***Computer Architecture Course***

**Getting Started Guide**

**Issue 1.0**

**Contents**

[**1**](#_heading=h.30j0zll) **Introduction 1**

[1.1](#_heading=h.1fob9te) Lab overview 1

[**2**](#_heading=h.3znysh7) **Requirements 1**

[2.1](#_heading=h.2et92p0) Software requirements 1

[2.2](#_heading=h.3dy6vkm) Hardware requirements 2

[**3**](#_heading=h.1t3h5sf) **Lab setup 3**

[3.1](#_heading=h.4d34og8) Setting up Icarus Verilog 3

[3.1.1](#_heading=h.17dp8vu) Verifying successful Icarus Verilog installation 4

[3.2](#_heading=h.3rdcrjn) Setting up GNU Toolchain for the A-profile Architecture 6

[3.2.1](#_heading=h.lnxbz9) Verifying successful toolchain installation 6

[3.3](#_heading=h.35nkun2) Setting up GTKWave 7

[3.4](#_heading=h.44sinio) Guidelines for creating a workspace for all lab work 8

[**4**](#_heading=h.2jxsxqh) **Compiling and simulating a simple Verilog code 9**

[4.1](#_heading=h.1y810tw) Dumping out waveforms 11

[**5**](#_heading=h.2xcytpi) **Troubleshooting 13**

[**6**](#_heading=h.1ci93xb) **Additional references 13**

# Introduction

|  |
| --- |
| Lab overview At the end of this Getting Started lab, you will be able to:   * Demonstrate how to set up the software environment with appropriate commands and tools including Icarus Verilog, GNU Toolchain for the A-profile Architecture (contains GCC compiler), and GTKWave. * Use an example Verilog file to verify successful installation and setup. * Demonstrate how to compile and simulate Verilog code using Icarus Verilog. * Demonstrate how to dump simulated variables from Icarus Verilog and view them using GTKWave. |

# Requirements

## Software requirements

The following table is a list of required software tools that you need for the labs.

**Note**:

* The software versions listed in the table are versions that have been verified working with the labs. **Therefore, we recommend that you use the versions listed below.** You can use the latest available (and most stable) versions of the software if backward and forward compatibility are supported by the versions. You can also find the exact versions ready for download in the ELC Getting Started lab and/or in the [Class-Toolchain repository on the class GitHub](https://github.com/Herring-UGACSEE-4290/Class-Toolchain). You should download the entire repository as a .zip file to save from having to download the files individually
* The instructions in the lab manuals are based on a Windows OS. However, most of the software tools in this lab have a Linux version too. Therefore, you may have to modify the lab instructions and/or environment settings on your own if you are running this lab on a Linux OS.

|  |  |  |  |
| --- | --- | --- | --- |
| **Software** | **Usage** | **Website** | **Version** |
| Operating System | All labs | – | Windows 10 |
| Icarus Verilog simulator | All labs | <https://github.com/Herring-UGACSEE-4290/Class-Toolchain/blob/main/Windows/iverilog-0.9.7_setup.exe> | iverilog-0.9.7 |
| Verilog tutorial online | All labs | https://www.cis.upenn.edu/~milom/cis371-Spring11/lab/textbook-verilog-tutorial/ | It is an old site, but may be helpful. |
| Text editor | All labs | https://notepad-plus-plus.org/ | Most recent |
| tar.gz extraction | All labs | <https://www.7-zip.org/> | Most recent |
| GNU Toolchain for the A-profile Architecture | All labs | <https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-a/downloads> | version 9.2-2019.12  **Windows AArch64 ELF bare-metal target (aarch-none-elf)** |
| GTKWave waveform viewer | All labs | GTKWave is provided as part of the Icarus Verilog Simulator package listed in this table | GTKWave Analyzer v3.3.48 |
| *Optional: i) Windows Subsytem for Linux + Ubuntu terminal*  *OR*  *ii) MSYS2* | Lab 4 onward (for automating assembly and compilation only) | * + 1. <https://docs.microsoft.com/en-us/windows/wsl/install-win10>   OR  ii) <https://github.com/Herring-UGACSEE-4290/Class-Toolchain/blob/main/Windows/msys2-x86_64-20210604.exe>  Lab 4 contains instructions about these software setups. | msys2-x86\_64-yyyymmdd.exe  (year, month, day of release) |
| *Optional: Vivado HDL WebPACK for RTL synthesis* | Lab 6 (synthesis only) | <https://github.com/Herring-UGACSEE-4290/Class-Toolchain/blob/main/Windows/Xilinx_Vivado_SDK_Web_2018.3_1207_2324_Win64.exe> |  |

*Table 1: Software tools and versions for Windows 10*

|  |  |  |  |
| --- | --- | --- | --- |
| **Software** | **Usage** | **Website** | **Version** |
| Operating System | All labs | – | Linux |
| Icarus Verilog simulator | All labs | **Ubuntu Linux**: <https://iverilog.fandom.com/wiki/Installation_Guide#Ubuntu_Linux> | iverilog-0.9.7  make sure to install version **0.9.7** |
| GNU Toolchain for the A-profile Architecture | All labs | <https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-a/downloads> | **Linux:** version 9.2-2019.12 X86\_64 Linux AArch64 ELF bare-metal target (aarch-none-elf) |
| GTKWave waveform viewer | All labs | after following the instructions listed for iverilog, use the command: **sudo apt-get install gtkwave** | GTKWave Analyzer v3.3.48 |
| *Optional: Vivado HDL WebPACK for RTL synthesis* | Lab 6 (synthesis only) | <https://github.com/Herring-UGACSEE-4290/Class-Toolchain/blob/main/Linux/Xilinx_Vivado_SDK_Web_2018.3_1207_2324_Lin64.bin> | Only available for **Ubuntu 18.04 LTS** and **Windows 10**  **Linux:** <https://www.xilinx.com/member/forms/download/xef-vivado.html?filename=Xilinx_Vivado_SDK_Web_2018.3_1207_2324_Lin64.bin> |

*Table 2: Software tools and versions for MacOS and Linux*

## Hardware requirements

The labs are purely based on simulation and do not require any other hardware other than a computer.

# Lab setup

## Setting up Icarus Verilog

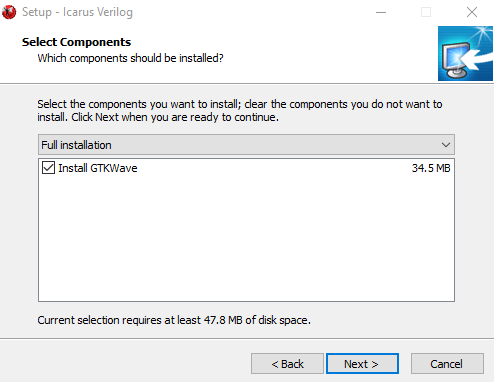
Icarus Verilog is a free compiler and Verilog simulator tool. You will be using this tool to compile and simulate your Verilog codes in the lab. For more information, see <http://iverilog.icarus.com/>.

For both Linux and MacOS, follow the installation instructions listed in [Software requirements](#_heading=h.2et92p0) and simply launch both iverilog and gtkwave from the terminal. To set up Icarus Verilog in a Windows OS, follow these steps:

1. Download the Icarus Verilog setup.exe listed in [Software requirements](#_heading=h.2et92p0).
2. Run the setup.exe on your Windows machine.
3. Accept all the default choices as you click through the installation.

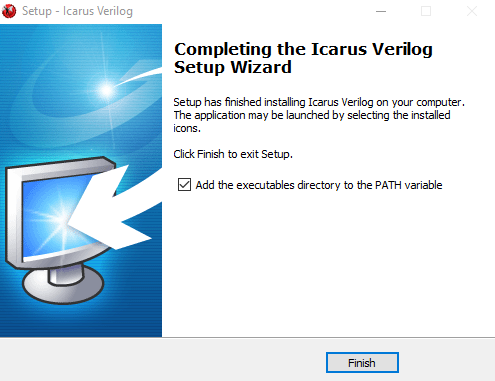
**Note:**

* + **We recommend that you install Icarus Verilog in C:\ and avoid long installation paths or any spaces in your installation path.**
  + Ensure that you select **Full installation** with the **Install GTKWave** option checked, as shown in the following diagram:



*Figure 1: Install GTKWave during installation*

1. When Icarus Verilog has completed installation, ensure that you have selected the “Add the executables directory to the PATH variable” option, shown in the following diagram. This option will allow us to easily run Icarus Verilog using command-line in a Windows terminal.



*Figure 2: Add Icarus to the PATH variable*

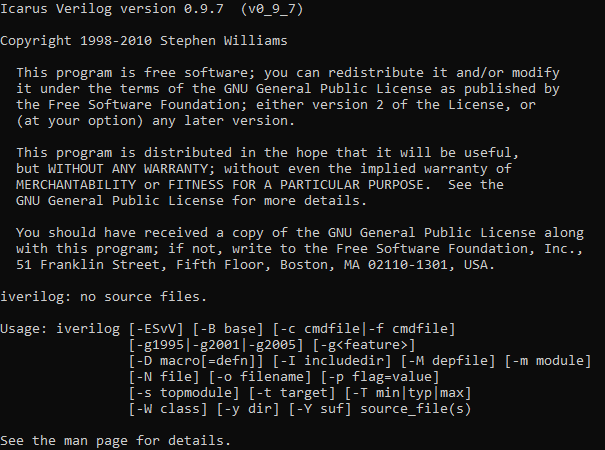
### Verifying successful Icarus Verilog installation

To ensure that the Icarus Verilog installation is successful, follow these steps:

1. Open a Windows command terminal and enter the following command:

C:\> iverilog -v

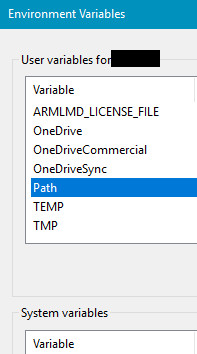
1. Ensure that you obtain the following output in the Windows command terminal:



*Figure 3: Snapshot of output of iverilog -v command*

If you see an error message instead, then try fixing your PATH variable in your Windows machine. To do this, follow these steps:

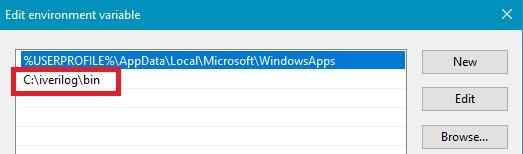
1. Open **Control Panel** in your Windows OS.
2. Select **System and Security** > **System** > **Advanced system settings** > **Environment Variables** > **User Variables** > **Path**, as shown in the snapshot below.



1. Double-click on **Path** in the user variable windows, select **New**, and add the following according to where Icarus Verilog is saved:

C:\iverilog\bin

For example:



*Figure 4: Example of added Iverilog path in Environment Variables*

## Setting up GNU Toolchain for the A-profile Architecture (Windows)

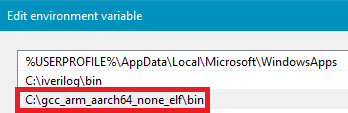
The lab exercises will require you to write software programs in Arm AArch64 Assembly and run these programs on the Arm Education Core. These programs are assembled and converted to hex code using the GNU Toolchain for the A-profile Architecture.

To set up the GNU toolchain, follow these steps:

1. Download the GNU Toolchain for the A-profile Architecture for Windows OS, as listed in Software requirements. This toolchain contains the GCC compiler necessary for the labs.
2. Extract the downloaded .tar.xz file using your favorite archive manager (or 7Zip). Then, open the extracted folder and extract the .tar file.
3. Rename the extracted file to a shorter name, like gcc\_arm\_aarch64\_none\_elf.
4. Move the folder to C:\. **We recommend that you avoid long folder paths or any spaces in your folder path.**
5. Open **Control Panel** in your Windows OS.
6. Select **System and Security** > **System** > **Advanced system settings** > **Environment Variables** > **User Variables** > **Path**.
7. Double-click on **Path** in the user variable windows, select **New**, and add the following according to where the toolchain is saved:

C:\gcc\_arm\_aarch64\_none\_elf\bin

For example:



*Figure 5: Example of added GNU compiler path in Environment Variables*

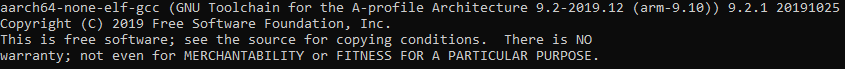
### Verifying successful toolchain installation

To ensure that the GNU Toolchain for the A-profile Architecture installation is successful, follow these steps:

1. Open a Windows command terminal and enter the following command (you may need to right-click on **Command Prompt** and run as **Administrator**):

C:\> aarch64-none-elf-gcc --version

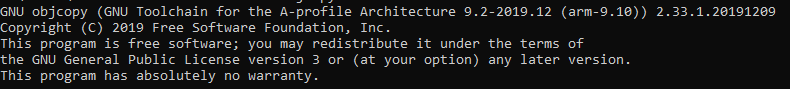
The expected output is as follows:



1. Run the following command to ensure that the objcopy tool is working:

aarch64-none-elf-objcopy --version

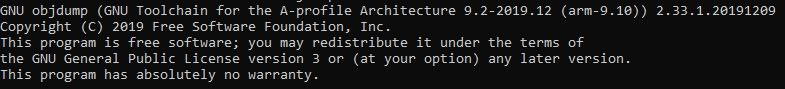
The expected output is as follows:



1. Run the following command to ensure that the objdump tool is working:

aarch64-none-elf-objdump --version

The expected output is as follows:



If you see any error message instead, try:

* Closing the terminal and reopening the terminal so that the command line sessions’ PATH variable is up-to-date.
* Double-checking that you have added your PATH variable correctly.

## Setting up GNU Toolchain for the A-profile Architecture (Linux)

The lab exercises will require you to write software programs in Arm AArch64 Assembly and run these programs on the Arm Education Core. These programs are assembled and converted to hex code using the GNU Toolchain for the A-profile Architecture.

To set up the GNU toolchain on Linux (Ubuntu), follow these steps:

1. Download the GNU Toolchain for the A-profile Architecture for X86\_64 Linux, as listed in Software requirements. This toolchain contains the GCC compiler necessary for the labs.
2. Extract the downloaded .tar.xz file. Then, open the extracted folder and extract the .tar file.
3. Rename the extracted file to a shorter name, like gcc\_arm\_aarch64\_none\_elf.
4. Move the extracted folder to a good place, such as your User’s home directory (/home/<username>/).
5. Finally, we’ll append this folder of executables to our system path. Execute the following command, replacing <path> with the fully qualified path (from /) to the gcc\_arm\_aarch64\_none\_elf folder:

PATH=/<path>/gcc\_arm\_aarch64\_none\_elf/bin:$PATH

for example, if you put the folder into your user’s home directory, you would use this exact command:

PATH=~/gcc\_arm\_aarch64\_none\_elf/bin:$PATH

to ensure the command above is executed and works for each new terminal you open, run the following command, which will edit your .bashrc (or .zshrc if you use zsh) to export the new PATH variable whenever you open a new terminal:

echo 'export PATH="$HOME/gcc\_arm\_aarch64\_none\_elf/bin:$PATH"' >> ~/.bashrc

To verify correct installation, simply follow the same instructions in Section 3.2.1 above, just in your Linux terminal instead.

## Setting up GTKWave

GTKWave is a wave viewer for standard Verilog VCD files. If you have installed Icarus Verilog following the procedures in [Setting up Icarus Verilog](#_heading=h.4d34og8), then you should have GTKWave as part of the Icarus Verilog package.

When using GTKWave in the labs, you need to use the gtkwave command to invoke GTKWave to view your simulated waveforms. Therefore, you will need to set the environment paths to point to where GTKWave is located by following these steps:

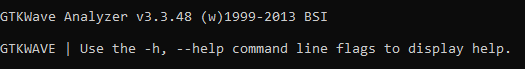
1. Open **Control Panel** in your Windows OS.
2. Select **System and Security** > **System** > **Advanced system settings** > **Environment Variables** > **User Variables** > **Path**.
3. Double-click on **Path** in the user variable windows, select **New**, and add the following path according to where the GTKWave folder in the Icarus Verilog file location. For example:

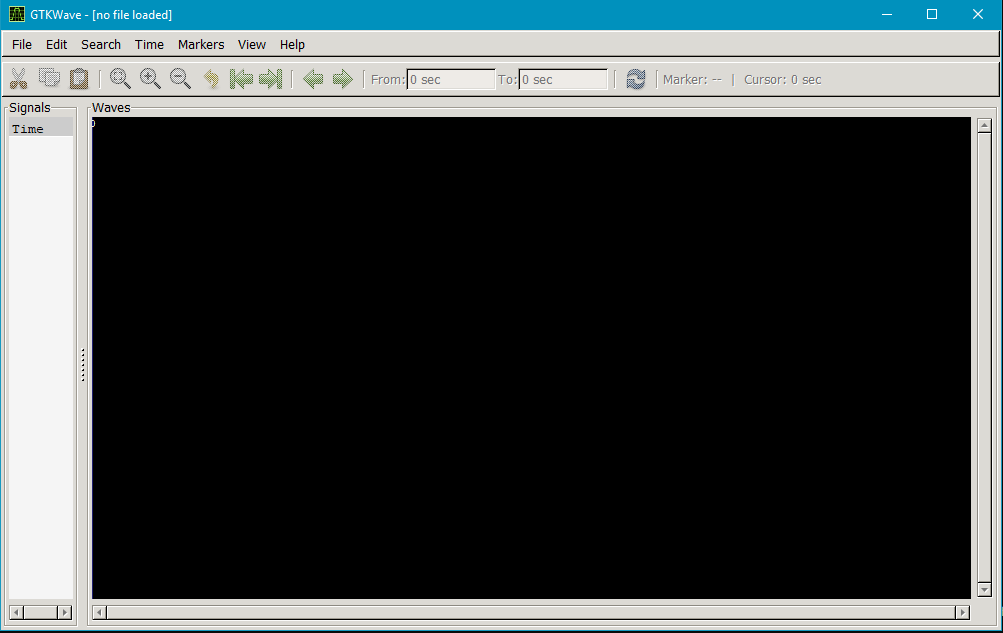
C:\iverilog\share\gtkwave

1. To ensure that GTKWave can be used in command-line, open a windows command terminal and enter the gtkwave command:

C:\> gtkwave

You should be able to observe the following outputs:





## Guidelines for creating a workspace for all lab work

Some of the tools installed do not respond well to long paths or spaces in file paths. We recommend that you:

1. Create a dedicated workspace folder for all your lab work. Ensure that this workspace hasn’t got a long folder path or spaces in the folder path. We recommend that you create a folder such as:

C:\Workspace

1. Make sure that all subsequent folders or files that you create in this workspace haven’t got any space in the file or folder names.

# Compiling and simulating a simple Verilog code

After you finish installing all the required software, compile and simulate a simple code to check that Iverilog can compile and simulate.

To compile and simulate a simple Verilog code of a FIFO, follow these steps:

1. Create a folder in your workspace; for example, C:\Workspace\FIFO\_TEST.
2. Note: You will need to enable viewing of file name extensions for windows in order to change the name of files from “.txt” to “.v”.
3. Use a good text editor (<https://notepad-plus-plus.org/>).
4. Use a text editor to create a new file called data.mem in FIFO\_TEST folder. Save the following contents in the file:

ab cd ef ab

1. Create a new file called FIFO\_test.v and save it in the FIFO\_TEST folder.
2. Copy and paste the following code in FIFO\_test.v:

module test();

reg clk, in, out;

reg [3:0] fifo;

reg [7:0] memory [0:3]; //four 8-bit data

reg [7:0] m0, m1, m2, m3;

initial

begin

clk = 0; in = 1; out = 0;

fifo[3] = 0; fifo[2] = 0; fifo[1] = 0; fifo[0] = 0;

$display("Reading memory file now.");

$readmemh("data.mem", memory);

m0 <= memory[0];

m1 <= memory[1];

m2 <= memory[2];

m3 <= memory[3];

#90 $finish;

end

always #5 clk = ~clk;

always@(posedge clk)

begin

out = fifo[3];

fifo[3] = fifo[2]; fifo[2] = fifo[1]; fifo[1] = fifo[0];

fifo[0] = in;

$display("At time = %t, fifo in = %d, fifo out = %d \n\n", $time, in, out);

end

endmodule

The above code demonstrates a simple FIFO with the use of Verilog system calls (functions) to access command-line arguments of simulation. For example, $finish, $display, $time, and $readmemh.

1. Save FIFO\_test.v and open a Windows command terminal.
2. In the Windows command terminal, change directory to the location where you saved FIFO\_test.v using the cd command. For example, enter the following command:

cd C:\Workspace\FIFO\_TEST

1. In Window Explorer click the “file name extensions” under view. Now rename each file by removing the “.txt” that now appears as part of the file name.
2. Run the following command so that Icarus Verilog compiles FIFO\_test.v and generate an output called fifo.vvp:

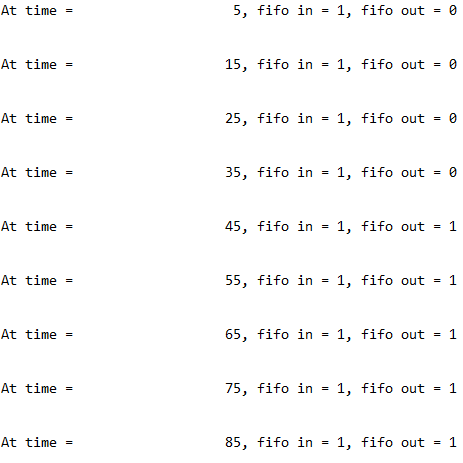
iverilog -o fifo.vvp FIFO\_test.v

1. Simulate the created fifo.vvp file by running the following command:

vvp fifo.vvp > run.log

The > run.log command outputs the display into a file called run.log saved in your current directory.

1. Open run.log and check you have the following content:



*Figure 6: Snapshot of run.log*

The text in run.log is derived from the following code in FIFO\_test.v:

$display**(**"At time = %t, fifo in = %d, fifo out = %d \n\n"**,** $time**,** in**,** out**);**

## Dumping out waveforms

When doing simulation, it is beneficial to be able to view your signals and registers’ values in terms of waveforms. To do this, you first need to dump out the values in a file. There are various dump file formats supported by GTKWave, such as VCD, LXT, and LXT2. In this course, we will use the LXT2 file format (InterLaced eXtensible Trace Version 2), which allows greater compression and access speeds.

To dump out waveforms, follow these steps:

1. Insert $dumpvars(0,test); in the **initial** block of FIFO\_test.v and resave the file:

initial

begin

$dumpvars(0,test);

clk = 0; in = 1; out = 0;

fifo[3] = 0; fifo[2] = 0; fifo[1] = 0; fifo[0] = 0;

$dumpvars(0,test) specifies that we want to dump all variables within module test and all module instances below if available.

1. Recompile the code since you have modified the code. This time, generate an output called fifo2.vvp, by entering the following command:

iverilog -o fifo2.vvp FIFO\_test.v

1. Simulate the created fifo2 and specify the dump file format in the switch by running the following command:

vvp fifo2.vvp -lx2 > run.log

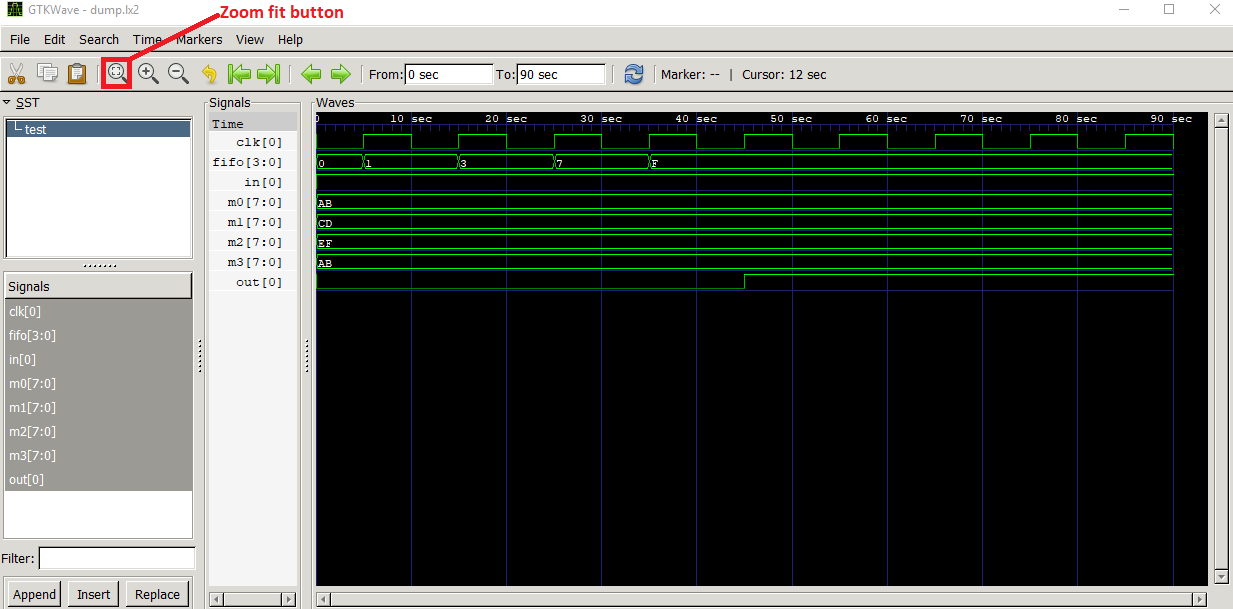
The specified command outputs a file called dump.lx2.

1. Launch GTKWave to view the VCD dump file by running the following command:

gtkwave dump.lx2

1. In the GTKWave window, double-click on the test module. All the signals in the test module should appear in the tab below the module. Drag and drop these signals to the Waves tab to view the waveforms. Alternatively, you can select all the signals and click the **Insert** button.

You can also use the **Zoom Fit** button to fit the waveform in the time range of your simulation, as shown in the following diagram.



*Figure 7: Snapshot of GTKWave and test module signals*

From the waveforms, you will observe that:

* The clk pulse width is 5s (and thus cycle = 10s) because it was specified by “always #5 clk = ~clk;” in FIFO\_test.v.
* The fifo values update at each positive edge of clock, as expected.
* The simulation ends after 90 seconds because “#90 $finish;” was specified in FIFO\_test.v.
* $readmemh reads contents in data.mem file. The data are now in registers m0, m1, m2, and m3.

# Troubleshooting

1. **GTKWave has error launching**

Solution:

* Ensure that you have followed instruction in [Setting up GTKWave](#_heading=h.35nkun2) and that the environment variables are set correctly.
* Ensure that your workspace or waveform file that you are trying to view in GTKWave is not located in a file path name that is too long. We recommend creating a workspace that has a short path. See [Guidelines for creating a workspace for all lab work](#_heading=h.1ksv4uv).

# Additional references

**MinGW**

* <http://www.mingw.org/wiki/Getting_Started>

**GTKWave**

* <http://gtkwave.sourceforge.net/>
* Manual: <http://gtkwave.sourceforge.net/gtkwave.pdf>

**Arm Education Core**

* Introduction to Arm Education Core document (provided in the Getting Started folder)

**Icarus Verilog user guide**

* <https://iverilog.fandom.com/wiki/User_Guide>
* <https://iverilog.fandom.com/wiki/Installation_Guide>

**Icarus Verilog commands and arguments**

* <https://iverilog.fandom.com/wiki/Iverilog_Flags>
* <https://iverilog.fandom.com/wiki/Vvp_Flags>

**Simulation**

* <https://iverilog.fandom.com/wiki/Simulation>