To run verilog in simulation without the Vivado software, you can use these two software:

1. Icarus Verilog (verilog simulator; runs in the terminal)
2. vaporview (waveform viewer; VSCode extension)

Installation

macOS:

If you have homebrew, then you can install Icarus verilog with `brew install icarus-verilog`. Otherwise, you may need to build it manually. See https://steveicarus.github.io/iverilog/usage/installation.html for more information.

Linux:

Check whether your package manager has Icarus verilog (the package will probably be called icarus-verilog or iverilog). If not, follow the link above for installation instructions.

Windows:

Prebuilt binaries can be found at: http://bleyer.org/icarus/

To install vaporview, simply install it through VSCode like any other extension.

Running a simulation

1. Suppose you have a counter module and a testbench, named `counter.v` and `counter_tb.v`.
2. In your testbench module, insert the following:

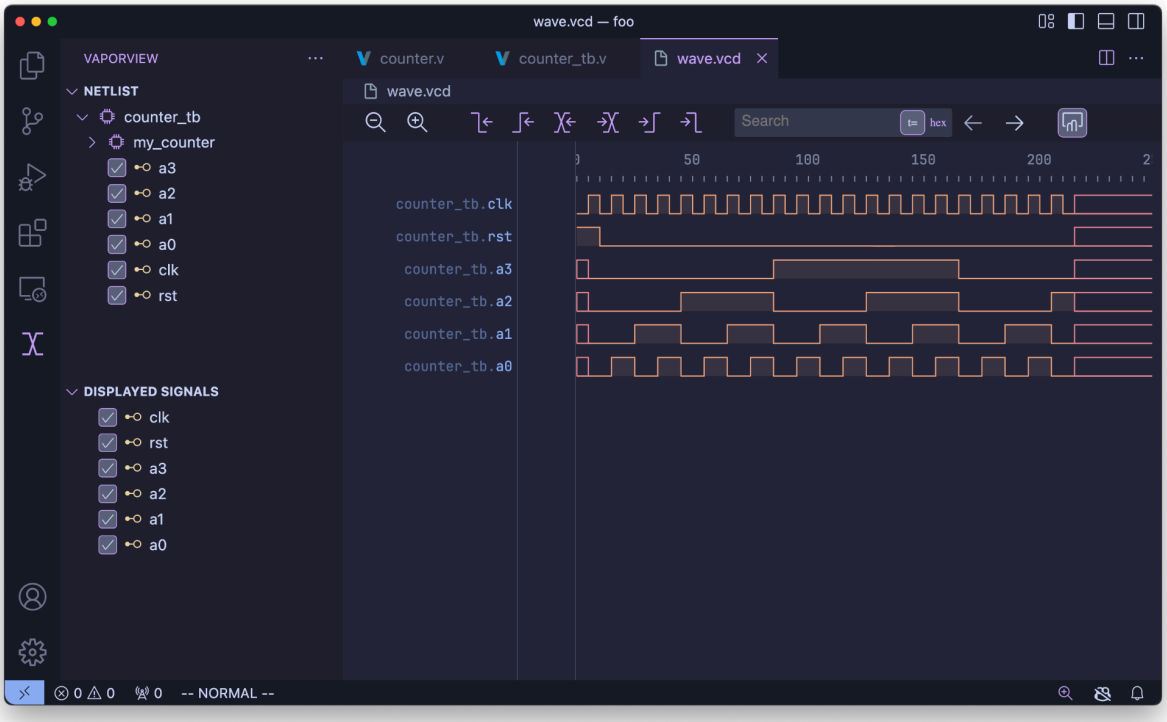
```
initial begin
    $dumpfile("wave.vcd");
    $dumpvars(0, counter_tb);
end
```

3. Open up your terminal. To compile your verilog code, run `iverilog counter.v counter_tb.v`. This will produce a file called `a.out`.

4. To run the simulation, run `./a.out`. This will produce a file called `wave.vcd` that contains your waveform data.

```
biqua@biquando-air ~/pg/verilog
 ❱ ls
counter.v     counter_tb.v
biqua@biquando-air ~/pg/verilog
 ❱ iverilog counter.v counter_tb.v
biqua@biquando-air ~/pg/verilog
 ❱ ./a.out
VCD info: dumpfile wave.vcd opened for output.
counter_tb.v:19: $finish called at 210 (1s)
biqua@biquando-air ~/pg/verilog
 ❱ ls
a.out*        counter.v     counter_tb.v  wave.vcd
```

5. In VSCode, double click on `wave.vcd`. This should bring up an empty view. To add signals to the waveform view, click on the Vaporview tab on the left sidebar. In this view, you can enable signals from each of your modules to appear in the waveform.



Tips:

To make your simulation match the Basys3 boards, we need to generate a 100MHz clock. To set the timescale of your simulation, put the following at the top of your testbench file:

```
`timescale 10ns/1ns
```

This means that each simulation time unit is 10ns (the cycle of a 100MHz clock), and you can delay for as short as 1ns (0.1 of a time unit). To generate the clock cycle in your testbench, do this:

```
initial begin
    clk = 0;
end
always @(*) begin
    #0.5; clk <= ~clk;
end
```