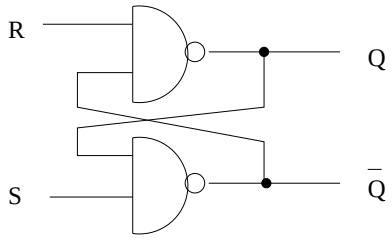
Questions

1. The figure below shows an RS latch made out of NAND gates (rather than NOR gates). How do Q and $\overline{Q}$ depend on the RS inputs? i.e. verify that the circuit can indeed be used as a RS latch.



2. Consider two types of RS latches:

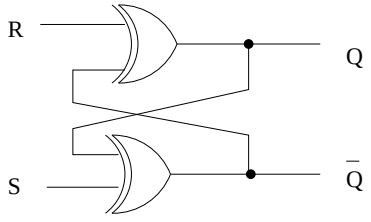
   (a) an RS latch made out of NOR gates, as seen in class

   (b) an RS latch made out of NAND gates, as in question 1.

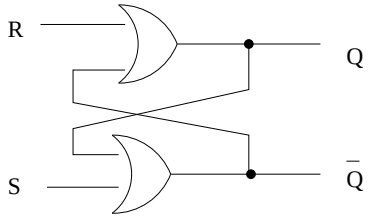
   For each type of latch, give an example of R and S timing diagrams that could produce the following timing diagram for Q:

   Use a "hold" signal to hold the values after Q changes.

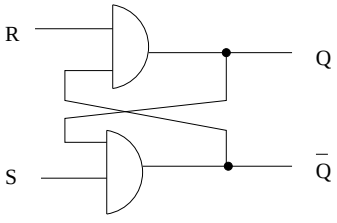
3. Consider the three circuits below, defined by XOR, OR, AND, NAND gates. In each case, argue whether or not the circuit has essentially the same behavior as the RS latch we saw in class (made of NOR gates), namely whether you can use it to *hold*, *set*, and *reset* Q.



(a)



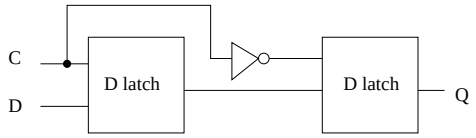
(b)



(c)

4. Draw a complete circuit diagram of a flip-flop that uses only AND, OR, and NOT (inverter) gates.

5. Consider the falling edge triggered D flip-flop:



   (a) Suppose we replaced the first D-latch of the flip-flop by an AND gate. Would this circuit be equivalent in behavior to the D flip-flop ? If not, then why not?

   (b) What if we instead replaced the *second* D-latch in the flip-flop by an AND gate. Would this circuit now be equivalent to the above D flipflop? If not, then why not?

6. (challenging)

   In class we saw a counter circuit. How would you modify this circuit to design a "mod 6" counter, that is, a counter whose Q outputs at successive clock cycles represent the binary code of

   $$\ldots,\ 0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 0,\ 1,\ \ldots?$$

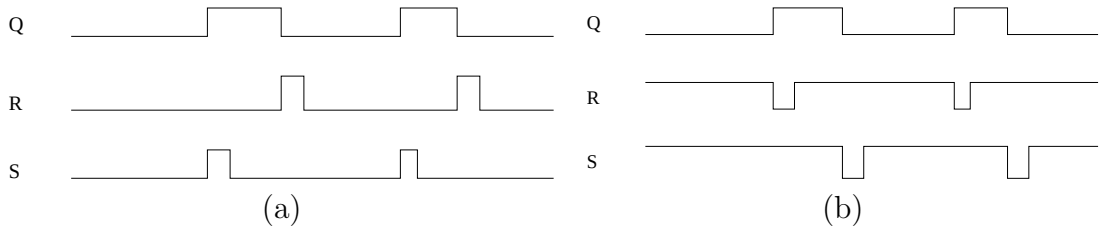
7. (challenging)

   Describe a circuit that converts a 32 bit unsigned integer to a single precision IEEE floating point number. (An approximation may be necessary in some cases.) Do not draw the circuit. Instead just describe what elements would be needed (registers, counters, adders, etc) and what they would do and in an order that makes sense.

Solutions

1. If RS is 01 then $Q = 1$ and $\overline{Q} = 0$ and if RS is 10 then $Q = 0$ and $\overline{Q} = 1$. To hold $Q\overline{Q}$ at either 01 or 10, you use $RS = 11$. When RS is 00, we have $Q\overline{Q} = 11$. This case is not permitted (for the same reason that RS = 11 was not permitted for the RS latch made of NOR gates).

2.



3. (a) XOR: If $R = 0,\ S = 1$ or if $R = 1,\ S = 0$ then the circuit is not well-defined. For example, suppose $R = 0,\ S = 1$. If Q=0, then $\overline{Q}$ would take the value 1, which would cause Q to take the value 1, which would cause $\overline{Q}$ to take the value 0, etc. Obviously the same problem happens if $R = 1,\ S = 0$.

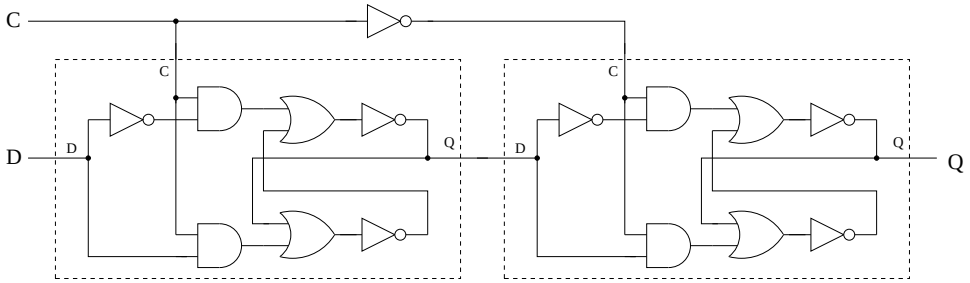
   What if $R = 0,\ S = 0$? If Q=1, then $\overline{Q}$ would be 1 also, whereas if $Q = 0$ then $\overline{Q}$ also would be 0. What if $R = 1,\ S = 1$? If Q=1, then $\overline{Q}$ would take value 0, and we would be stable, and if Q=0, then $\overline{Q}$ would take value 1, and again we would be stable.

   Note there is no way to hold a signal, so this circuit cannot be used as an RS latch.

   (b) OR: if either R or S is 1, then $Q = \overline{Q} = 1$ and there is no way to get them back to 0.

   (c) AND: if either R or S is 0, then $Q = \overline{Q} = 0$ and there is no way to get them back to 1.

4. All I wanted you to do in this question is to take the diagram of the D flip-flop, which has two D-latches, and expand the D-latches into their underlying logic gates. The only other thing you need to do is to replace the NOR gate with an OR gate followed by an inverter (since the question says that you are only allowed to use AND, OR, and inverter gates).



5. (a) If the first D latch were replaced by an AND gate, then when C = 1, this AND gate would pass through the value D, which is correct. This value wouldn't enter the second D latch, however, because the C input to the second D latch would be 0 (because of the inverter). When C becomes 0, the output of the AND gate changes to 0 and the value 0 is written into the second D latch. Hence, the output Q is always 0. (Note that I am ignoring what happens in the small transition times, that is, while C and D are changing.)

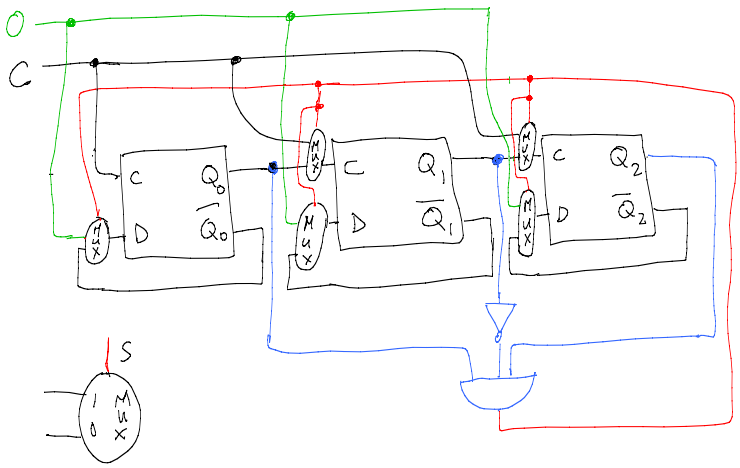
   (b) If the second D latch were replaced by an AND gate, then when $C = 0$ the output Q would be identical to the output of the first D latch. Moreover, if D is changing during the $C = 0$ phase, then the output Q would change with it. This is not how a D flip flop should work.

   When $C = 1$, the output of the AND gate would always be 0 (because the C input to the AND gate would be 0), and hence the output of the circuit Q would be always 0 (not D). This is not the behaviour we want for a flip flop.

6. For a "mod 6" counter, we need to force (Q_2, Q_1, Q_0) to make a transition from 5 to 0 rather than from 7 to 0, that is, we need to make a transition from (1,0,1) to (0,0,0) rather than its usual transition from (1,0,1) to (1,1,0). We can do so by adding a multiplexor in front of the D input of each T flip-flop. In the case that $(Q_2, Q_1, Q_0) = (1, 0, 1)$, the multiplexors select the 0 input. Otherwise, the multiplexors select the usual input, namely the $\overline{Q}$ of the flip-flop, which achieves a toggle. (As indicated by the the multiplexor in lower left corner, the 1 selector selects the upper input.)

   We also need to add a multiplexor in front of the clock inputs. If $(Q_2, Q_1, Q_0) = (1, 0, 1)$, then each of the flip-flops sees the true clock as its input. In all other cases, each flip-flop sees the same clock signal as the original counter did.

   Note that the flip-flops are falling edge triggered, as in the counter circuit in lecture 6.



You may be wondering why we need multiplexors on the clock inputs. Doesn't the circuit work fine if the clock inputs are the same as in the original counter circuit ? Note that the flip-flops are falling edge triggered and they all need to be reset to 0 on the clock cycle after the Q variables have values (1,0,1), that is, when the normal counter would turn from values 5 to 6. To see why, draw the timing diagram. When the Q values are (1,0,1), the multiplexor selects the real clock C to be fed into the C input for each flip-flop (and selects the data input of each flip-flop to 0). If you don't have a multiplexor on the C input, then Q_2 will not get a falling edge clock input (from Q_1) and so Q_2 will not change its value. Q_2 will stay at 1 rather than going to 0.

In fact, the selector in front of A1 is not necessary. Both Q_0 and the real clock (sometimes called CLK) have a falling edge at the end of the (1,0,1) state and so it doesn't matter which of these C signals is selected by Q_1, and there doesn't need to be a selector there. The reason I put in the selector is to make the circuit more general: If you wanted a mod n circuit for some other n, you could achieve this by changing only the condition on the AND gate at the bottom of the drawing.

7. The solution below has many details. I am not expecting you to get all these details on your own. I am expecting, rather, for you to think about what needs to be done. If you were doing the conversion from int to float, how would you manipulate the bits ? What operations would you need to do?

   Use an 8-bit timer T, a 32-bit left-shift register I which will hold the input, and a 32-bit register F to hold the final result.

   - Check whether the input is 0. If this is the case, copy the floating-point representation of 0 to F and we are done. Otherwise....
   - Since the int is *unsigned*, we know the number is positive and therefore that the sign bit of the float will be 0. Set the float's sign bit (bit 31) to 0.
   - Copy the unsigned integer input to a register I.
   - Initialize the timer T to $127 + 32 = 159$. This will be used to compute the exponent. See next step.
   - You want to find what is the most significant bit of the int which has the value 1, since this will serve as the 1 bit to the left of the binary point in the normalized representation. To find the most significant 1 bit, left-shift the contents of I and decrement the timer T with each clock pulse, until the bit shifted out is a 1. (Fill the LSB of I with 0's when shifting.)
   - Copy the significand from the high-order 23 bits of I to the lower order 23 bits of the output register F.
   - Copy the exponent from T to bits 22 through 30 of F.