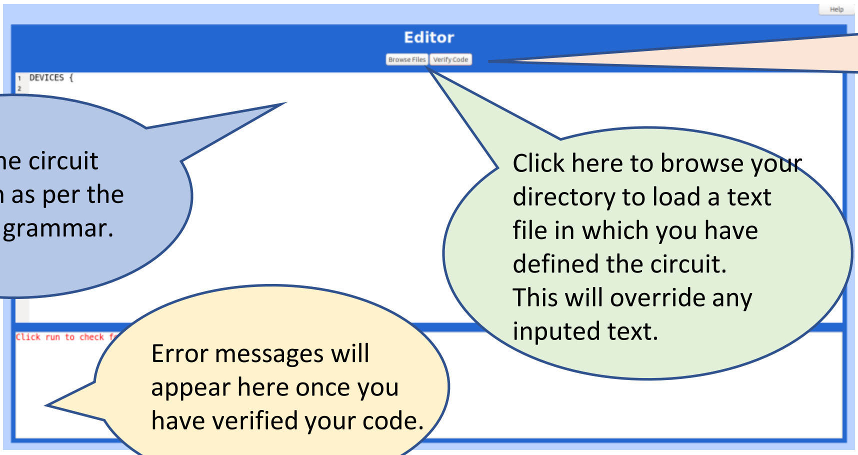
USER GUIDE

STEP 1: getting started

1. Navigate to the 'main_project' directory using the command line
2. Type 'export PATH=/usr/local/apps/anaconda3-5.0.1/bin:$PATH'
3. type 'python3 main.py' + press ENTER

STEP 2: define your circuit



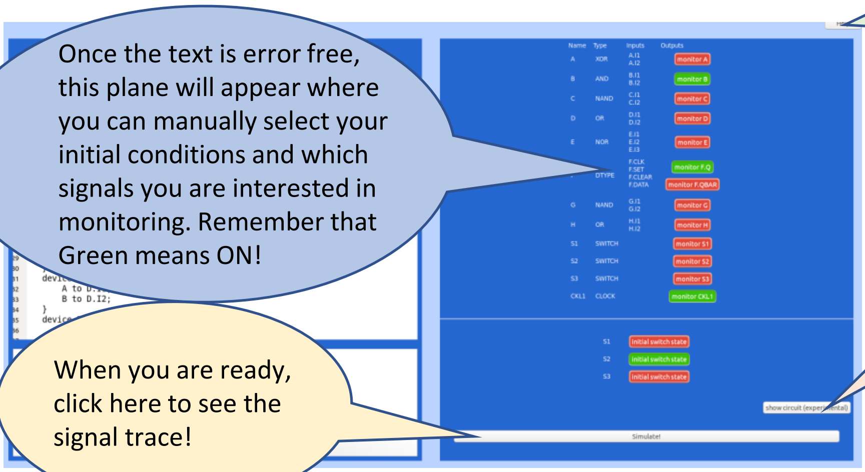
Once you are happy with your text, click here.

Type in the circuit definition as per the provided grammar.

Click here to browse your directory to load a text file in which you have defined the circuit. This will override any inputed text.

Error messages will appear here once you have verified your code.

STEP 3: define your circuit



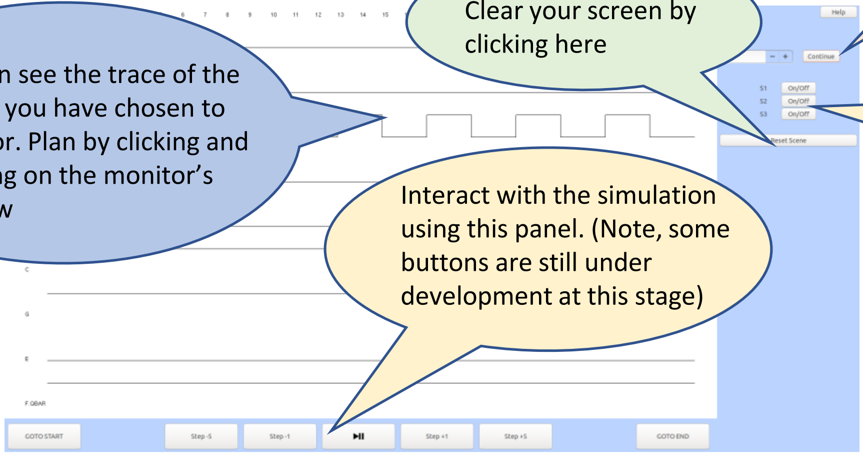
You can always click help if you are unsure what to do

Once the text is error free, this plane will appear where you can manually select your initial conditions and which signals you are interested in monitoring. Remember that Green means ON!

Click here to see the logic circuit that you are modelling (slightly experimental)

When you are ready, click here to see the signal trace!

STEP 4: run your simulation



Choose a time scale and click continue if you want to run your simulation for longer

Clear your screen by clicking here

You can see the trace of the signals you have chosen to monitor. Plan by clicking and scrolling on the monitor's window

Change the switch values here

Interact with the simulation using this panel. (Note, some buttons are still under development at this stage)

GF2 Logic Simulator Detail and Sample Files (Shared)

The circuit file is made up of three parts:

- DEVICES
- CONNECTIONS
- MONITORS

In terms of a regular circuit: DEVICES is to select how many of each device type as well as the specifications and names of each device, CONNECTIONS is the wiring up of the different devices together and MONITORS is the decision of where to place the voltmeter to find out what's going on inside the circuit.

Each part starts with the name, followed by an open curly bracket "{", and ends with a close curly bracket "}".

DEVICES

Each line must end in a colon ";". All device names must consist of an integer number of alpha-numeric characters and must start with a letter. Spaces are not allowed.

Each device must first be defined. This can be done in several ways:

- *Device name* is/is a *device type*
- *Multiple device names separated by a comma (e.g. A, B, C)* are/are some *device types*
- *Range of devices* are/are some *device types*

A range consists of a series of devices which end in numbers and have the same string preceding the final numbers. The beginning and end will be separated by the symbol "=>". As such, C1 => C3 will consist of C1, C2 and C3 but C1 => Dandelion3 will be invalid.

Each device must then have its number of inputs defined. This can be done in similar ways to the definitions:

- *Device name* has *number* inputs
- *Multiple device names separated by a comma (e.g. A, B, C)* have *number* inputs
- *Range of devices* have *number* inputs

DTYPES have pre-ordained inputs (clock, reset, clear and data) so there is no need to define those.

For clocks, there is no need to define inputs, as they have no inputs. Simply write the following:

Clock name has cycle *clock cycle time*

CONNECTIONS

Each device must have its inputs done separately. Start talking about a particular device by writing:

Device *device name* {

What will then follow will be a list of everything connecting to the inputs of this device. Input ports for DTYPES are called CLK (clock), SET, CLEAR and DATA. All other input ports will be labelled "I" followed by a number, e.g. I1, I2, I3.

Connections are made in the following way:

*Device name at start of connection** to *device name at end of connection* . *device port at end of connection* ;

e.g. "Andrew to Fantastic.I1;" will connect the output of Andrew to the first input port of Fantastic.

*For DTYPES, add either ".Q" or ".QBAR" depending on which output being connected.

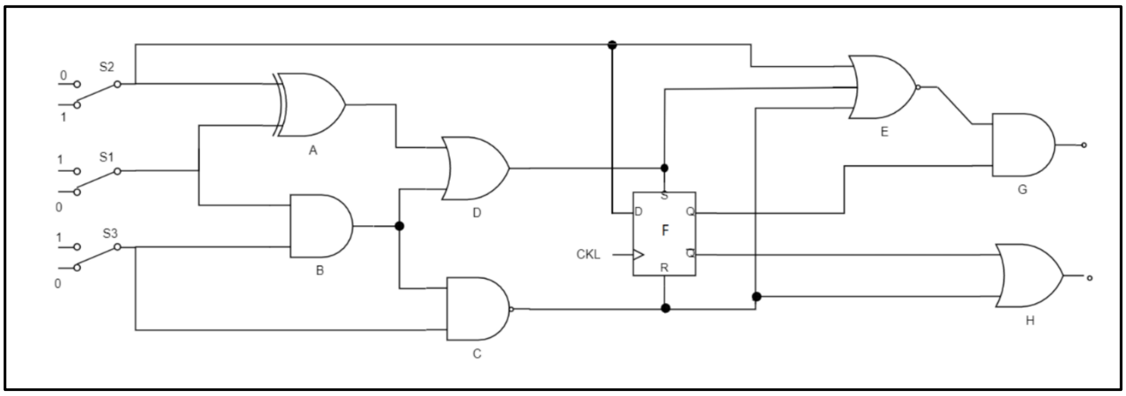
Finish talking about a specific device by writing a closing curly bracket "}".

MONITOR

Simply list the device names that should be monitored followed by a colon at the end of each line. As with connections, simply add ".Q" or ".QBAR" at the end of the device name.

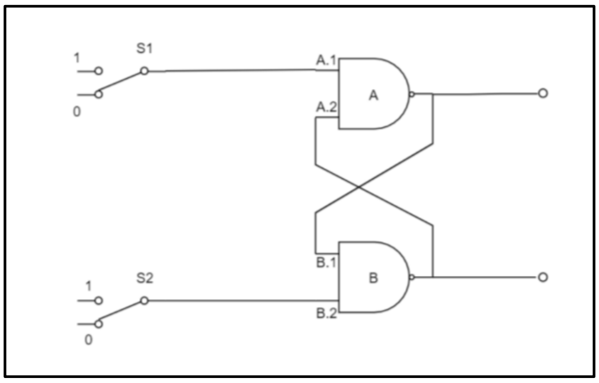
Worked Example 1 – Complex Circuit

```
DEVICES {
  A is XOR;
  B is AND;
  C is NAND;
  D is OR;
  E is NOR;
  F is DTYPE;
  G is NAND;
  H is OR;
  A,B,C,D,G,H have 2 inputs;
  E has 3 inputs;
  S1 => S3 are SWITCH;
  S2 set 1;
  CKL1 is CLOCK;
  CKL1 has cycle 5;
}

CONNECTIONS {
  device A {
     S1 to A.I1;
     S2 to A.I2;
  }
  device B {
     S1 to B.I1;
     S3 to B.I2;
  }
  device C {
     S2 to C.I1;
     B to C.I2;
  }
  device D {
     A to D.I1;
     B to D.I2;
  }
  device E {
     D to E.I1;
     S2 to E.I2;
     C to E.I3;
  }
  device F {
      CKL1 to F.CLK;
      D to F.SET;
      C to F.CLEAR;
      S2 to F.DATA;
  }
  device G {
     E to G.I1;
     F.Q to G.I2;
  }
  device H {
     C to H.I1;
     F.QBAR to H.I2;
  }
}

MONITOR {
  B, F.Q, CKL1;
}
```



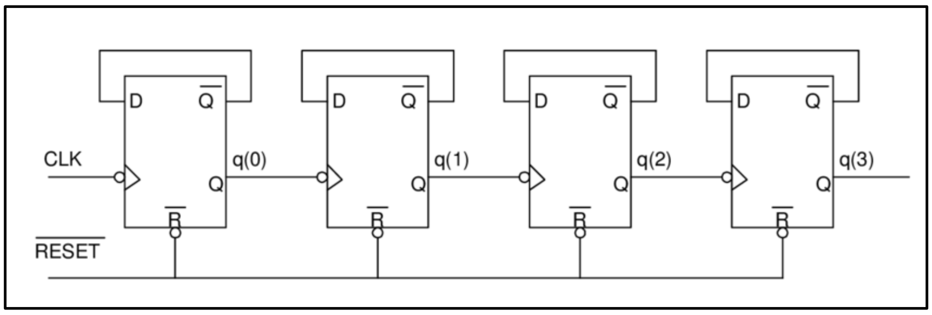
Worked Example 2 – SR Flip-Flop

```
DEVICES {
   A, B are NAND gates;
   S1, S2 are SWITCH;
   A, B have 2 inputs;
}

CONNECTIONS {
   device A {
      S1 to A.I1;
      B to A.I2;
   }
   device B {
     A to B.I1;
     S2 to B.I2;
    }n
}
MONITOR {
   A, B;
}
```



Worked Example 3 – 4 Bit (1 Nibble) Ripple Counter

```
DEVICES {
  A, B, C, D are DTYPE;
  RST, ST are SWITCH;
  CKL1 is CLOCK;
  CKL1 has cycle 1;
}

CONNECTIONS {
  device A {
     A.QBAR to A.DATA;
     RST to A.CLEAR;
     ST to A.SET;
     CKL1 to A.CLK;
  }
  device B {
     B.QBAR to B.DATA;
     RST to B.CLEAR;
     ST to B.SET;
     A.Q to B.CLK;
  }
  device C {
     C.QBAR to C.DATA;
     RST to C.CLEAR;
     ST to C.SET;
     B.Q to C.CLK;
  }
  device D {
     D.QBAR to D.DATA;
     RST to D.CLEAR;
     ST to D.SET;
     C.Q to D.CLK;   }
}

MONITOR {
  A.Q, B.Q, C.Q, D.Q, CKL1;
}
```



CLK
$\overline{\text{RESET}}$
D
$\overline{Q}$
Q
$\overline{R}$
q(0)
q(1)
q(2)
q(3)