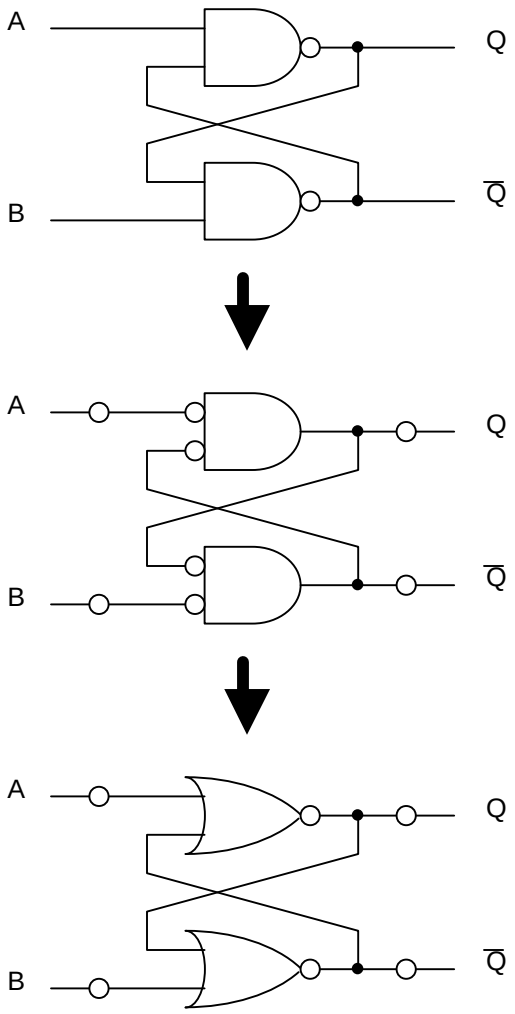
University of California at Berkeley
College of Engineering
Department of Electrical Engineering and Computer Sciences

EECS150 — J. Wawrzynek
Spring 2000 — E. Caspi

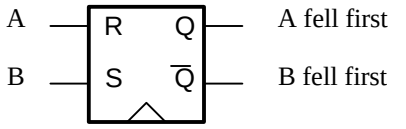
Homework #10 – Solution

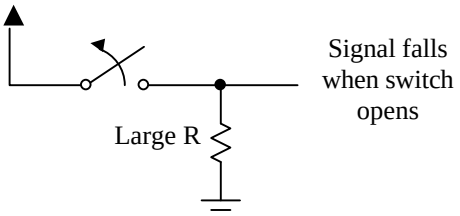
6.1 By pushing bubbles, we can convert the cross-coupled NAND pair into a cross-coupled OR pair (an RS latch!) with inverted inputs and outputs. Thus the cross-coupled NAND behaves like an RS latch with inverted inputs and outputs. Complementary inputs {0,1}, {1,0} set and reset the latch, respectively. High inputs {1,1} hold the present value. Low inputs {0,0} are forbidden.



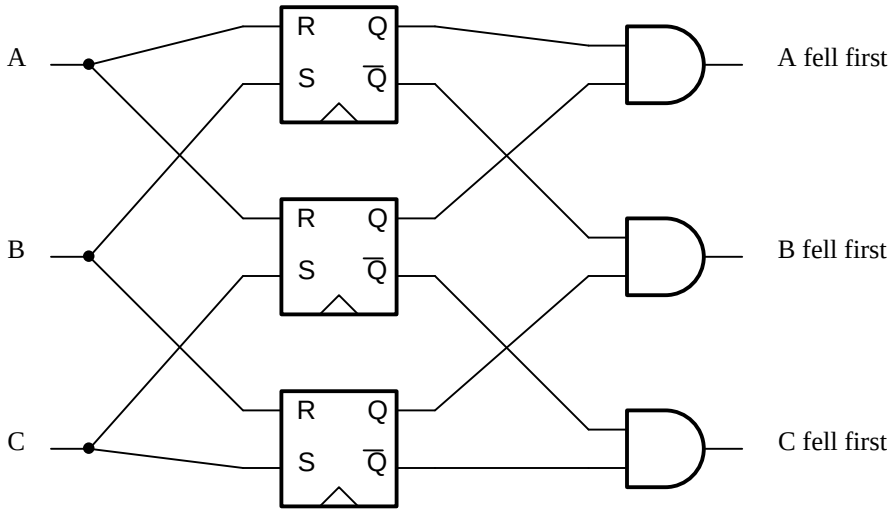
A	B	Q
0	0	unstable
0	1	1
1	0	0
1	1	hold

6.2 An RS latch can identify which of two signals falls 1→0 first. The signals are attached to inputs {R,S}. With both signals high, the output {Q,Q'} is {0,0} and stable. When one signal falls, the input {0,1} or {1,0} forces the RS latch to set or reset. When both signals have fallen, the input {0,0} makes the latch hold its output, "remembering" which signal fell first – {1,0} indicates that R fell first, and {0,1} indicates that S fell first. Katz actually asks us to detect the opening of a switch, but it is easy to convert that into the falling of a signal.





To identify which of three signals falls first, we can use 3 RS latches to perform 3 pairwise comparisons. Each signal participates in 2 pairwise comparisons. The first signal to fall "wins" both of its comparisons. The AND of a signal's 2 comparison results indicates whether it was the first to fall.

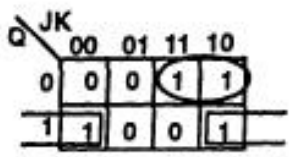


This method can be extended to identify which of *n* signals falls first. Each signal participates in *n*-1 pairwise comparisons, once with every other signal. The AND of a signal's comparison results indicates whether the signal was the first to fall. This method requires a total of *n*(*n*-1) pairwise comparisons (RS latches) and *n* (*n*-1)-input AND gates. The circuit area grows roughly quadratically with *n*.

Interestingly, this method can be extended to find the order of *n* falling signals. The number of pairwise comparisons won by a signal indicates its rank. The first signal to fall wins all *n-1* of its comparisons. The next signal to fall wins *n*-2 of its comparisons. The last signal to fall wins none of its comparisons. Thus, in place of AND gates, we can use adders to compute a signal's rank.

6.14

Implement a JK flip-flop from a D flip-flop

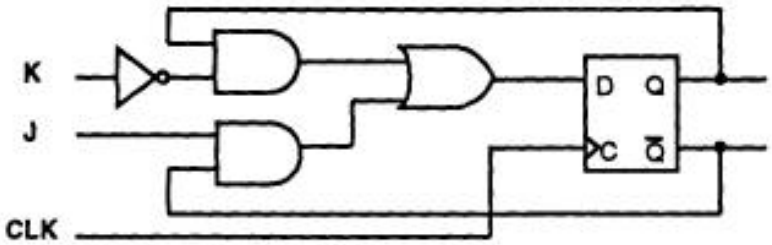


$D = J\overline{Q} + \overline{K}Q$

	J	K	Q	Q+	D
0	0	0	0	0	0
1	0	0	1	1	1
2	0	1	0	0	0
3	0	1	1	0	0
4	1	0	0	1	1
5	1	0	1	1	1
6	1	1	0	1	1
7	1	1	1	0	0

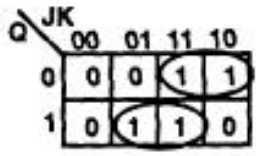
Excitation Table

Q	Q+	D
0	0	0
0	1	0
1	0	1
1	1	1



6.15

Implement a J-K flip-flop from a T flip-flop

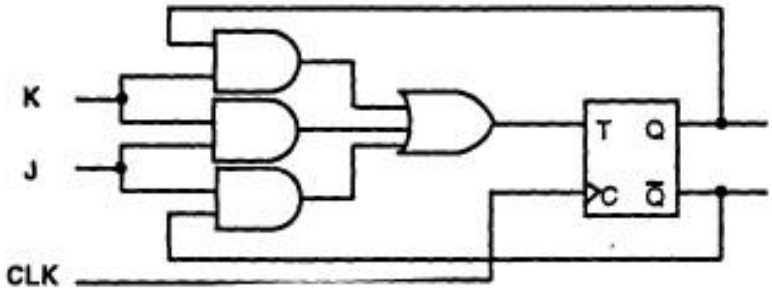


$T = J\overline{Q} + KQ$

	J	K	Q	Q+	T
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	0	0
3	0	1	1	0	1
4	1	0	0	1	1
5	1	0	1	1	0
6	1	1	0	1	1
7	1	1	1	0	1

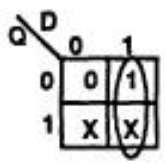
Excitation Table

Q	Q+	T
0	0	0
0	1	1
1	0	1
1	1	0

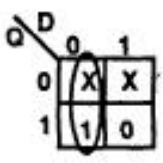


6.16

Implement a D flip-flop from a J-K flip-flop



$J = D$

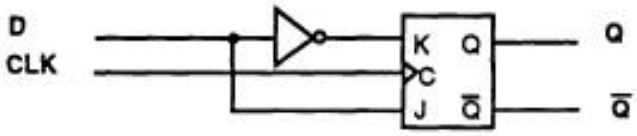


$K = \overline{D}$

D	Q	Q+	J	K
0	0	0	0	X
1	0	1	X	1
2	1	0	1	X
3	1	1	X	0

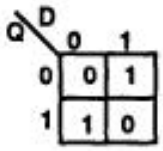
Excitation Table

Q	Q+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0



6.17

Implement a D flip-flop from a T flip-flop

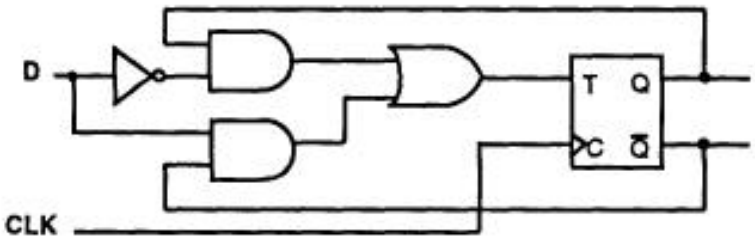


$T = D\,\overline{Q} + \overline{D}\,Q$

D	Q	Q+	T
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	0

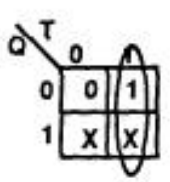
Excitation Table

Q	Q+	T
0	0	0
0	1	1
1	0	1
1	1	0

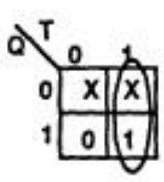


6.18

Implement a T flip-flop from a J-K flip-flop



J = T

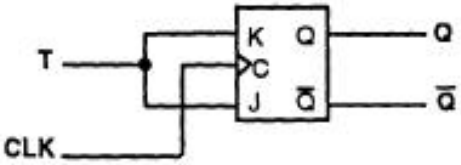


K = T

	T	Q	Q+	J	K
0	0	0	0	0	X
1	0	1	1	X	0
2	1	0	1	1	X
3	1	1	0	X	1

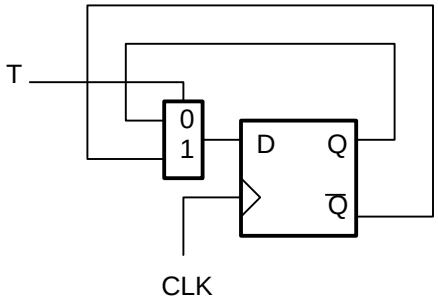
Excitation Table

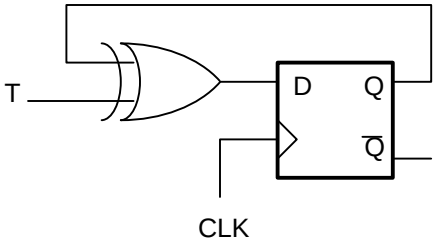
Q	Q+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0



6.19 Implement a T flip-flop from a D flip-flop.

Both implementations shown here are derived by inspection from the behavior of a T flip flop: T=0 retains Q, T=1 inverts Q.





Error Correction Codes (CRC)

The long division modulo 2 is as follows, yielding remainder 00000.
Recall that in modulo 2, addition and subtraction become XOR without carry or borrow.

```
               110001010
110101 | 101000110101110
         110101
         0111011
          110101
          00111010
            110101
            00111110
              110101
              00101111
                110101
                0110101
                 110101
                 0000000
```

A trace of the corresponding LFSR computation follows:

Time	Bit	FFFFF 43210
0		00000
1	1	00001
2	0	00010
3	1	00101
4	0	01010
5	0	10100
6	0	11101
7	1	01110
8	1	11101
9	0	01111
10	1	11111
11	0	01011
12	1	10111
13	1	11010
14	1	00000
15	0	**00000**