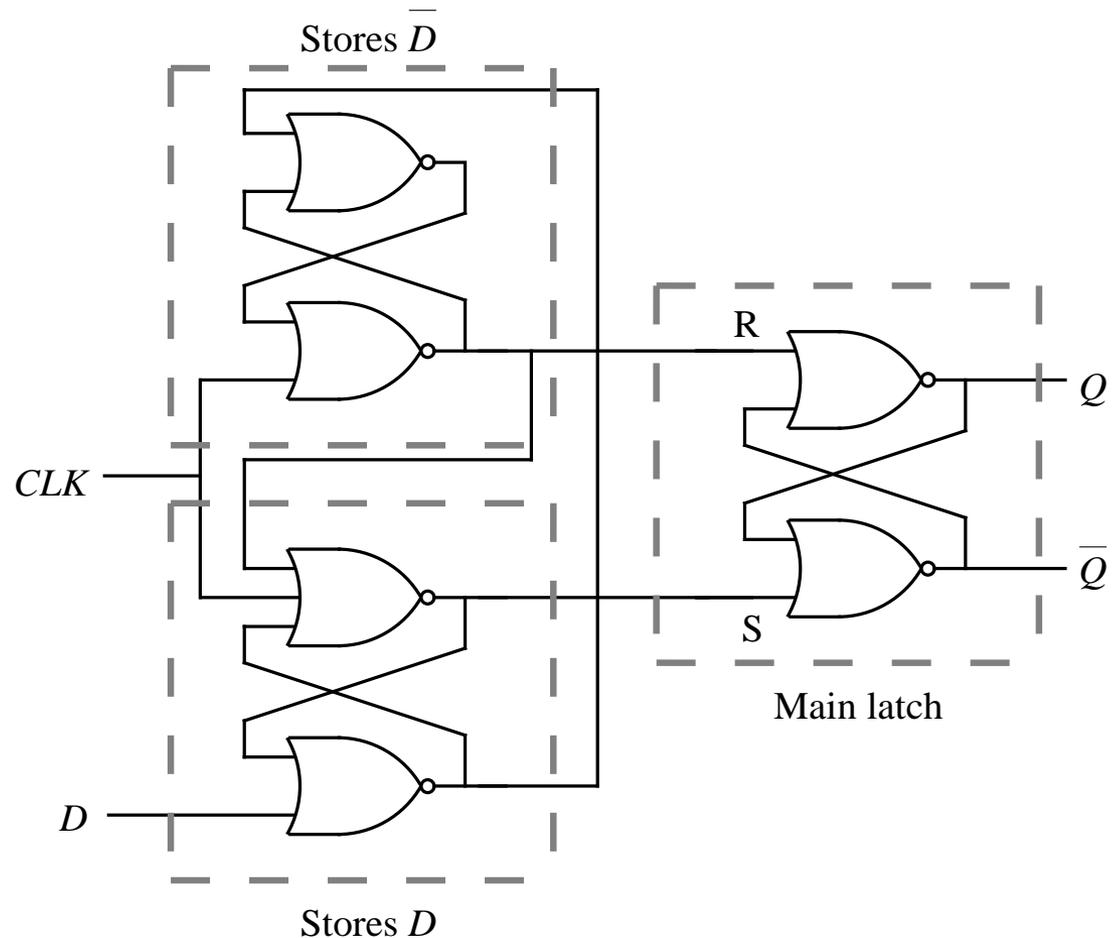
Edge-Triggered Flip-Flops

- Records input during a low-to-high (positive edge) or a high-to-low (negative edge) clock transition
- Signal must be stable before “setup time” and continue to be stable for “hold time”



Negative Edge-Triggered D Flip-Flop

- When the clock is high, the two input latches output 0, so the Main latch remains in its previous state, regardless of changes in D.
- When the clock goes high-to-low, values in the two input latches will affect the state of the Main latch.
- While the clock is low, D cannot affect the Main latch.

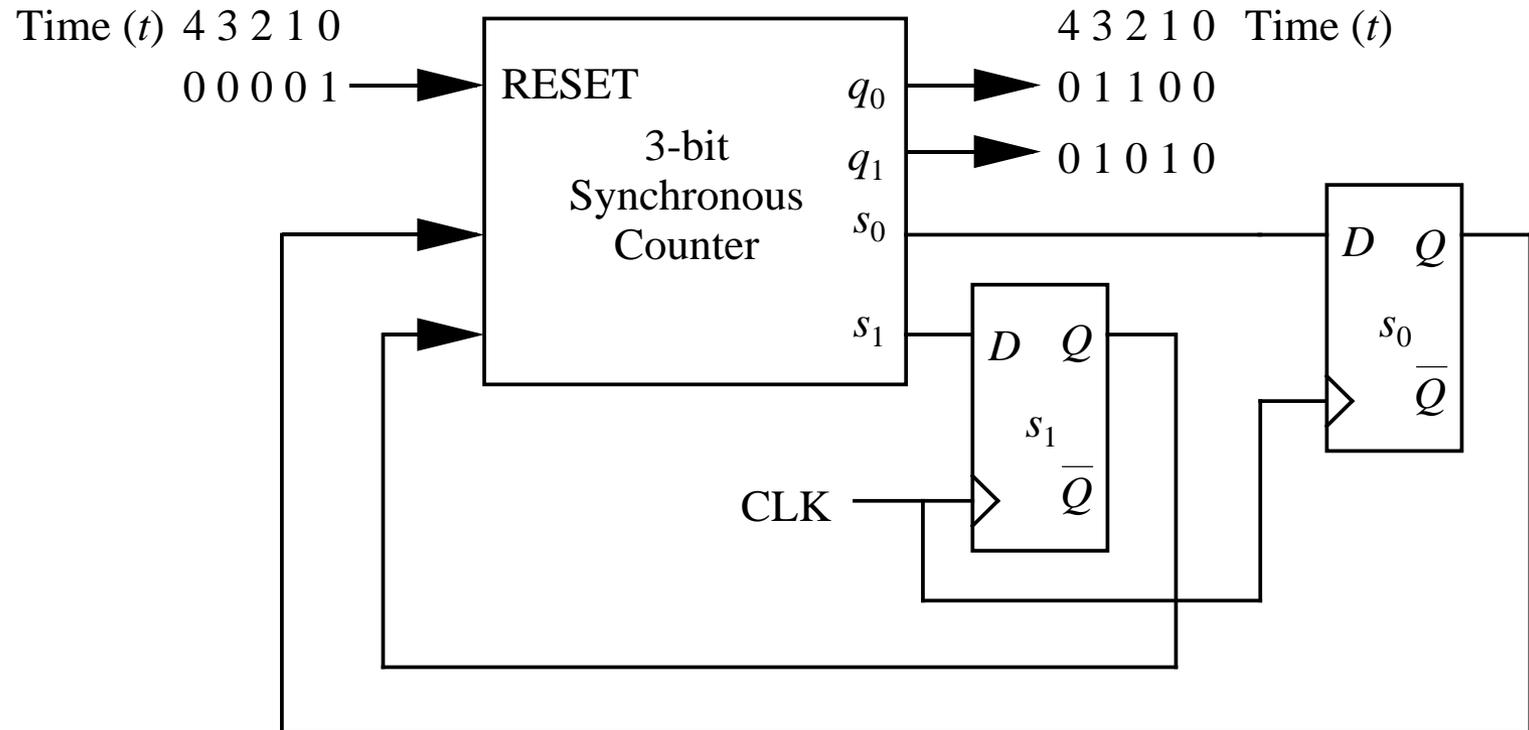


Master-Slave vs Edge-Triggered

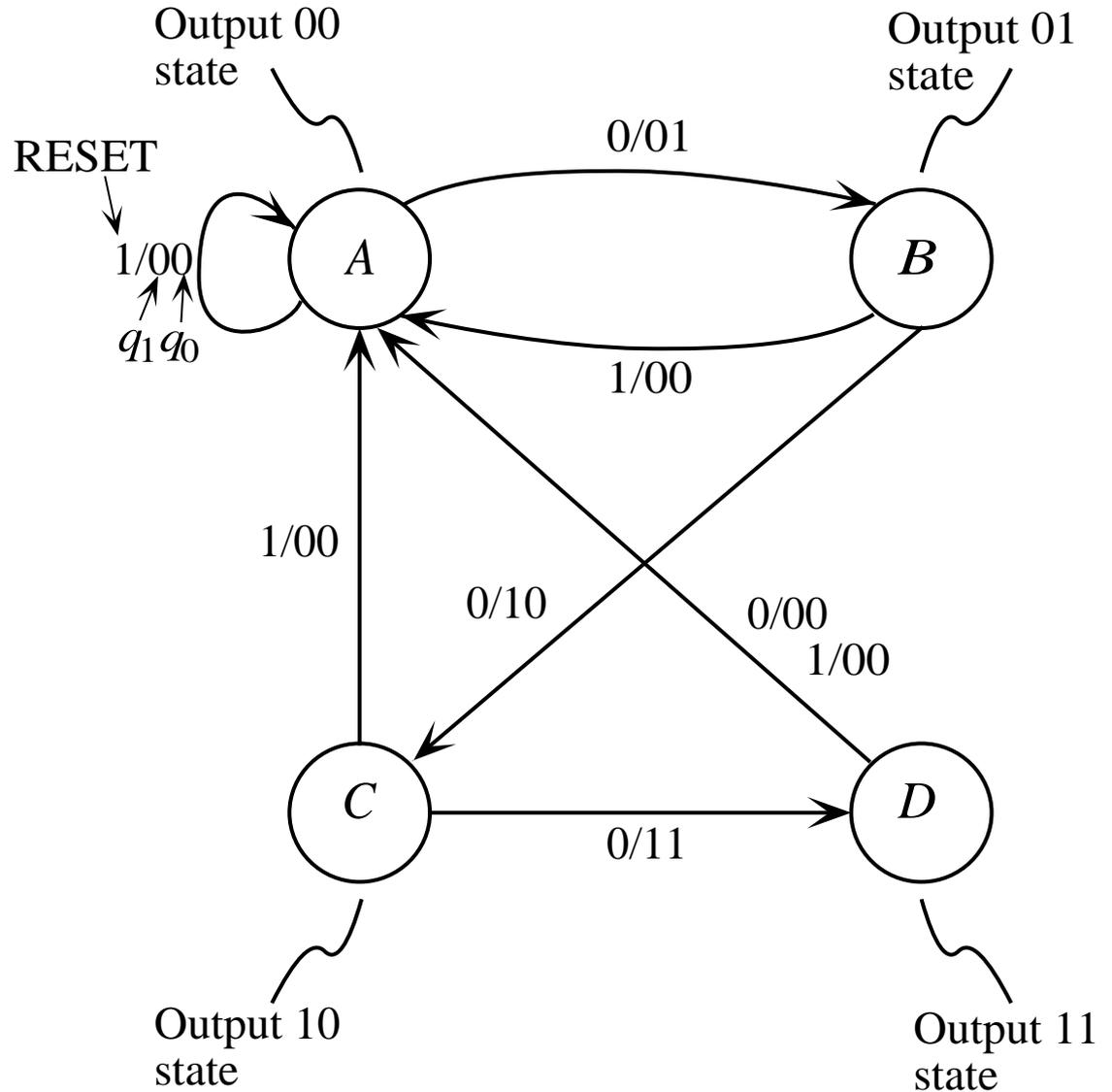
- **Master-slave flip-flops record the input in the slave when the clock goes from high to low**
- **Are master-slave flip-flops negative edge-triggered flip-flops?**
 - ◇ **Some textbooks say “yes” others say “no”**
 - ◇ **Master-slave JK flip-flops have the ones catching problem (momentary high signal at J when the clock is high is “caught” and recorded.)**
 - ◇ **Master-slave D flip-flops do not have the ones catching problem**

Example: Modulo-4 Counter

- Counter has a clock input (CLK) and a RESET input.
- Counter has two output lines, which take on values of 00, 01, 10, and 11 on subsequent clock cycles.



State Transition Diagram for Mod-4 Counter



State Table for Mod-4 Counter

Present state \ Input	<i>RESET</i>	
	0	1
<i>A</i>	<i>B/01</i>	<i>A/00</i>
<i>B</i>	<i>C/10</i>	<i>A/00</i>
<i>C</i>	<i>D/11</i>	<i>A/00</i>
<i>D</i>	<i>A/00</i>	<i>A/00</i>

Next state Output

State Assignment for Mod-4 Counter

Present state (S_t) \ Input	<i>RESET</i>	
	0	1
A:00	01/01	00/00
B:01	10/10	00/00
C:10	11/11	00/00
D:11	00/00	00/00

Truth Table for Mod-4 Counter

<i>RESET</i> <i>r(t)</i>	<i>s</i> ₁ (<i>t</i>)	<i>s</i> ₀ (<i>t</i>)	<i>s</i> ₁ <i>s</i> ₀ (<i>t</i> +1)	<i>q</i> ₁ <i>q</i> ₀ (<i>t</i> +1)
0	0	0	01	01
0	0	1	10	10
0	1	0	11	11
0	1	1	00	00
1	0	0	00	00
1	0	1	00	00
1	1	0	00	00
1	1	1	00	00

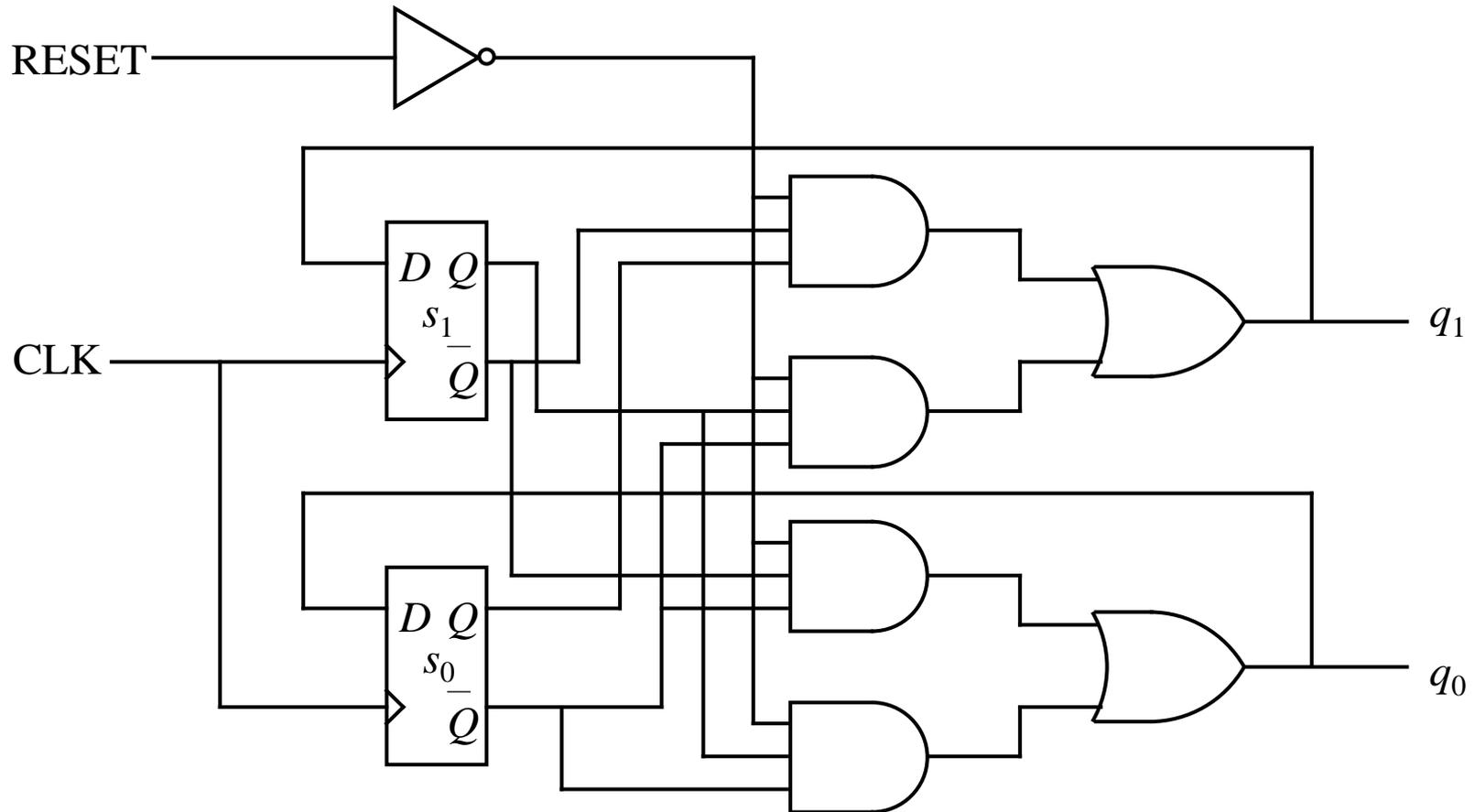
$$s_0(t+1) = \overline{r(t)}\overline{s_1(t)}\overline{s_0(t)} + \overline{r(t)}s_1(t)\overline{s_0(t)}$$

$$s_1(t+1) = \overline{r(t)}\overline{s_1(t)}s_0(t) + \overline{r(t)}s_1(t)s_0(t)$$

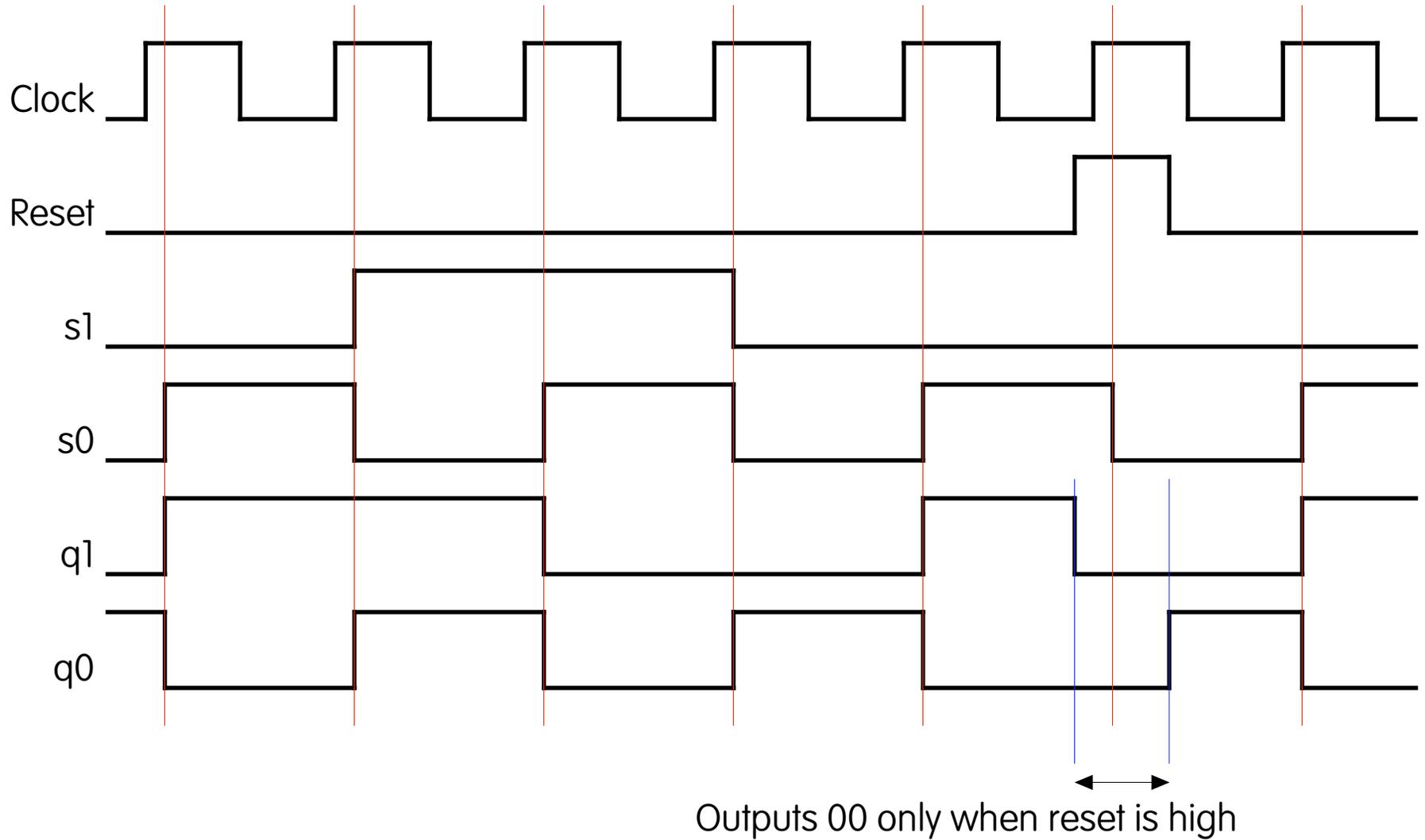
$$q_0(t+1) = \overline{r(t)}\overline{s_1(t)}\overline{s_0(t)} + \overline{r(t)}s_1(t)\overline{s_0(t)}$$

$$q_1(t+1) = \overline{r(t)}\overline{s_1(t)}s_0(t) + \overline{r(t)}s_1(t)s_0(t)$$

Logic Design for Mod-4 Counter

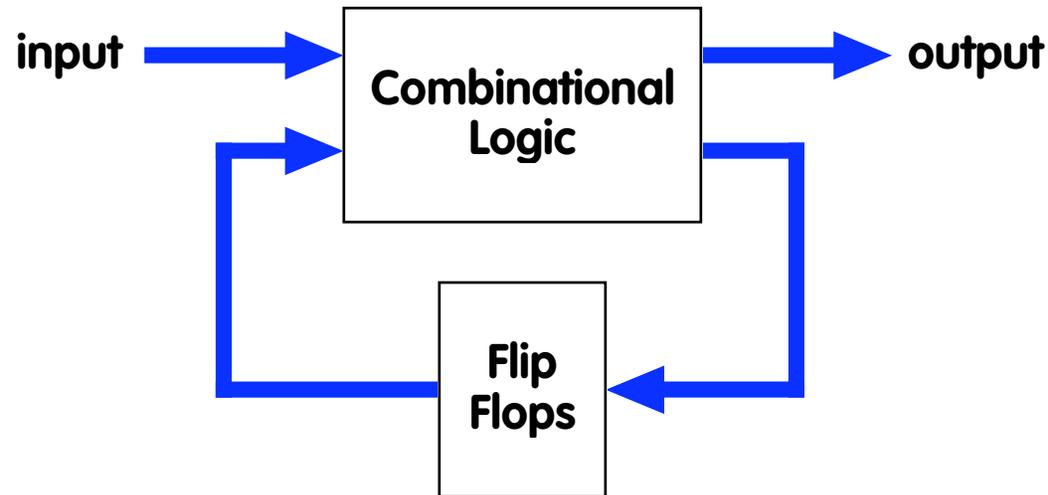


Mod 4 Counter Timing

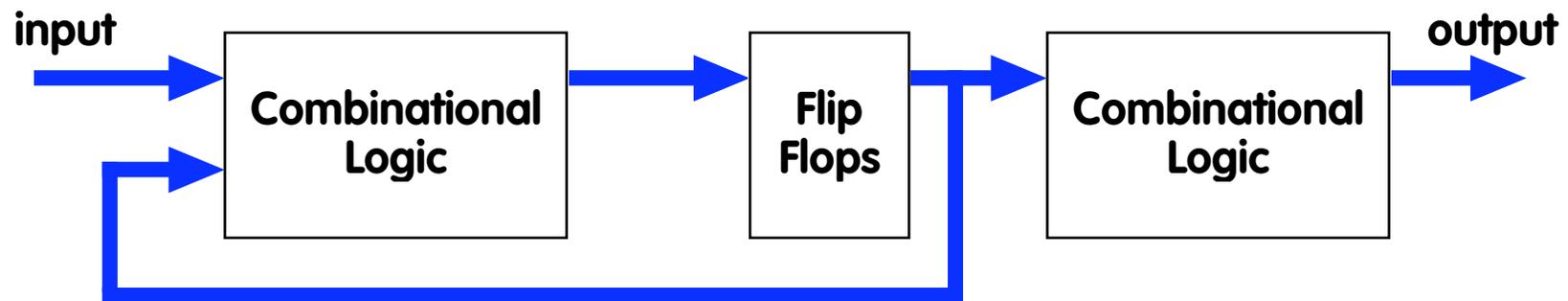


Mealy vs Moore Finite State Machines

- **Mealy: output depends on input and state bits**



- **Moore: output depends only on state bits**

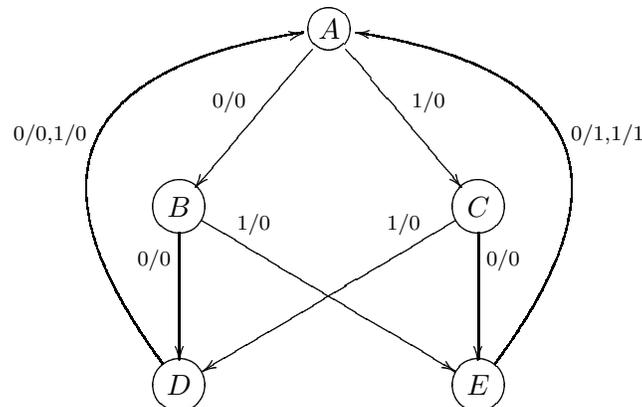


DigSim Assignment 2: A Finite State Machine

Due: Tuesday November 30, 2004

Objective: The objective of this assignment is to implement a finite state machine using DigSim.

Assignment: Consider the finite state machine from question A.34 of Murdocca & Heuring:



This finite state machine starts in state A and has one input bit and one output bit. The machine outputs 1 only after it has entered state E .

Your assignment is to implement this finite state machine in DigSim. You must:

1. Use three D flip-flops to store the 5 states of the machine. State A will be represented as 000, $B = 001$, $C = 010$, $D = 011$ and $E = 100$. The bit patterns 101, 110 and 111 are not used.
2. Let s_2, s_1, s_0 be the state bits stored in the D flip-flops, x be the input bit and z be the output bit. Fill in the attached truth table for the next state bits s'_2, s'_1, s'_0 and the output bit z .
3. For s'_2, s'_1, s'_0 and z , use Karnaugh maps to obtain simplified SOP or POS Boolean formulas.
4. Implement the finite state machine in DigSim. You should study the “Sequence Detector” example in DigSim (use “Open example” under the File menu) for suggestions on the layout of your finite state machine.

Implementation notes:

- Label the switches and flip-flops in your circuit appropriately.
- If you need a 4-input OR gate, you need to use two layers of 2-input or 3-input OR gates to accomplish the same function. Ditto for 4-input AND gates.
- The D flip-flops in DigSim are positive-edge triggered. To change the state of the flip-flop, change the input when the clock is low, then bring the clock from low to high. The input to the D flip-flop when the clock changes from low to high will be stored in the flip-flop.

What to submit:

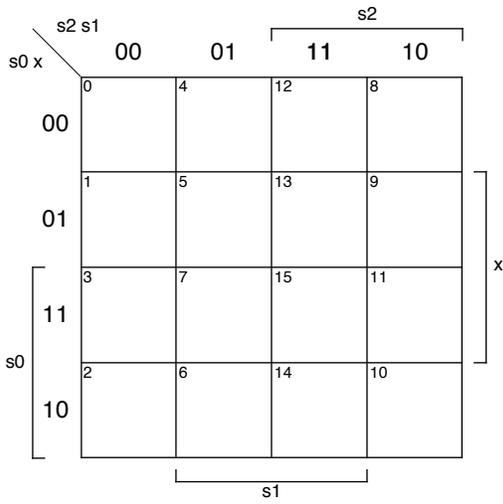
1. Make a copy of your truth-table and Karnaugh maps and submit the hard copy in class on Tuesday, November 30.
2. Save your circuit as you did in the DigSim Assignment 1. Submit the circuit file using the Unix `submit` command as in previous assignments. The submission name for this assignment is: `digsim1`. The UNIX command to do this should look something like:

```
submit cs313_0101 digsim2 a34.sim
```

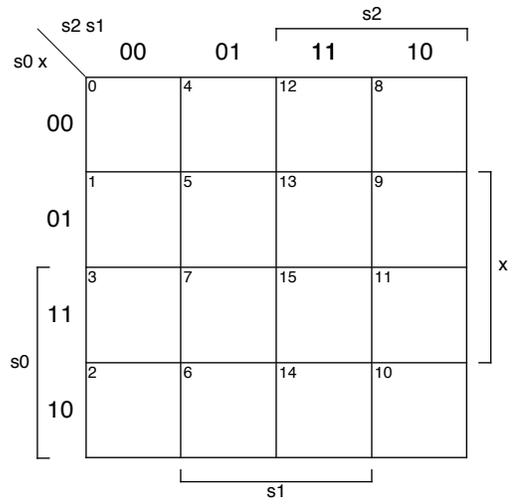
Name: _____

Truth table:

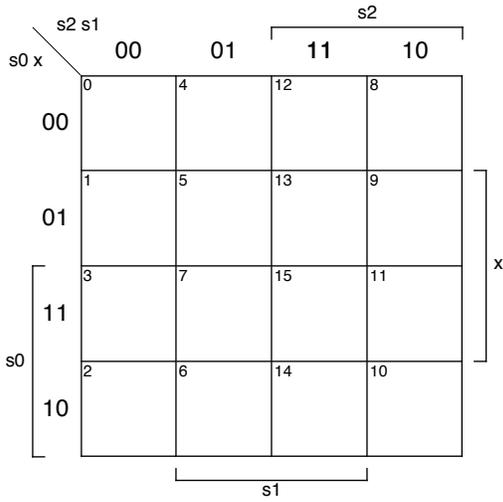
m	s_2	s_1	s_0	x	s'_2	s'_1	s'_0	z
0	0	0	0	0				
1	0	0	0	1				
2	0	0	1	0				
3	0	0	1	1				
4	0	1	0	0				
5	0	1	0	1				
6	0	1	1	0				
7	0	1	1	1				
8	1	0	0	0				
9	1	0	0	1				
10	1	0	1	0	d	d	d	d
11	1	0	1	1	d	d	d	d
12	1	1	0	0	d	d	d	d
13	1	1	0	1	d	d	d	d
14	1	1	1	0	d	d	d	d
15	1	1	1	1	d	d	d	d



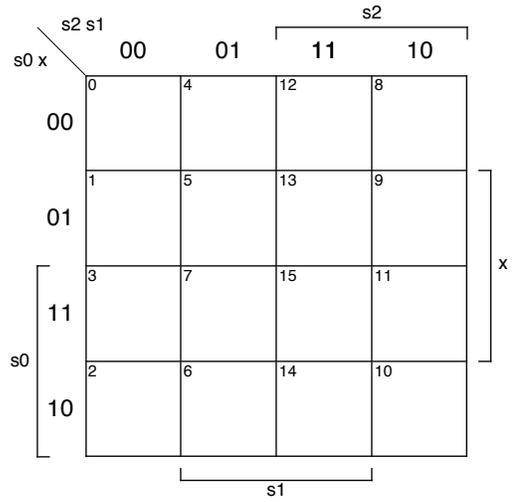
s2' =



s1' =



s0' =



Z =

Next Time

- **designing finite state machines**