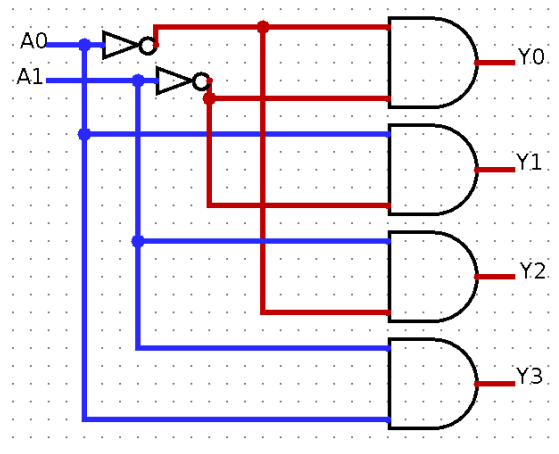


Assignment 1: Decoder

1.1.1.



1.1.2.

I confirmed my solution works by creating a table for all the different inputs.

A1	A0	Y0	Y1	Y2	Y3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

According to H & H, the decoder outputs exactly one output regardless of the number of inputs. Also, I have 2 inputs, and $2^2 = 4$ outputs which correctly represents a 2:4 decoder.

Assignment 2: Multiplexer

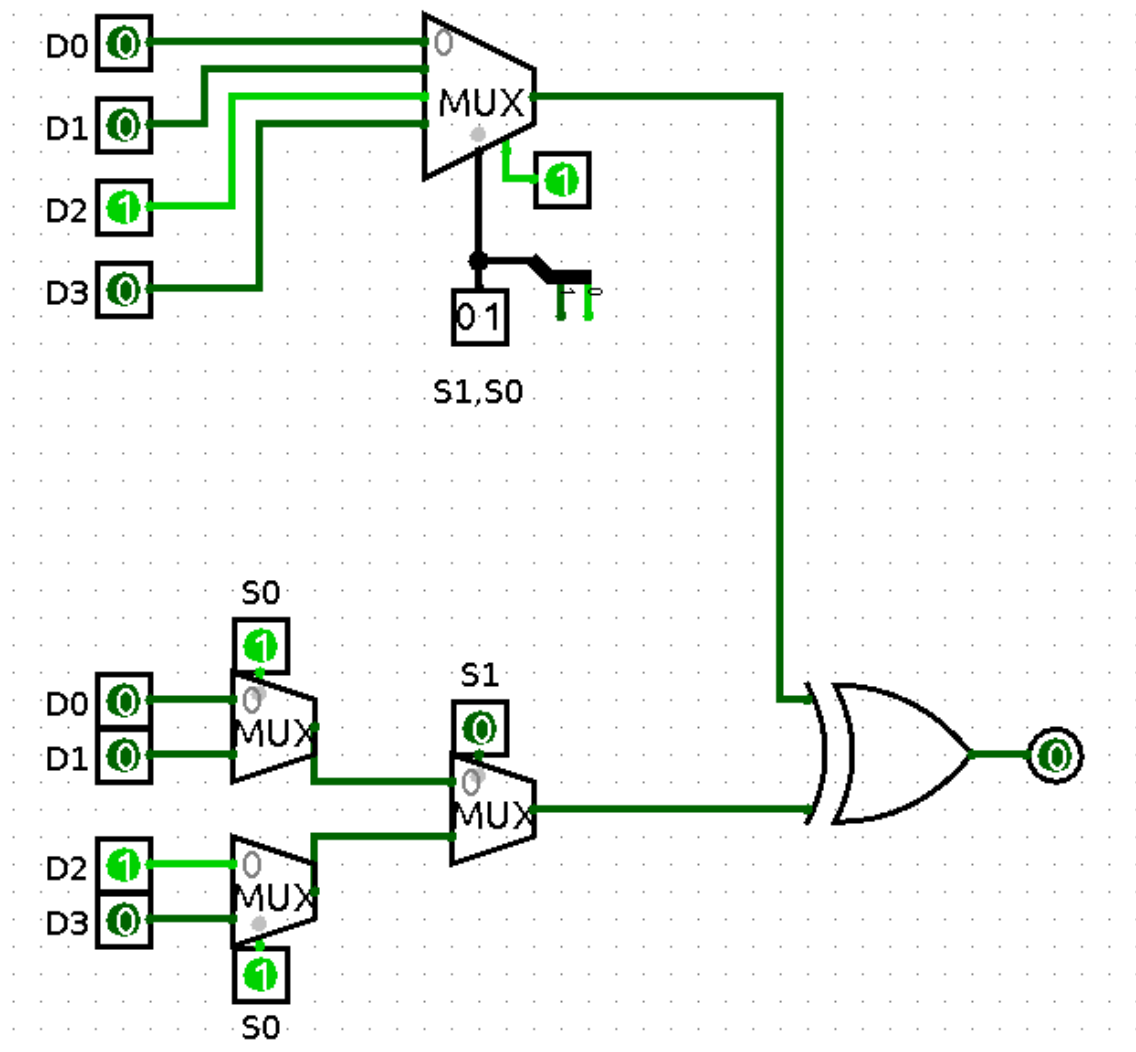
2.1.1.

The wire is blue because it is an unknown one-bit value. This depends on the enabler, if I set the enabler to 1 the wire will be dark-green indicating output = zero.

The equal signal is red because it is indicating an error, this is because of conflicting values.

The select wire is black since it is carrying multiple values, of which some may or may not be specified.

2.2.1.



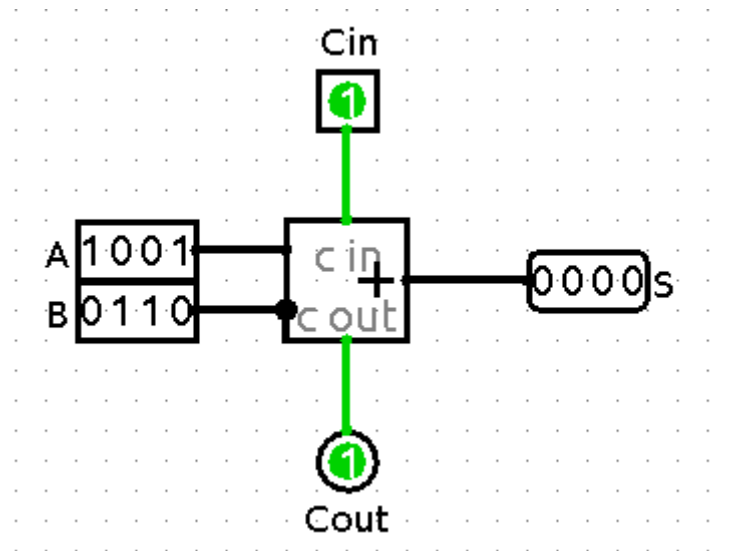
2.2.2.

In my provided screenshot, the lower S0's and S1 are swapped in place in comparison to the select-port in the upper 4:1 multiplexer. S0 and S1 (lower) represent each of the numbers on the 2-bit select port. In my picture, I set both S0's to 1 and S1 to 0 for it to be equal to the 2-bit select port 01 in the multiplexer above.

The enabler is set to 1 for me to be able to see the output of the upper multiplexer. The lower multiplexers S0 inputs have to be set to the same input for it to correctly correspond to the upper multiplexers inputs. Using the XOR-gate I can see that the inputs are both zeros and the output is an expected zero, in accordance with a XOR-gates functionality.

Assignment 3: Adder

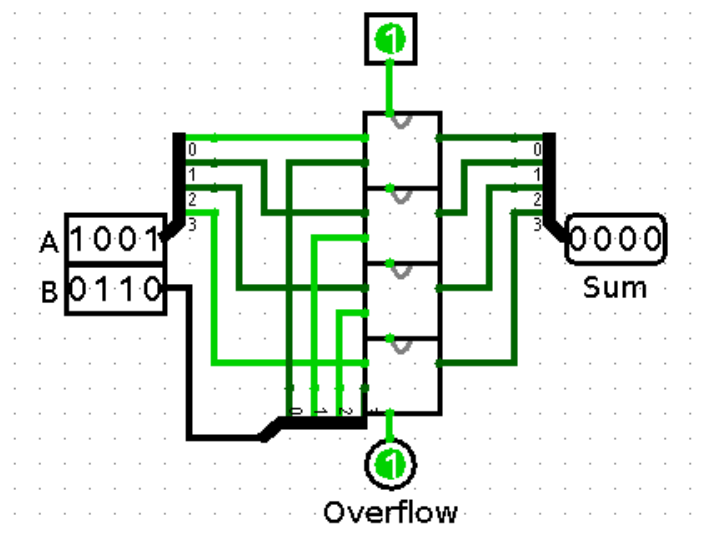
3.1.1



This configuration is pretty self-explanatory since the overflow occurs when the sum is above the allowed 16-bit limit, A and B equals 15 (16-bit), and Cin adds another bit to it, equaling 17-bits causing an overflow in Cout.

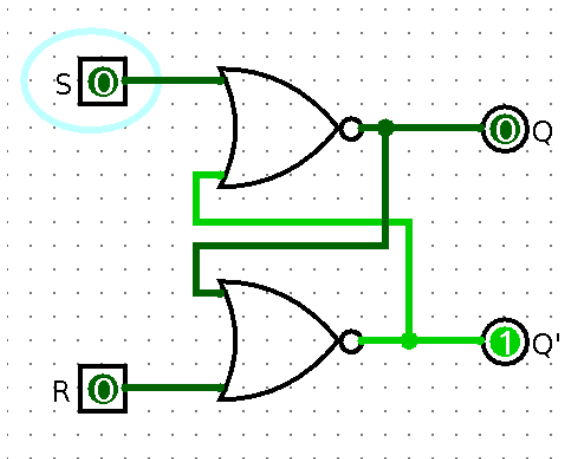
3.2.1

For me to create a 4-bit ripple carry adder, I had to change the previous adder to a one-bit by changing the inputs, outputs and adder to one-bit only. The provided screenshot uses four of these. The provided screenshot has the same values as the previous assignment and results in the same output.



Assignment 4: Latches

4.1.1

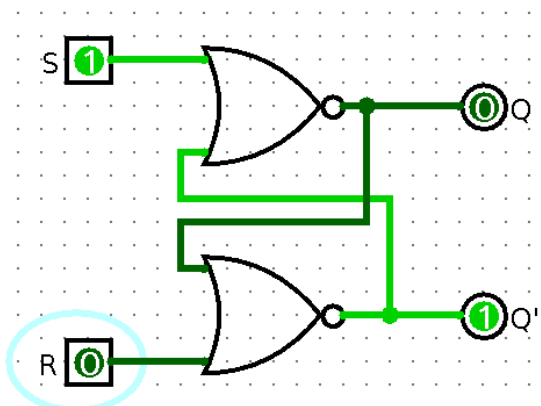


If both inputs are 1 (high), the output is reset to zero. Otherwise, depending on which input I set to one the other output is going to become true (1).

An example of this is the provided screenshot.

This picture has Q' to one and Q to zero. This is because the previous input from S was one, but was saved after I set it to zero.

4.1.2.



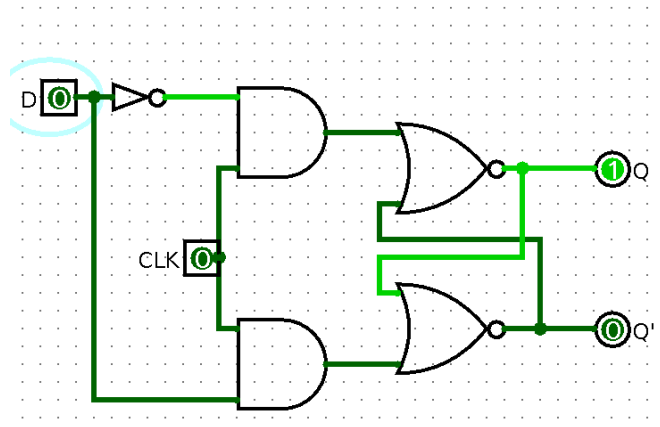
When both inputs are zero, the output is the previously saved one. When they're both one, the output is undefined, the outputs should complement each other but them being ones contradict this principle.

When the set input is high (1), the flip flop will be in the set state. When the reset input is high (1), the flip flop will be in the reset state. In our case: $S = 1$, $R = 0$, $Q = 0$, $Q' = 1$. Q' is the inverse of Q .

4.1.3.

The Q output is set to true (1) and stays that way however I toggle the R input.

4.2.1.



When D is set to zero and I toggle CLK, the upper and-gate will pass a true value and therefore set Q' to true (1). The SR-latch will be in a set state. Setting the CLK to zero and toggling the D input will not have any effect since the enabler is not true and therefore none of the and-gates can pass through a value.

4.2.2.

If the CLK is set to true, both the and-gates have one true-input. Toggling the D input will toggle which and-gate passes a true value and therefore decide if Q or Q' will be set to true.

4.2.3.

It is sometimes to ones benefit to have some sort of gatekeeper for the SR latches, in our case it's the CLK input. It adds another level of security (another condition which has to be met) for the values to pass through. This can be in the form of a physical switch or something similar which only passes the signal if the switch is on.

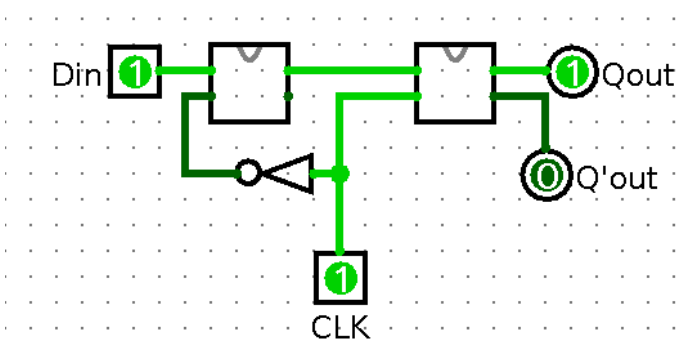
One practical use of a D latch is to eliminate mechanical "bounce". Mechanical bounce implies a state of uncertainty in whether a switch, button, keypad etc. has been fully pressed or released. The trigger can bounce and send incorrect signals. The latch can be used to solve this problem by virtue of its functionality.

4.2.4.

The problem with D latches is that you can not reliably change the state of the latch since it's not edge-triggered such as the one I'm constructing in assignment 5. This D-latch is level triggered since it doesn't trigger on an edge condition, instead it checks the level of the clock and whether to pass on the input or not. This D-latch also don't fulfill the required conditions to be a considered synchronized sequential circuit.

Assignment 5: Flip-Flops and Registers

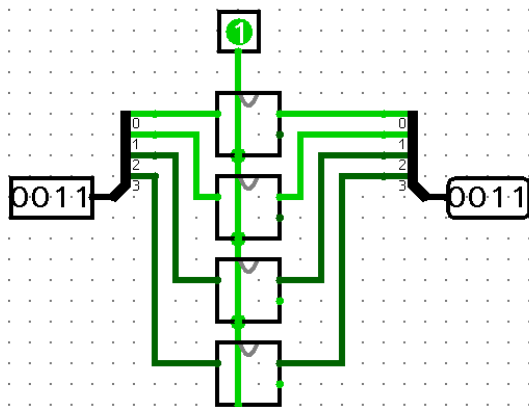
5.1.1.



5.1.2.

The value of Q changes only if the CLK value is true, since the edge is high. Changing the Din value doesn't affect the output Qout if the CLK is set to false. This corresponds to a synchronized sequential logic circuit since the output is defined based on the current clock-state and of course, the input.

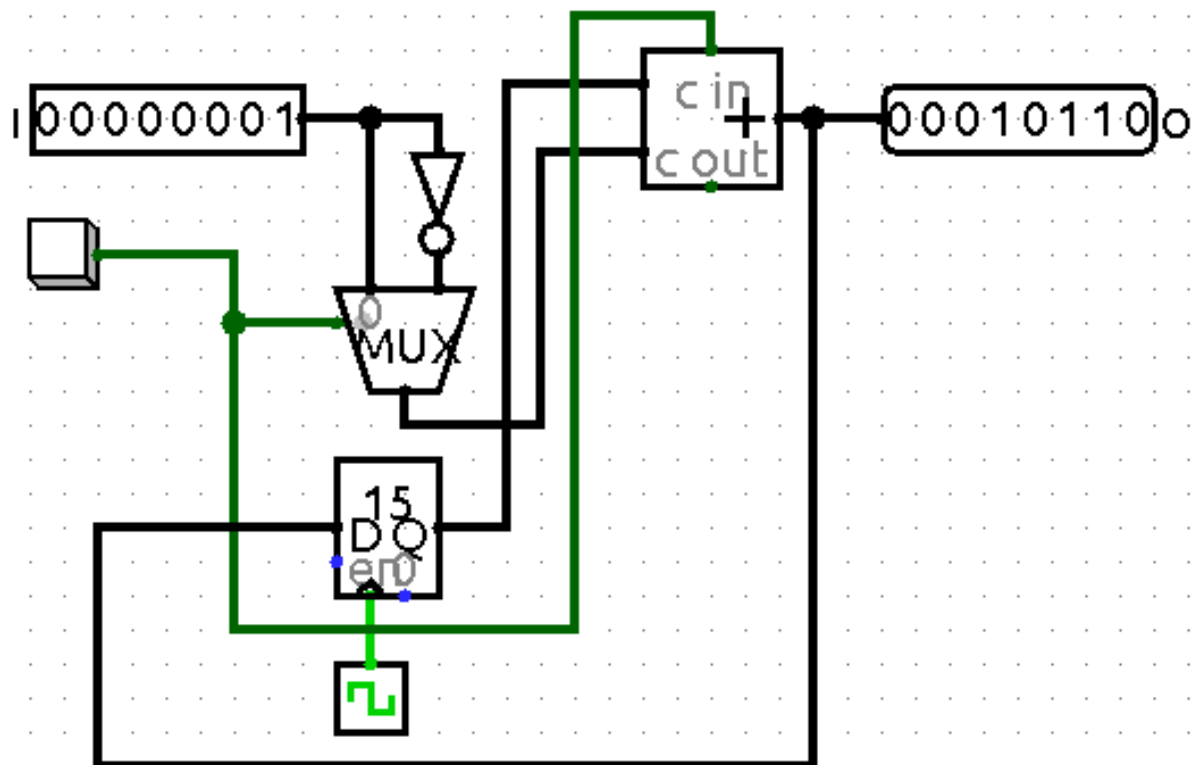
5.2.1.



My solution utilizes the D flip-flops from the previous assignment. Some small tweaks were made, like changing the position of the outputs to both be on the right side, and setting the CLK input the way it is in the screenshot to make it more visually appealing and readable. Input X on the left, output Y on the right and clock CLK signal at the top.

Assignment 6: Design of a Synchronous Sequential Circuit

6.1.1



This circuit's parts are specified in the assignment. The circuit works like this: I set the input and send the input in two parts, one inverted and one original. The 8-bit multiplexer takes these inputs and also the button's input. If the button is pressed, the multiplexer sends the inverted signal to the adder and also sets the Cin in the adder to true. The register takes the output of the adder and sets it to the input. When the CLK signal is high, it sends the register-value to the other input of the adder.

The reason I send in an inverted input and enable the Cin on the button press is because of two's complement. If I want to subtract instead, I invert it and add one, which equals subtraction in two's complement.

The circuit's flow:

Input → InvI & I → Multiplexer → Multiplexer check button press → Adder → Register & Output
Register on clock signal → Adder → Register & Output

6.1.2.

A synchronous sequential circuit consists of the following parts according to H & H:

“

- Every circuit element is either a register or a combinational circuit.
- At least one element is a register.
- All register elements receive the same clock signal.
- Every cyclic path contains at least one register.

“

The state of the circuit is synchronized to the clock state. Synchronous sequential circuits have a clock input which indicates a change in the state of the circuit. In our assignments, the ones who are a synchronous sequential circuit are this assignment and assignment 5. This is because they utilize a clock and are sequential, since all assignments past the first three are about sequential circuits.