Digital signal holds one of the 2 states: TRUE/HIGH or FALSE/LOW.
Digital circuit manipulates digital inputs and produces digital outputs.
Basic building blocks of Digital Circuits are:
+ Gates
+ And Flip-flops

AND Table

I0	I1	O
0	0	0
0	1	0
1	0	0
1	1	1

OR Table

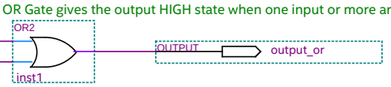
I0	I1	O
0	0	0
0	1	1
1	0	1
1	1	1

NOT Table

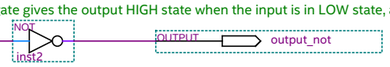
I0	O
0	1
1	0

AND Gate gives the output HIGH state when all the inputs are in HIGH state
Digital logic can be described as a "Truth Table" which shows how the output changes with the inputs

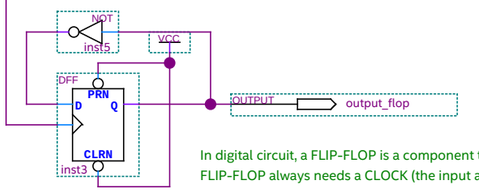
OR Gate gives the output HIGH state when one input or more are in HIGH state



NOT gate gives the output HIGH state when the input is in LOW state, and gives the output LOW state when the input is in HIGH state (invert operation)



one or more logic gates can be combined to produce more complex operations. These are called combinatorial logic.
Combinatorial logic updates its outputs immedately on the change of inputs.



In digital circuit, a FLIP-FLOP is a component that can hold its state until an update condition happens. Flip-flop belongs to storage element class of components
FLIP-FLOP always needs a CLOCK (the input at the triangular). The flip-flop in this example updates the output Q with the value of D when the CLOCK goes from LOW to HIGH. (rising-edge)

If the rising-edge of the clock doesn't happen, the Flip-flop output stays put.
In this example, the output changes from 0 --> 1 --> 0 ---> 1 ever time the CLOCK signal goes from 0 to 1.
The feedback with the NOT gate computes the NEXT output of the flip-flop to be NOT of the current output.
The next output value is pushed to Q and becomes the current output on the rising-edge of the clock.

1 In programmable devices, you have a fair bit of freedom of where you want to allocate your output pins
This type of allocation is called assignments or pin constraints
With quartus II, you can open the pin assignment by (Assignments --> Pin Planner)
In the pin planner you'll see the current assignments of CLOCK pins, and the outputs.
You'll need to make changes to suit how you plug in your LEDs (drag the signal and drop it to the pin map or type in the sheet)

2 After making changes to the pin assignments, you can run full compilation
Press Ctrl-L.
Full compilation completes with pof file in the output_files folder. This is the binary file that you'll use to configure the CPLD

3 Open the programmer (Tools --> Programmer)
Select Hardware setup to detect your USB blaster
Then click "Auto Detect". You should see your device listed.
If your device is listed, then this is where you need to tell the tool what is your programming file. Use "Change file", don't use "Add file".
After change programming file from "none" to your output_files/logic_gates.pof. Tick the program/configure box, then click start. Your device will now be configured
After programming, then you can verify the operatation of the Logic Gates with the buttons

Another interesting component is the MULTIPLEXER (or Mux). The multiplexer is built from the 3 basic components above.
You can "google" Multiplexer to see how it works and devise an experiment to understand its operation
Hint: If you look at the AND gate again, and look at one of the Inputs, you can see that it's really acting as a gate.
If it's a 0 --> The output is always 0. If it is a one, the output follow the OTHER input (Gating).
For A 2 to 1 multiplexer, when S = 0, I1 is gated, output follows I0. When S = 1, I0 is gated and output follows I1
Google Karnaugh map to see a systematic way of designing Digital Logic.