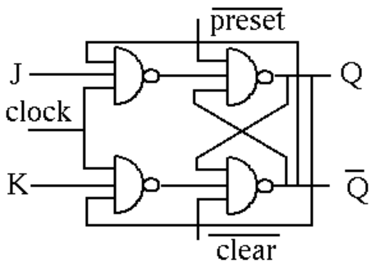
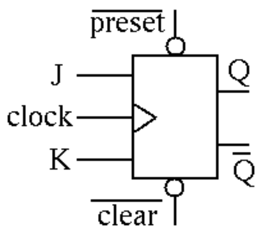
Homework 3

1. Verify the behavior of the JK-flipflop discussed in class (see background notes) by filling out the tables below. Make comment on each case in terms of whether there is a state change, and if so, whether it is a set (Q=1) or reset (Q=0).

Clock	J	K	Q_t	$\overline{Q}_t$	$\overline{set}$	$\overline{reset}$	Q_{t+1}	$\overline{Q}_{t+1}$	comment
0	x	x	x	x					
1	0	0	x	x					
1	0	1	0	1					
1	0	1	1	0					
1	1	0	0	1					
1	1	0	1	0					
1	1	1	0	1					
1	1	1	1	0					

Here $\overline{\text{set}}$ and $\overline{\text{reset}}$ are the outputs of the two NAND gates on the left with input J and K, respectively.





J	K	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	$\overline{Q}_t$

2. In class (see background notes) we built four types of clocked flip-flops (RS, JK, D, T) using the basic cell of two NAND gates. Now build these four types of clocked flip-flops using a basic cell (the latch) composed of two NOR gates connected exactly the same as the NAND basic cell described in class. Use the minimum number of any types of gates (AND, OR, NAND, NOR, etc.) for the rest of the circuit. Clearly label all input and output signals of each type of the flip-flops (e.g., the inputs S, R and Clock, and the outputs Q and $\overline{\text{Q}}$ of a RS flip-flop).

3. Assume the register file (RF) of a computer is formed by a set of n = 16 registers each composed of eight D-FFs.

   - How many bits are needed to address a register in the RF?
   - Design the circuit for reading the content (an 8-bit word) of an addressed register. The input of your circuit should include the addressing signals and a control signal *read* (set to 1 for read). The output is the 8-bit word read from the addressed register.
   - Design the circuit for writing an 8-bit word into an addressed register. The input of your circuit should include the address, a control signal *write* (set to 1 for write), and an 8-bit word to be written into the addressed register.

   Hints: For simplicity, you can show just one of the eight bits of your circuit, because other bits are the same. You should use combinational circuits such as MUX, DeMUX, decoder, ROM, etc., to simplify your design. As always, you need to minimize the number of additional gates. Take a look at these additional notes.

4. Convert a T Flip-Flop to a JK Flip-Flop.

5. Convert an RS Flip-Flop to a T Flip-Flop.

6. An *even parity checker* is a sequential logic circuit that counts the number of `1`s in a bit-serial input stream and outputs `1` whenever this number is even (including 0). For example, if the input stream is `0110100101`..., the corresponding output stream is `1011000110`.... Implement such an even parity checker as a finite state machine using (1) RS-FFs, (2) JK-FFs, (3) D-FFs and (4) T-FFs. Draw the logic diagram for each of the implementations. Which one requires minimum combinational logic?

7. Design an FSM for a subtracter which takes two bits a_i and b_i from two binary numbers $A = (a_{n-1}a_{n-2}\ldots a_1a_0)$ and $AB = (b_{n-1}b_{n-2}\ldots b_1b_0)$ as inputs and generates an output in the difference $D = (d_{n-1}d_{n-2}\ldots d_1d_0) = A - B$.

   Follow the steps to create a state diagram, a state table and a final gate level design. Use D type flipflops.

   Hint: For simplicity, we assume *A > 0*, *B > 0*, and *A > B*. In case you are not familiar with binary subtraction, think what you do in decimal subtraction and go through the example here:

```
    1  1  0  1  0  0  0  1  1  0
 -  0  1  1  0  0  1  1  0  1  0
--------------------------------
```

   As in decimal subtraction, sometimes you may need to borrow from the next higher bit on the left.