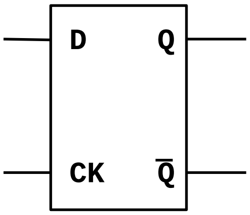
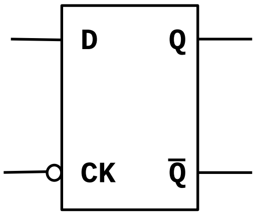
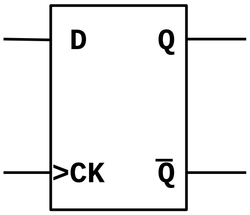
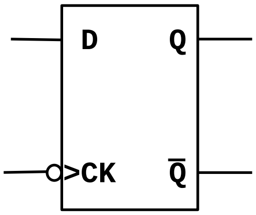
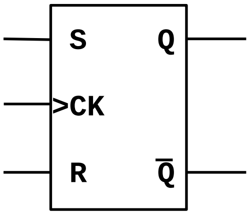
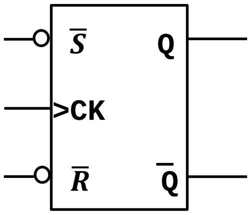
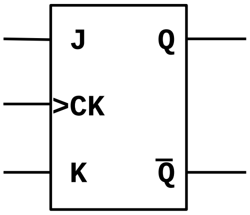
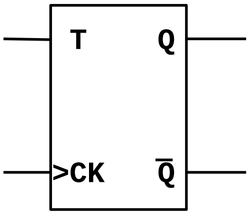
Lecture 8: Flip-Flops

1. Terminology

   1.1. “Level sensitive” = output controlled by the level of the clock input. “Edge triggered” = output changes only at the point in time when the clock changes from value to the other. Can be positive edge-triggered (0 to 1), or negative edge-triggered (1 to 0).

   1.2. “Flip-flops” are edge-triggered while clocked (gated) latches are level sensitive. The advantage of flip-flops over latches is that the signal on the input pin(s) is captured the moment the flip-flop is clocked, and subsequent changes on the input(s) will be ignored until the next clock event. This provides better timing control in complex circuits.

 **Positive D Latch**	 **Negative D Latch**	 **Rising Edge D Flip-Flop**	 **Falling Edge D Flip-Flop**
Can change when the clock is high.	Can change when the clock is low.	Can change when clock goes from 0 to 1.	Can change when clock goes from 1 to 0.

 **S-R Flip-Flop (NOR)**	 $\overline{S}$ Q >CK $\overline{R}$ $\overline{Q}$ **S-R Flip-Flop (NAND)**	 **J-K Flip-Flop**	 **T Flip-Flop**

S	R	*Q(t + 1)*
0	0	*Q(t)*
0	1	0
1	0	1
1	1	Undefined

S	R	*Q(t + 1)*
0	0	Undefined
0	1	1
1	0	0
1	1	*Q(t)*

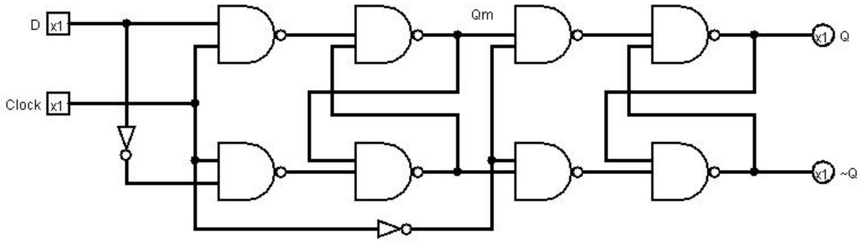
J	K	*Q(t + 1)*
0	0	*Q(t)*
0	1	0
1	0	1
1	1	$\overline{Q}(t)$

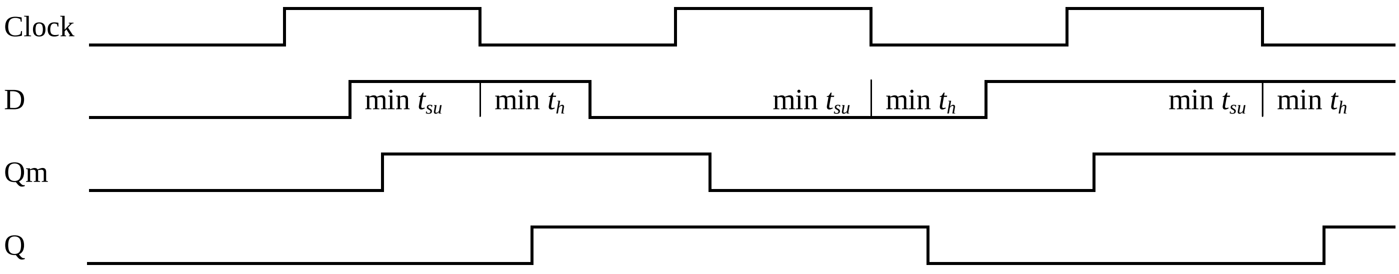
T	*Q(t + 1)*
0	*Q(t)*
1	$\overline{Q}(t)$

   1.3. For flip-flops, “Setup” time = t_{su} = the minimum time before the clock arrives (in below example goes from 1 to 0) that the inputs have to be stable and unchanging to ensure the first latch is stable, typically 0.3 ns. Needs to get through first latch NOT gate and clock NANDS.

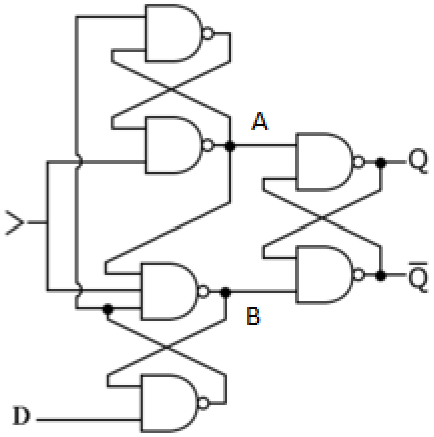
   1.4. For flip-flops, “Hold” time = t_h = the minimum time after the clock arrives that the inputs have to continue to be stable to and unchanging to ensure the first latch clock NAND is off. Not important for this course.

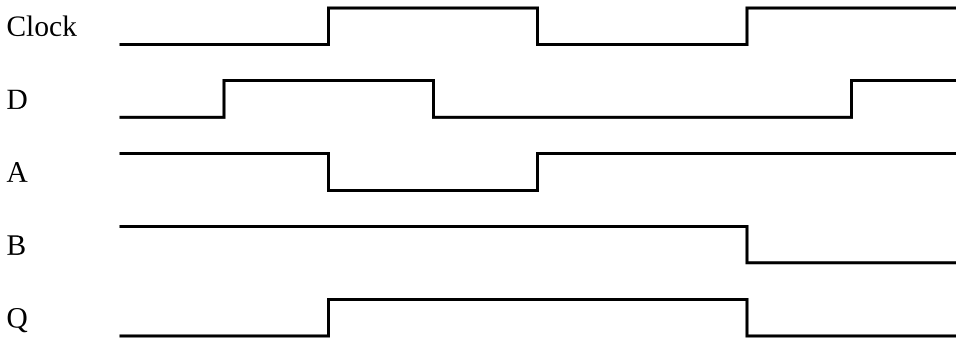
2. Falling-edge Triggered Master-Slave D Flip-Flop





3. Rising Edge Triggered D flip-flop





3.1. The second stage can be thought of as an S-R latch of made of NANDs. Can A be used to Set or Reset? How? ~A= Set

3.2. If A and B are both zero, then what happens? Q = ~Q = 1

3.3. In this circuit, can A and B both be zero? No. Why or why not? The wire from A to B's previous NAND ensures that if A is zero, B must be 1.

3.4. Note that A and B are always high when the clock is low.

3.5. Fill inn A, B, and the rest of Q.

4. Designing a T Flip-Flop (that toggles the output) from S-R Flip-Flops

4.1. Karnaugh Maps for S-R Flip-Flops and T Flip-Flops, where *Q* is the present state, and *Q'* is the next state.

Q'		***SR***			
		00	**01**	**11**	**10**
Q	**0**	0	0	d	1
	1	1	0	d	1

Q'		***T***	
		0	**1**
Q	**0**	0	1
	1	1	0

S		***T***	
		0	**1**
Q	**0**	0	1
	1	d	0

R		***T***	
		0	**1**
Q	**0**	d	0
	1	0	1

4.2. To solve this we note what SR combination(s) will create each T next state for each Q.

4.2.1. If *T* = 0 and Q = 0, then Q' = 0. If Q = 0 and Q' = 0, then *S* = 0, and *R* is in either state.

4.2.2. If *T* = 0 and Q = 1, then Q' = 1. If Q = 1 and Q' = 1, then *R* = 0, and *S* is in either state.

4.2.3. If *T* = 1 and Q = 0, then Q' = 1. If Q = 0 and Q' = 1, then *S* = 1 and *R* = 0.

4.2.4. If *T* = 1 and Q = 1, then Q' = 0. If Q = 1 and Q' = 0, then S = 0 and *R* = 1

4.2.5. Solving for *S* in terms of *T* and *Q* using the *S* K-map, $S = T\bar{Q}$

4.2.6. Solving for *R* in terms of *T* and *Q* using the *R* K-map, $R = TQ$

4.2.7. So to use an S-R flip-flop as a T flip-flop, we need to connect an AND of *T* and $\bar{Q}$ to the *S* input, and connect an AND of *T* and *Q* to the *R* input.

5. J-K Flip-Flop

5.1. The J-K flip-flop augments the behavior of the S-R flip-flop (J=Set, K=Reset) by interpreting the *S* = *R* = 1 condition as a "flip" or toggle command.

5.2. To create a J-K flip-flop from an S-R flip-flop, we'll create a truth table. The truth table starts with all the combinations of J, K, Q, and their resulting Q'. After filling the Q', we fill in the S and R that will create that Q' given the row's Q. Once the table is complete, generate the S K-map based on JKQ combinations and S value for each combination. Do the same thing for the R K-map. From the K-maps, *S* =

J	**K**	**Q**	**Q'**	**S**	**R**
0	0	0	0	0	d
0	0	1	1	d	0
0	1	0	0	0	d
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	d	0
1	1	0	1	1	0
1	1	1	0	0	1

S		***JK***			
		00	**01**	**11**	**10**
Q	**0**	0	0	1	1
	1	d	0	0	d

R		***JK***			
		00	**01**	**11**	**10**
Q	**0**	d	d	0	0
	1	0	1	1	0

5.3. Based on the S K-map, $S = J\bar{Q}$, and based on the *R* K-map, *R* = *KQ*. Again, the addition of two AND gates can transform an S-R Flip-Flop into a J-K flip-flop.