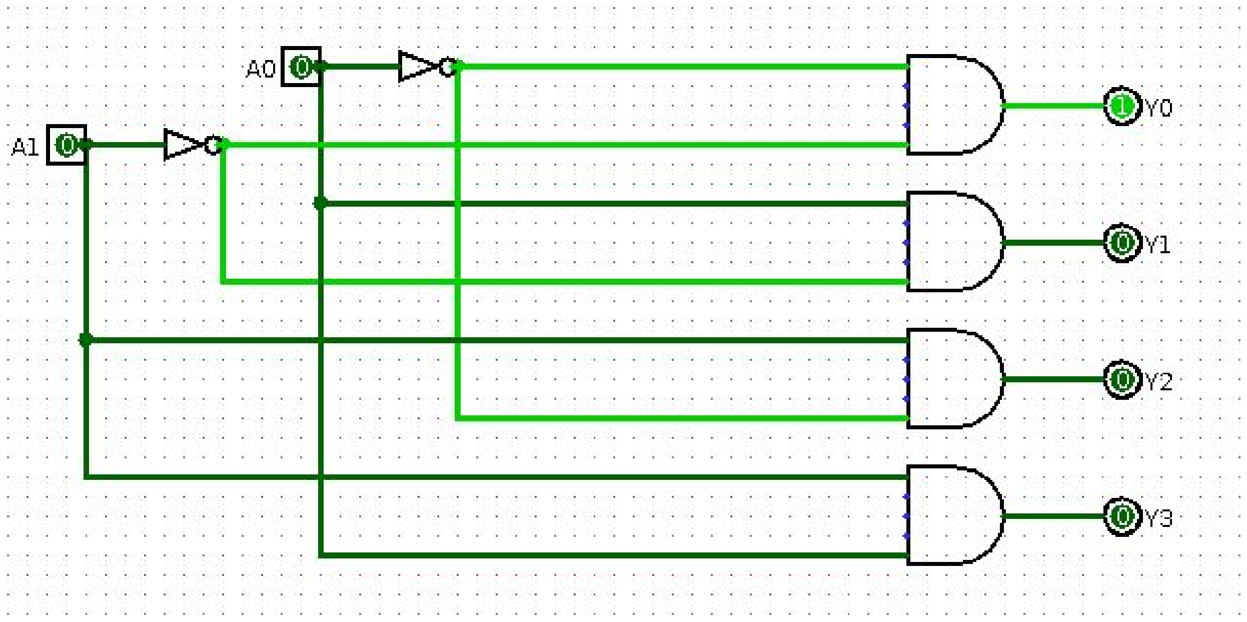
William Berg
willb@kth.se

LD-LAB IS1500

Assignment 1: Decoder

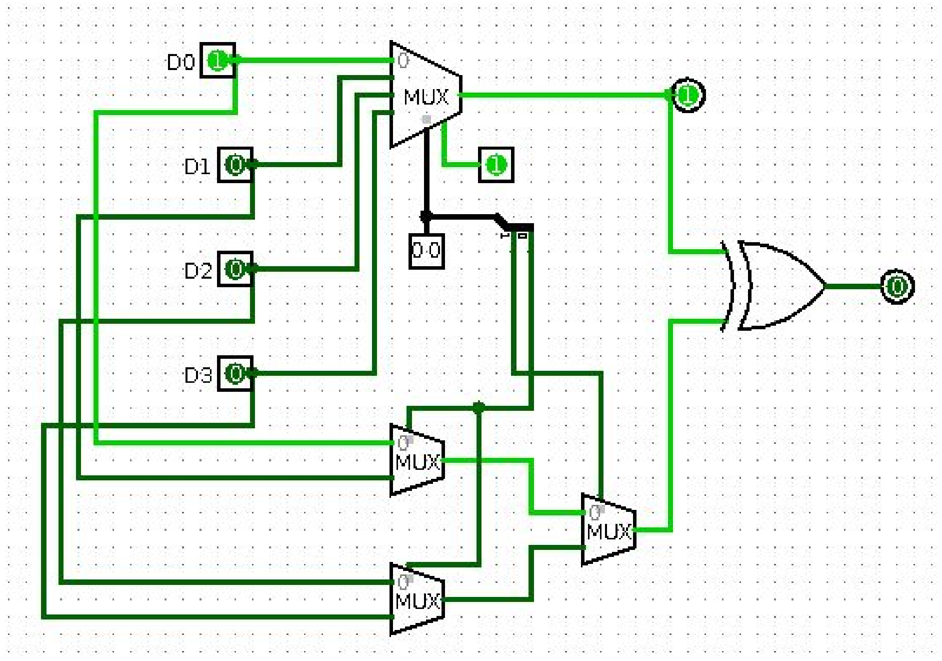
I have verified that the design was correct by testing all the different input values of A0 and A1 (nudging in Logisim) to see if each distinct combination asserts exactly one unique output.



William Berg
willb@kth.se

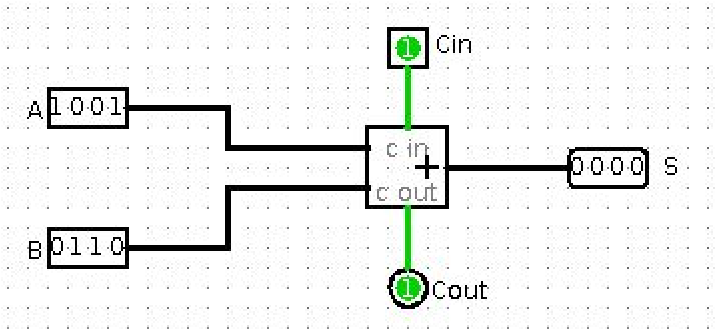
Assignment 2: Multiplexer
Task 2.1. MUX out is blue because the wire carries a one-bit value (it has a bit width of 1), but it is not carrying a specific value at this time. It is what is called a floating bit (not 1, not 0).
Output signal “Equal” is red because the wire is carrying an error value. This is because the gate cannot determine the proper output seeing as it has no inputs.
The select signal of the 4:1 multiplexer is black because the wire is carrying a multi-bit value (in this case we need two bits to choose between four inputs).

Task 2.2. If the output signal “Equals” is a 0, that means that the same input signal has been passed through the output of both of the multiplexers (the 4:1 multiplexer provided by Logisim and the one we made with three 2:1 multiplexers). If it is a 1 that means that they are passing through different inputs and the component is faulty. This is because XOR outputs 1 if the two inputs are different and 0 if they are the same. Since the output of the XOR gate is 0 for all of the different inputs, we know that the implementation is correct.

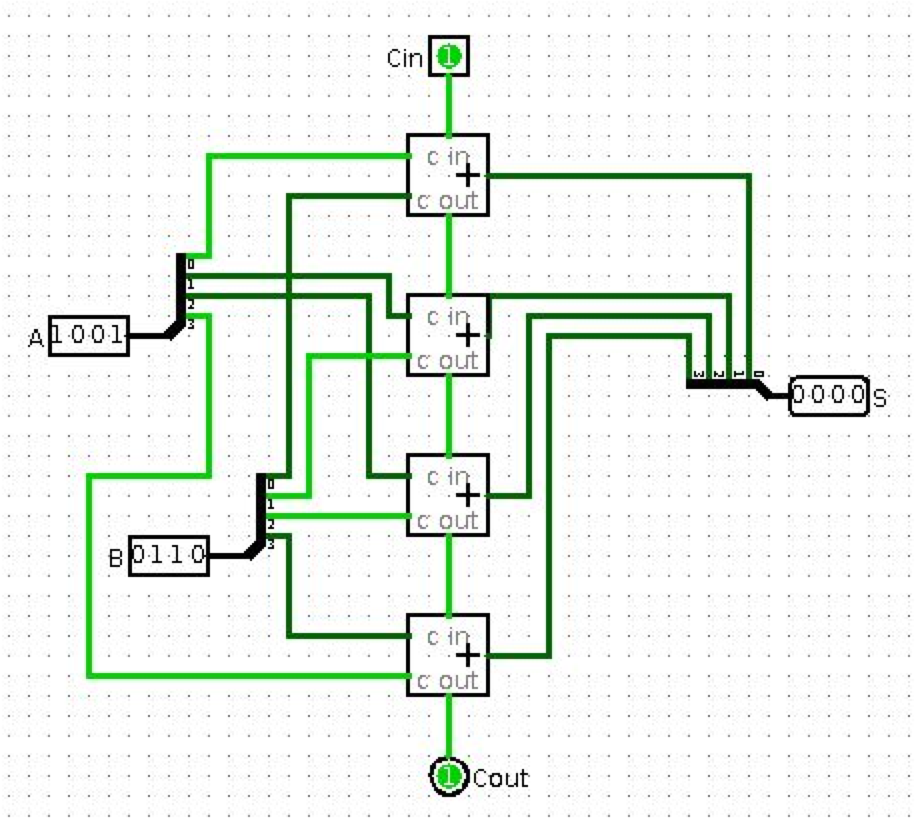


William Berg
willb@kth.se

Assignment 3: Adder
Task 3.1.

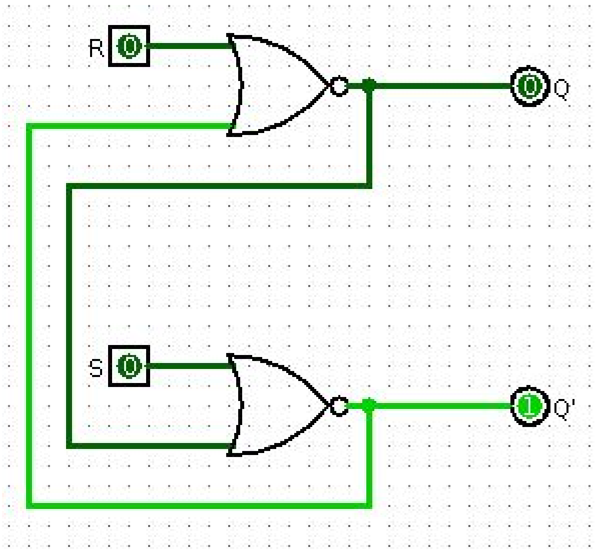


Task 3.2.

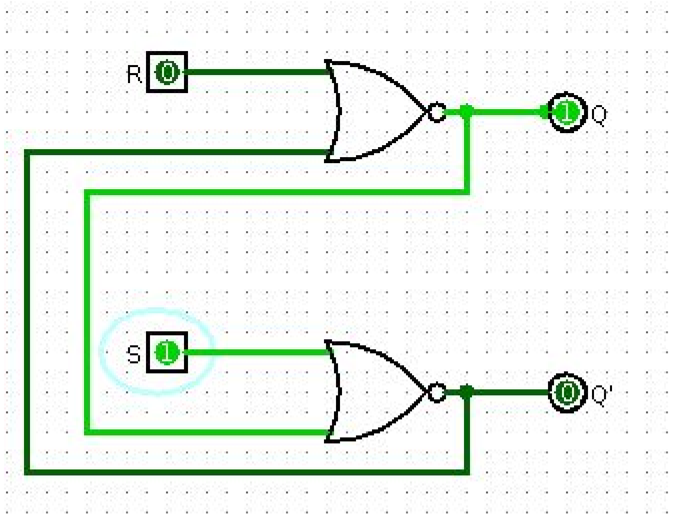


William Berg
willb@kth.se

Assignment 4: Latches
Task 4.1.



Q can be either 1 or 0, as can Q’. This is because when R and S are both 0, the previous values of Q and Q’ are remembered (the value of Q and Q’ does not change when R and S are both set to 0). Since the previous value of Q could have been either 1 or 0, the possible values for Q and Q’ is 1 or 0 - they will however always be opposites.

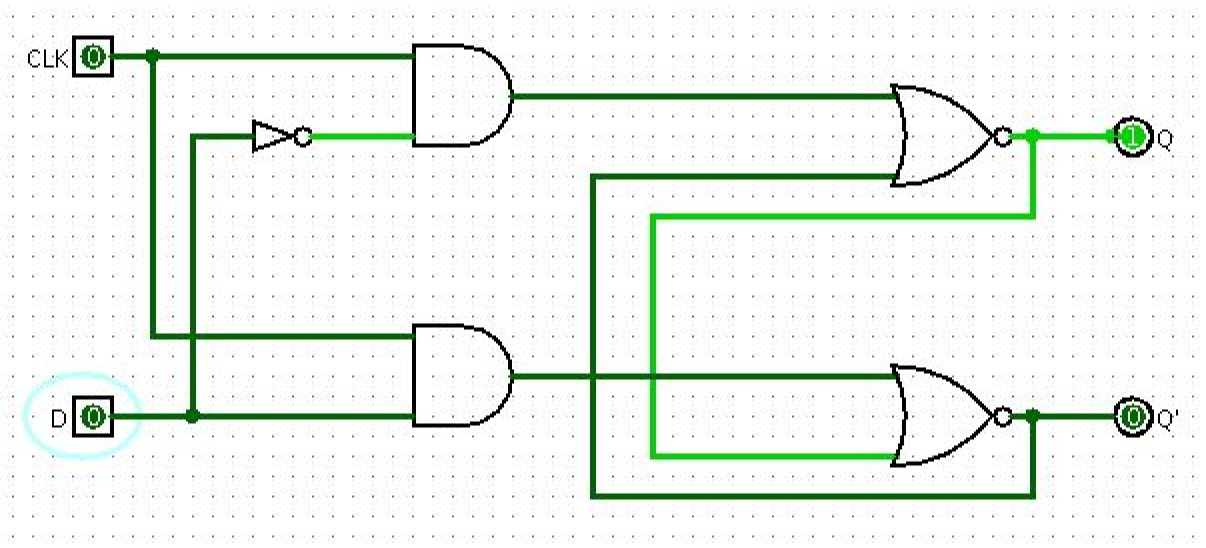


When S is 1 and R is 0, the possible values for Q and Q’ are only 1 and 0 respectively (Q = 1 and Q’ = 0). When the S switch is toggled, the values for Q and Q’ do not change; this is the value of the SR latch - the ability to remember the previous values of Q and Q’ when R and S are both 0. This is possible because when the value of S is toggled, the output of the NOR gate it is connected to does not change (since the output of the NOR gate is only 1 if both of the inputs to it are 0).

William Berg
willb@kth.se

If S is equal to 0 and R is toggled between 1 and 0, the value of Q and Q’ will remain the same at Q = 0 and Q’ = 1. Toggling does nothing since when both R and S are 0, the Q and Q’ values are remembered.

Task 4.2.



If D is equal to 0 and you toggle the clock, the value of D (which is zero) will flow through to Q and Q will equal 0. When the clock is 1, Q will always be equal to the value of D and when the clock is 0, Q will always be equal to the previous value of D. So in this example, the value of Q will always be 0 because we never change the value of D to set the value to anything else.
If CLK is equal to 0 and you toggle D, the value will never change (it will retain the previous value from when CLK was 1). This is because when the CLK is 0, the latch is opaque and the value in D will never flow through to Q.

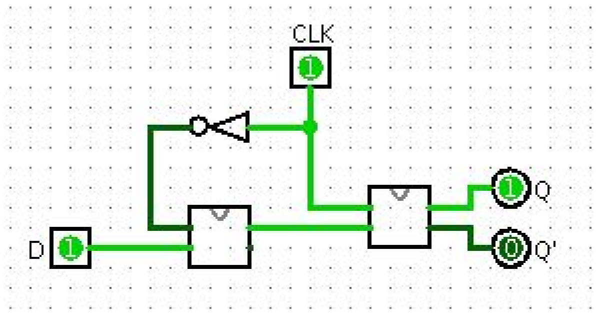
If CLK is 1 and you toggle D, the value of Q will be updated on every toggle of D. This is because the D latch is level triggered and Q will update to the D value as long as the CLK is turned on (set to 1).

The D latch is better than the SR latch because there is no longer any strange behavior when both S and R are set to 1 (in the SR latch this would render Q and Q’ equal which is not what we want). It also separates the questions of what (what value should Q be) and when (when should the value change). The D latch has two separate inputs (D and CLK) to answer these questions.

It can be problematic that the D latch updates its state continuously when CLK is 1, since you might want to update the state only at a specific instant in time. The flip-flop is better suited for synchronous sequential logic because of this.

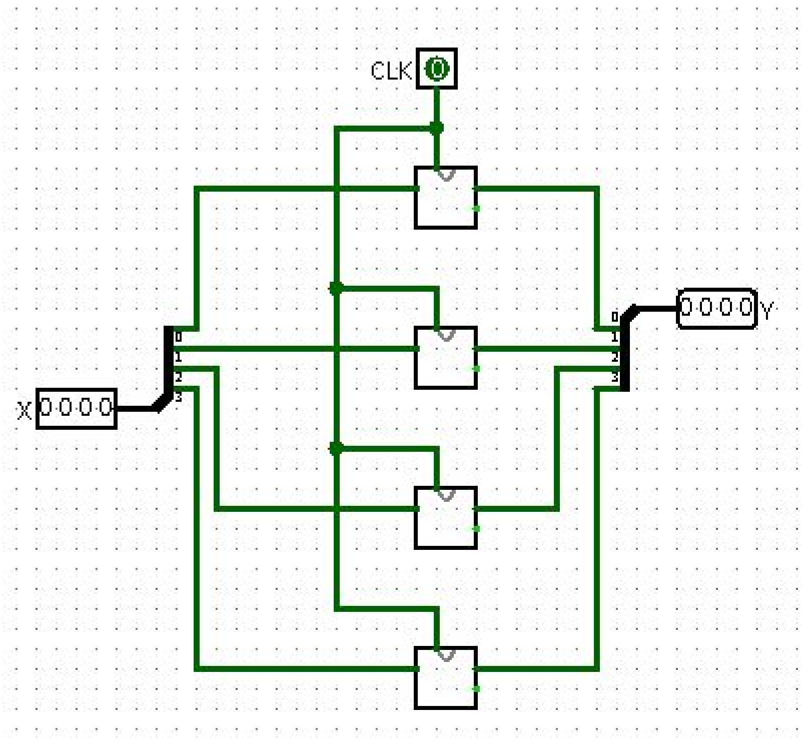
William Berg
willb@kth.se

Assignment 5: D Flip-Flops and Registers
Task 5.1.



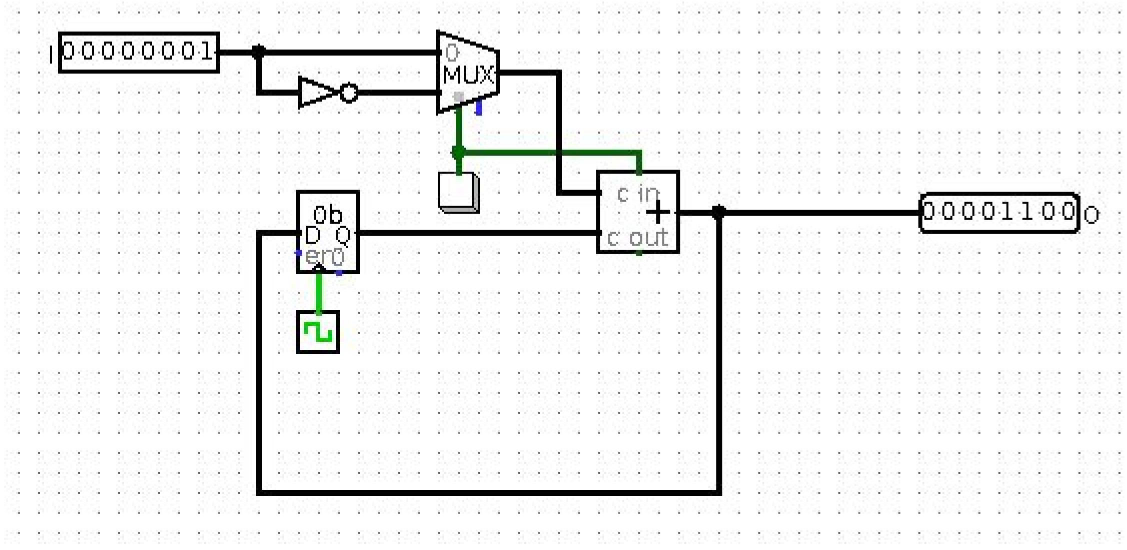
A flip-flop is edge triggered, meaning that the value of Q only changes in the instant that CLK is toggled from 0 to 1. Looking at the above screenshot for example, the value would change to 0 if we toggled the D input to 0 and then toggled the CLK input twice to 0 and then back to 1 again.

Task 5.2.



William Berg
willb@kth.se

Assignment 6: Design of a Synchronous Sequential Circuit
Task 6.1.



The multiplexer is used to determine if we want to decrement or increment (the control signal is toggled by the button). If we want to increment, the positive value stored in I passes through the multiplexer and is added with the value stored in the register. If we want to decrement, we press the button and the multiplexer will choose the second input that is the negative equivalent of the value stored in I and that will be added with the value in the register - therefore decrementing the output. The negative value is obtained by inverting and adding 1 (by passing 1 to the carry in input of the adder).

A synchronous sequential circuit is a sequential circuit that controls the circuit's memory using a clock. The state or the memory (or the output of the flip-flops) of such a circuit is only changed on clock pulses. Synchronous logic is therefore synchronized by a clock signal, meaning that the circuit's outputs are changed at discrete times in response to this clock signal (and not directly when there are changes in input). Task 5 and 6 in this lab can be classified as synchronous sequential circuits.

William Berg
willb@kth.se