BBM233 Logic Design Lab
Fall 2023

Guide to Verilog Lab Experiments
Getting Started With Icarus Verilog and GTKWave
(Unofficial software but good enough for this course)
Installation Steps For Windows

November 17, 2023

Icarus Verilog

Icarus Verilog is a free and open-source Verilog compiler and simulator which can be used to develop HDL code. You don't need any accounts or large storage space available to download this tool. You can learn more about this software, read installation and usage instructions both for Windows and for other platforms from their official website. However, if you are looking for some simple instructions for Windows, you can just keep following our guide.

Icarus Verilog can come in handy if you are planning to use a 🥔 PC or if you have a thing for open-source software. For instance, a very old computer, rocking some lightweight Linux distro (e.g Puppy Linux [1]), can be used to develop HDL code, thanks to this amazing tool.

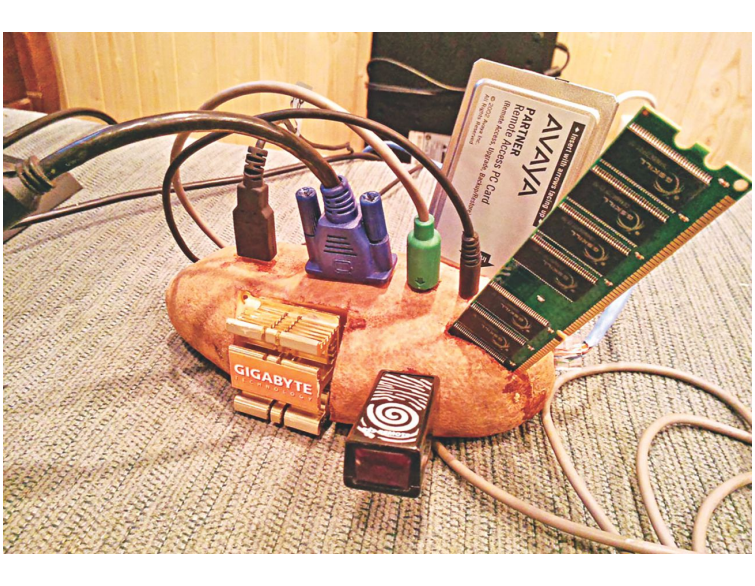


Figure 1: A 🥔 PC designed for Cutting-Edge FPGA simulation purposes

1 Downloading Icarus Verilog

The first thing you should do to install Icarus Verilog on your computer is to go to this website and download an installer that is suitable for your PC. We advise you to download the latest installer, which is marked in 2. Even with a slow internet connection, downloading the installer should not take long. (6 minutes 15 seconds for 44.8 MB for 1 Mbps connection [2])

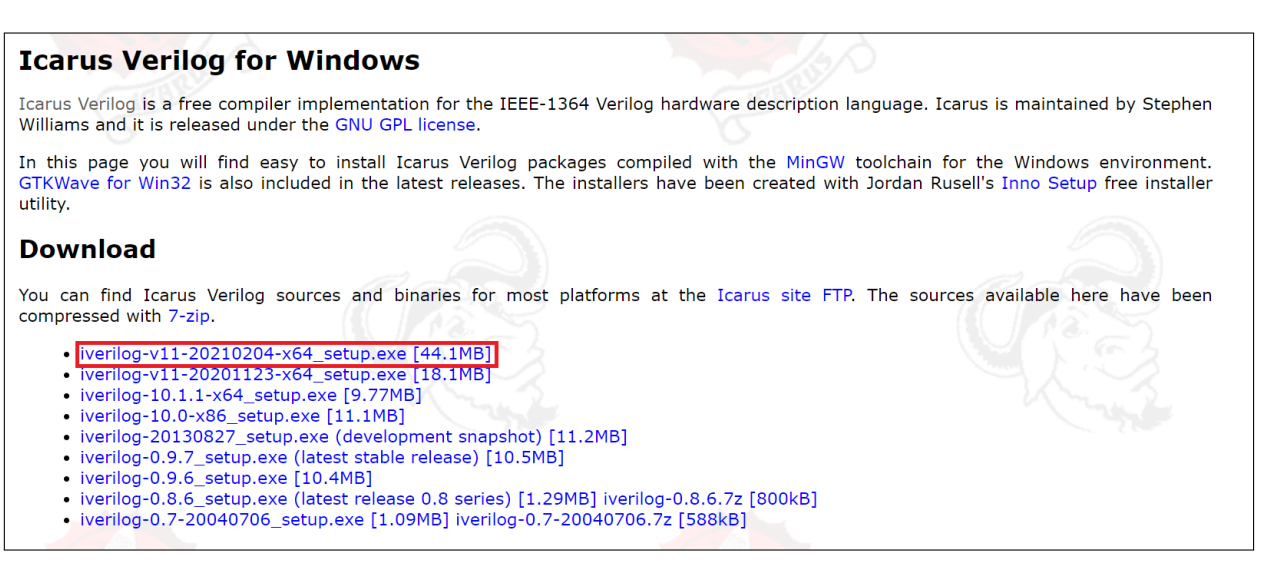
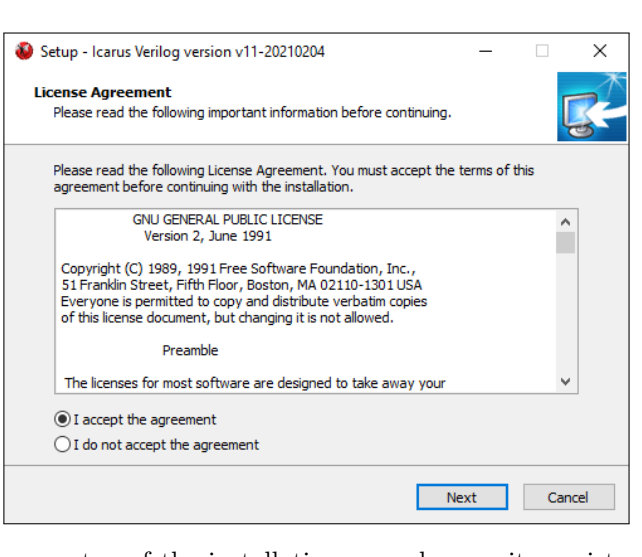


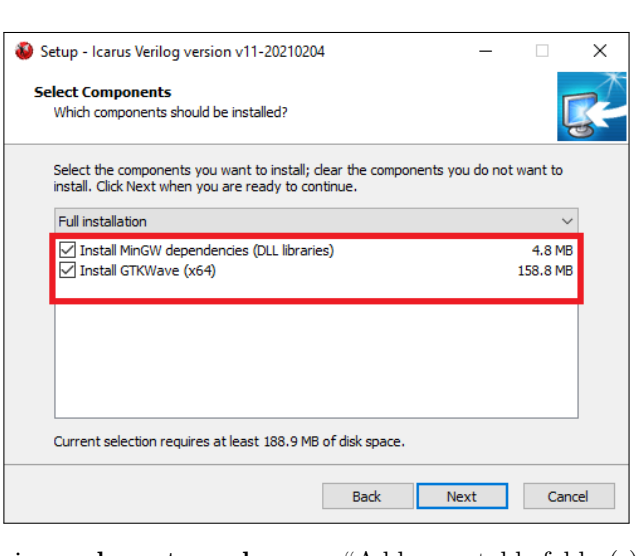
Figure 2: Downloading the installer

2 Installation Steps

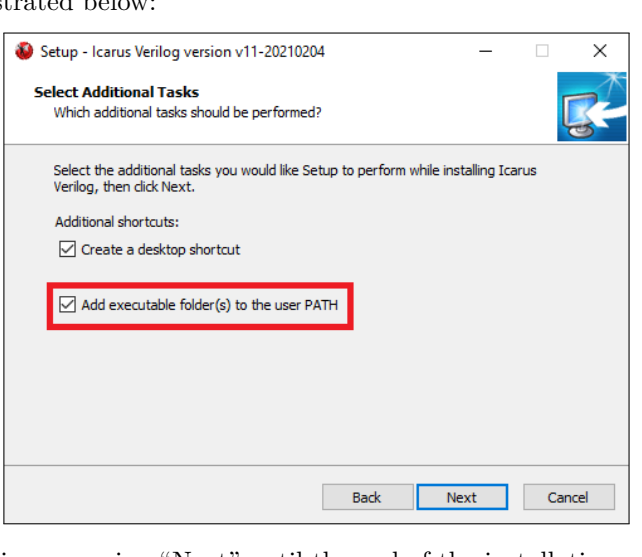
Even though installing software on Windows is pretty straightforward, some steps that can potentially make you suffer if you don't pay attention to them; but don't worry, we explain those steps below.



We will not demonstrate every step of the installation procedure as it consists of mostly pressing "Next". But some steps are important. The first one is **you have to make sure** both of the options are selected in the step illustrated below:



The second important step is **you have to make sure** "Add executable folder(s) to the user PATH" option is selected in the step illustrated below:

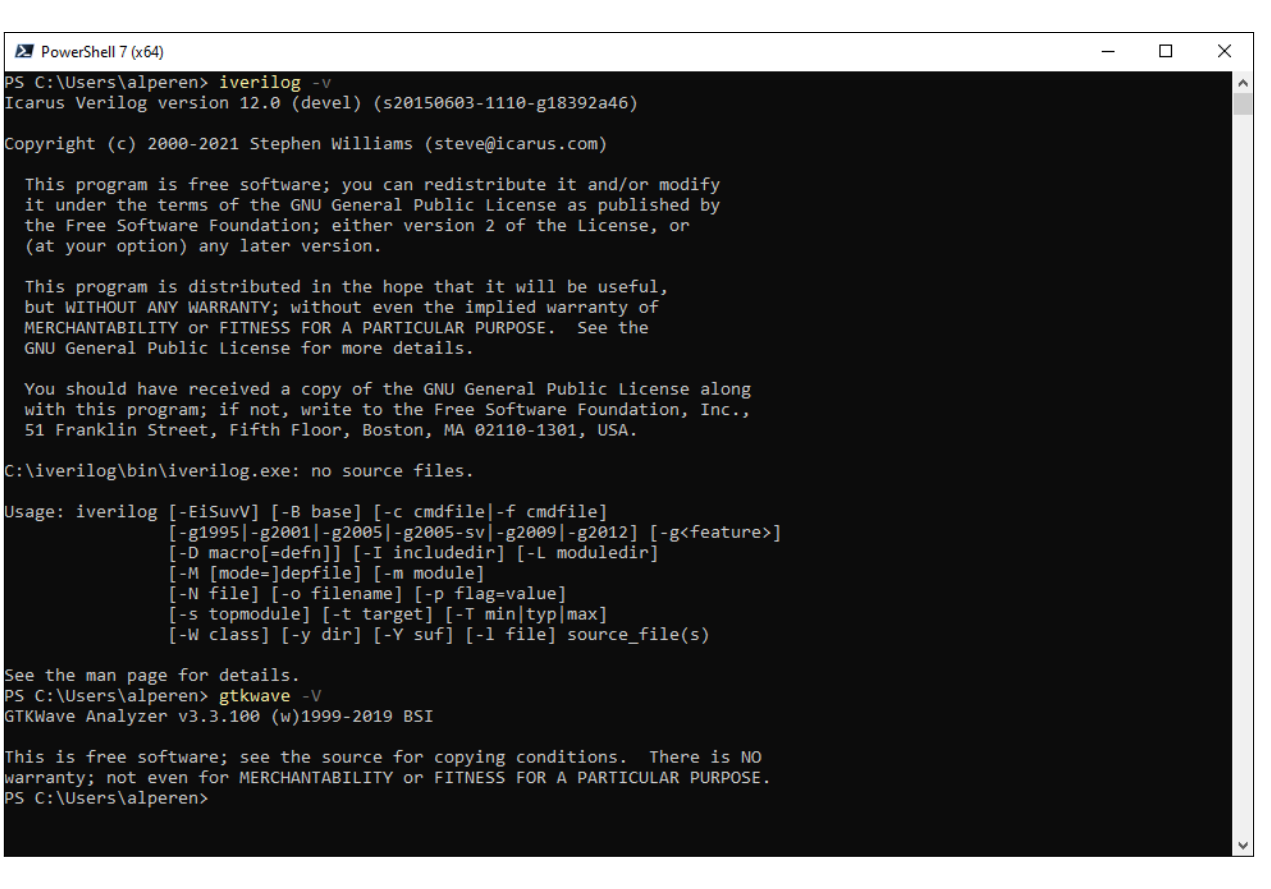


You can trust us and continue pressing "Next" until the end of the installation.

After the installation, all the necessary folders which contain Icarus Verilog binaries should be added to the user PATH. You should test if everything is alright by typing the following commands on your preferred Windows terminal application:

```
iverilog -v
gtkwave -V
```

The result of those commands should be similar to the below figure:



After seeing the proper output of those commands, you are pretty much ready to go. You can just start compiling and simulating Verilog code.

3 Compiling and Running Your First Verilog Module

3.1 Example 1

The first example will consist of a single module, which is going to print "Hello world!" to the console. This is not a very good use case in Verilog as it is used to describe hardware, however, this example is quite good for demonstrating the basic usage of Icarus Verilog.

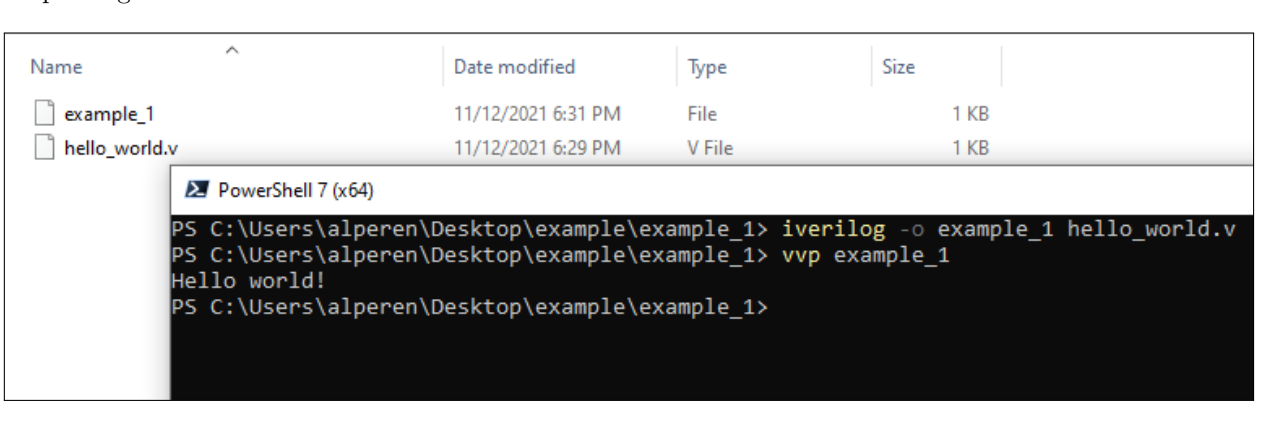
In this example, we are going to compile and simulate the following source code which is saved in a text file named **"hello_world.v"**:

```
module hello_world;
    initial begin
        $display("Hello world!");
    end
endmodule;
```

To compile and simulate this piece of code, we are going to open a new terminal in the same folder as the source file, and enter the following commands:

```
iverilog -o example_1 hello_world.v
vvp example_1
```

The first ***iverilog*** command is used to compile the given source file into an output file named ***example_1***, which can be simulated using the second ***vvp*** command. As you can see from the following figure, the correct output is generated as the result of the simulation:



3.2 Example 2

The second example consists of two source files; a Verilog module and a testbench to test the first one. This is a much more realistic use case in Verilog, and for our experiments, you will have to use at least two source files.

The first source code is as follows, saved in a text file named **"and_module.v"**:

```
module and_module(input wire a, b, output wire f);
        and(f, a, b);
endmodule
```

This is a simple module that takes two inputs ***a*** and ***b***, ANDs them and writes the result to the wire ***f***. To test this simple module, we are going to use the following truth table of the AND gate:

a	b	f
0	0	0
0	1	0
1	0	0
1	1	1

To test all the values in the above truth table, the following Verilog testbench is used, saved in a text file named **"and_module_tb.v"**:

```
`timescale 1 ns/10 ps

module and_module_tb;
        localparam period = 20;
        reg a,b;
        reg[2:0] count = 2'b00;
        wire f;
        integer i;

        and_module uut(.a(a), .b(b), .f(f));

        initial begin
                $dumpfile("result.vcd");
                $dumpvars;
                for (i = 0; i < 4; i++) begin
                        {a,b} = count;
                        count += 1;
                        #period;
                end
                $finish;
        end

endmodule
```

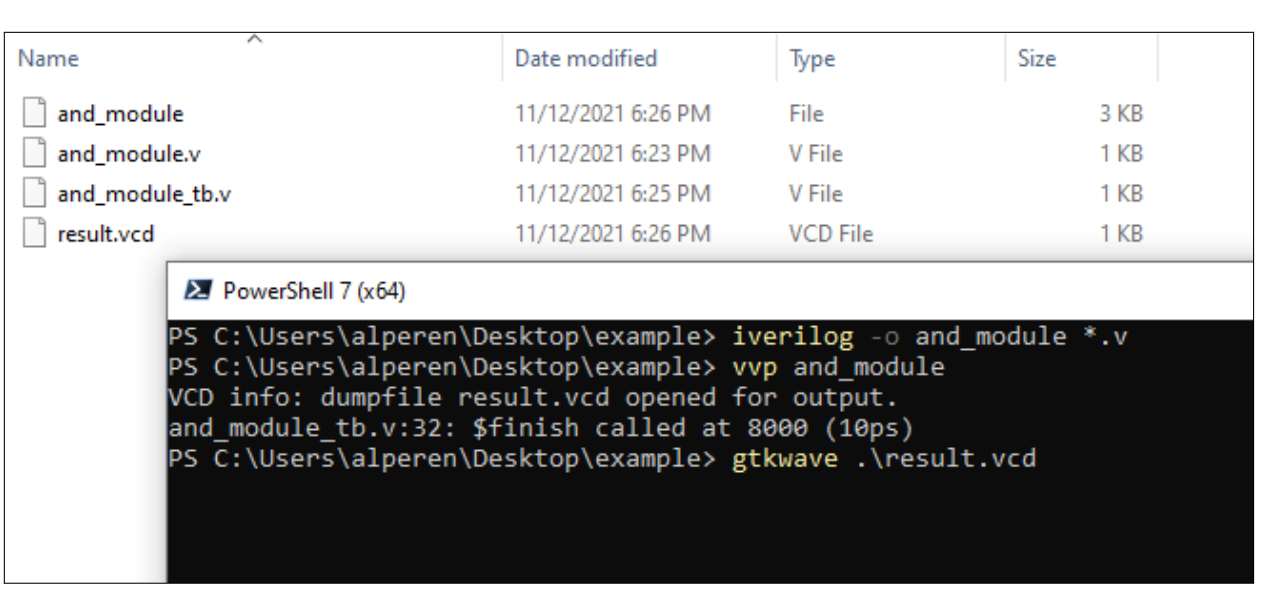
To compile and those two source files into one output file then simulate it, the following set of commands are used:

```
iverilog -o and_module *.v
vvp and_module
```

The difference from the first example is, in the first command which instructs the binary ***iverilog*** to compile source files, **"*.v"** wildcard is used, which finds every Verilog source file in the folder. This is a shortcut instead of typing every source file's name one by one as:

```
iverilog -o and_module and_module.v and_module_tb.v
```

The following figure demonstrates the output of the mentioned commands:



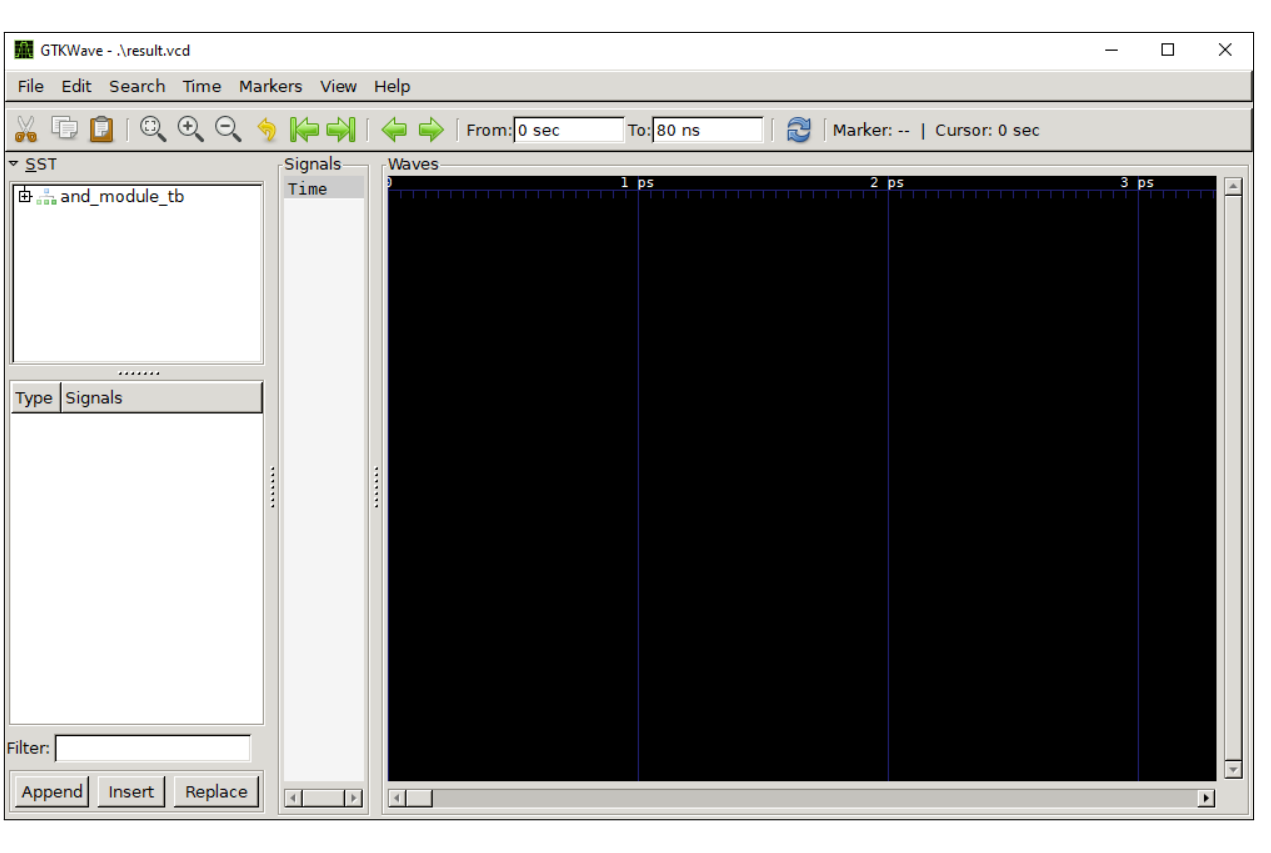
After compilation and simulation are complete, you can see that a new file named "result.vcd" is created. This situation is reported to the user by the following message:

```
VCD info: dumpfile result.vcd opened for output.
and_module_tb.v:32: $finish called at 8000 (10ps)
```

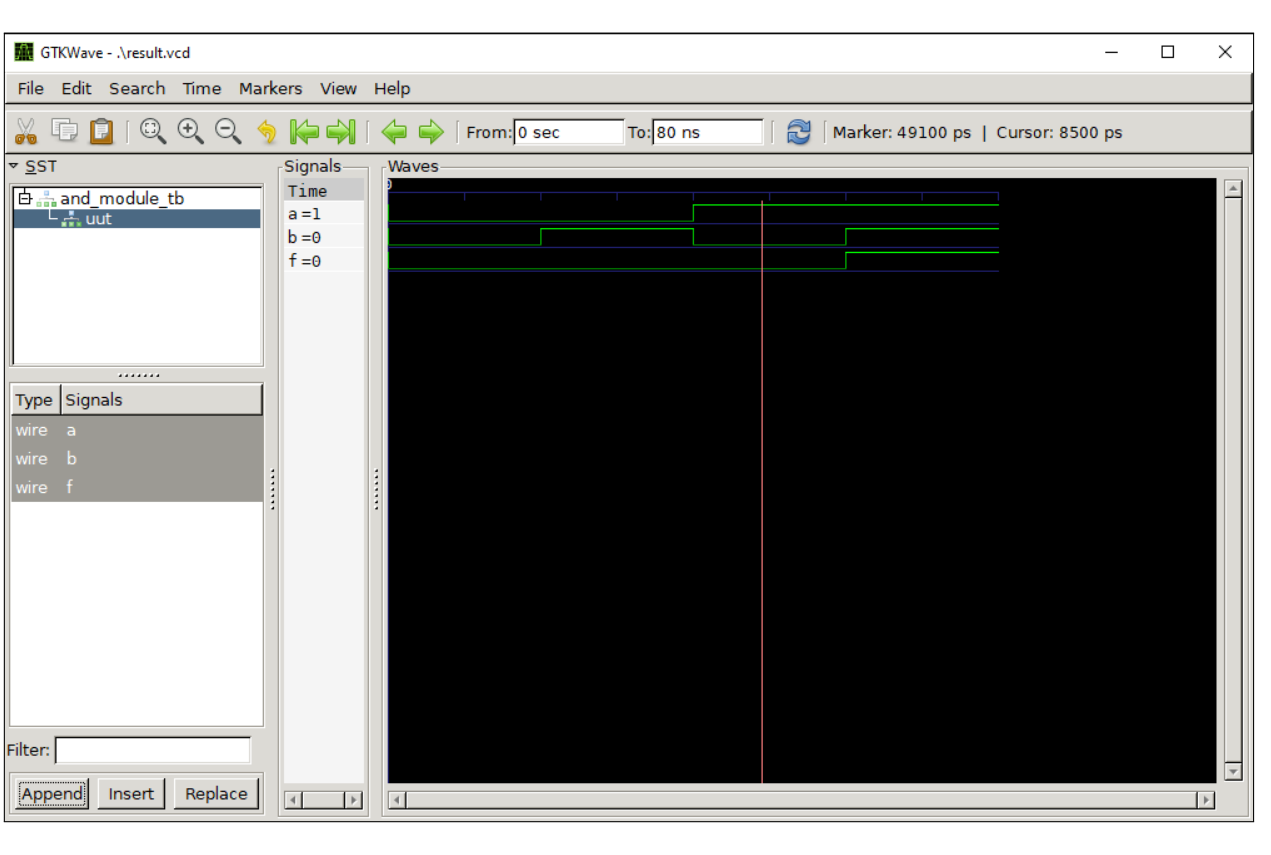
This is a waveform file which includes the result of the simulation. To view this file, the following command is used:

```
gtkwave result.vcd
```

When entered, a new GTKWave window will open which allows you to investigate the result of the simulation:



Using the left panel, you can select the signals that you need to see and add them to the right panel using the **"Append"** button. After adding the signals, you can zoom in or scroll through the waveform illustrated below:



4 Some Tips & Tricks

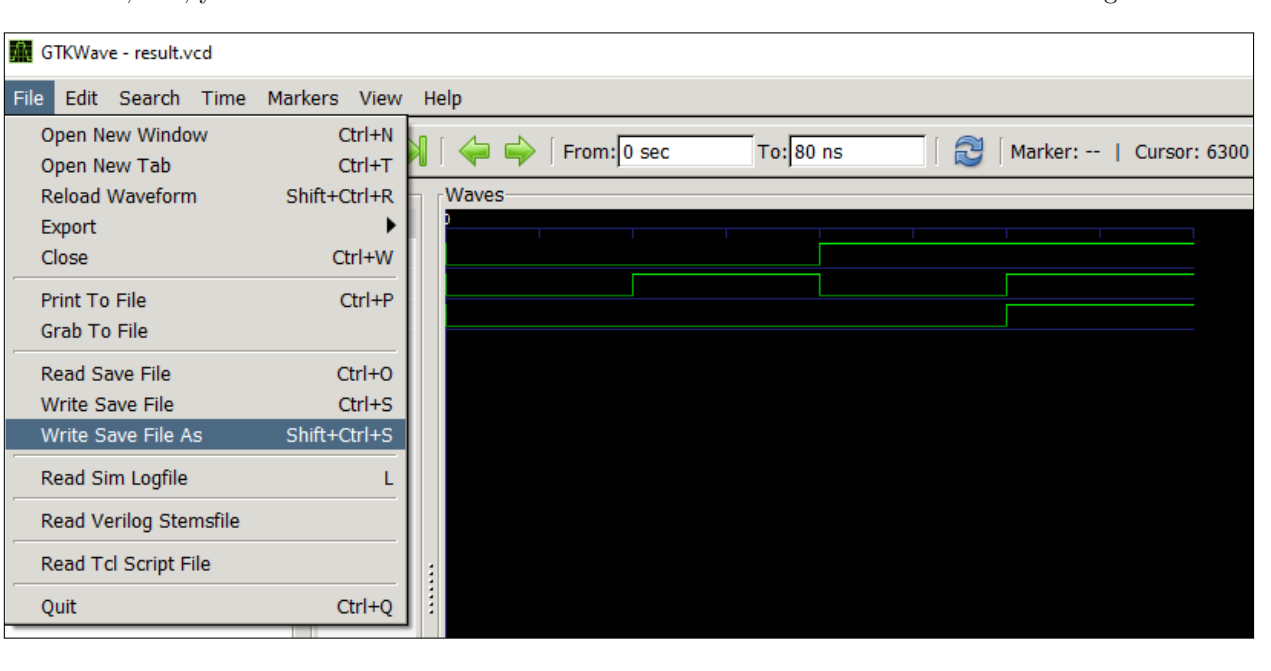
4.1 Additional iverilog Parameters

You may prefer to include the *-Wall* parameter while compiling your Verilog code to enable more warnings during the compilation. This parameter may come in handy while trying to figure out what's wrong with your code. The below figure summarizes an example usage and output of the *-Wall* paramater:

```
iverilog  -Wall  -o and_module *.v
warning: Some modules have no timescale. This may cause
       : confusing timing results.      Affected modules are:
       :   -- module and_module declared here: and_module.v:1
```

4.2 Saving and Re-using GTKWave Configuration

You may prefer to use a configuration file for GTKwave for ease-of-use. This approach allows you to get the same view after fixing your code and creating another waveform. After choosing which signals to use, the zoom level, etc., you should select "Write Save File As" from the "File" menu to create a configuration file.



Once you have created a configuration file, you may load the file by simply giving it as a second argument to GTKWave:

```
gtkwave result.vcd conf.gtkw
```

4.3 Development Pipeline using a Makefile

You may consider coming up with a Makefile to reduce the process of compilation, simulation and waveform viewing into a single command. The following Makefile contains useful commands regarding the process:

```
CC = iverilog
FLAGS = -Wall
library_input: *.v
        $(CC) $(FLAGS) -o and_module *.v
        vvp and_module
        gtkwave result.vcd conf.gtkw
```

You can download and install *make* for Windows using the following link:
https://gnuwin32.sourceforge.net/packages/make.htm

You are also recommended to add the make binary to the path for ease-of-use.

5 Useful Resources

You can visit the official website of Icarus Verilog, using the following link (as stated in the first section):
https://steveicarus.github.io/iverilog

You can learn about the detailed usage of ***iverilog*** command, using the following link:
https://steveicarus.github.io/iverilog/usage/command_line_flags.html

You can learn about the detailed usage of ***vvp*** command, using the following link:
https://steveicarus.github.io/iverilog/usage/vvp_flags.html

You can get a better idea about the detailed usage of Icarus Verilog, using the following link:
https://steveicarus.github.io/iverilog/usage/getting_started.html

You can learn about installing Icarus Verilog on different systems, using the following links:
https://iverilog.fandom.com/wiki/Installation_Guide
https://steveicarus.github.io/iverilog/usage/installation.html

You can also check out this tutorial which helped us through section 4:
https://usermanual.wiki/Document/IcarusVerilogGTKWaveguide.238990987/view

You can download the sample files used in this tutorial using the following link:
https://web.cs.hacettepe.edu.tr/~alperencakin/tutorial.zip

References

[1] https://puppylinux.com/
[2] https://downloadtimecalculator.com/