arm Education

Computer Architecture Course

Getting Started Guide

Issue 2.0

arm Education

Contents

1 Introduction

1.1 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.

2 Requirements

2.1 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 Class-Toolchain repository on the class GitHub. 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)	i) 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	All labs	https://developer.arm.com/tools-and-software/open-source-	**Linux:** version 9.2-2019.12 X86_64 Linux AArch64 ELF

the A-profile Architecture		software/developer-tools/gnu-toolchain/gnu-a/downloads	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

2.2 Hardware requirements

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

3 Lab setup

3.1 Setting up Icarus Verilog and GTKWave (Linux)

3.1.1 Installing Icarus Verilog on Ubuntu (Ver. 24.04.1 LTS):

In general, Ubuntu is the most compatible and well-documented Linux distribution. However, most package installation takes place in the console. So, to install Icarus Verilog make sure it's done exclusively through the console to avoid unnecessary troubleshooting.

Therefore, to install Icarus Verilog on Ubuntu (Ver. 24.04.1 LTS), open the console and follow the steps below:

1) Make sure all your repositories are up to date with: sudo apt-get update
2) Then, once that's done, type: sudo apt-get install verilog
3) Authorize the installation and Verilog will be installed.

3.1.2 Installing GTKWave on Ubuntu (Ver. 24.04.1 LTS):

To install GTKWave on Ubuntu, simply follow the same process:

1) If you already updated all repos, you do not have to do so again.
2) In the console, type: sudo apt-get install gtkwave
3) Authorize the installation and GTKWave will be installed.

Now, all that's left is to set-up the GNU Toolchain for the A-profile Architecture (in Section 3.3), and compile and create the Verilog simulation workspace (in Section 4.0). The rest of the "Getting Started" guide is OK to follow. The instructions are not distro-dependent, just make sure you're carefully following the steps on the specified "Linux" sections.

3.1.3 Installing Icarus Verilog through the Konsole; OpenSUSE (Ver. Leap 15.6):

If you're on OpenSUSE you might already be familiar with the various ways of installing applications. For the purposes of this guide, I will be going over how to install Icarus Verilog and GTKWave through the console ("konsole"), and through YaST Software Management.

The quickest and simplest way is through the console. However, you may also use YaST if you're new to Linux, or if you're uncomfortable with the console.

To install Verilog on OpenSUSE, open the “konsole” and follow the steps below:

1) First, make sure all your repositories are up to date with: sudo zypper dup
2) Once that’s done, type: sudo zypper install iverilog
3) Authorize the installation and Verilog will be installed.

Note: You may see red errors but you’re safe to ignore them as $bash will cycle through previous deprecated Leap versions while searching for a compatible version.

Installing GTKWave through the Konsole; OpenSUSE (Ver. Leap 15.6):

To install GTKWave on OpenSUSE, simply follow the same process through “konsole” or “YaST”.

3.1.4 Installing GTKWave through the console:

1) If you already updated all repos, you do not have to do so again.
2) In the console, type: sudo zypper install gtkwave
3) Authorize the installation and GTKWave will be installed.

Note: You may see red errors but you’re safe to ignore them as $bash will cycle through previous deprecated Leap versions while searching for a compatible version.

Now, all that’s left is to set-up the GNU Toolchain for the A-profile Architecture (in Section 3.3), and compile and create the Verilog simulation workspace (in Section 4.0). The rest of the “Getting Started” guide is OK to follow. The instructions are not distro-dependent, just make sure you’re carefully following the steps on the specified “Linux” sections.

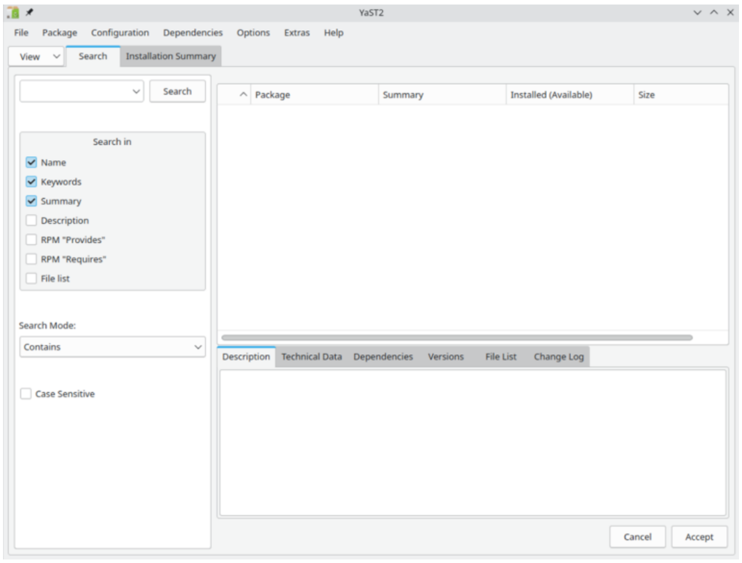
Installing Icarus Verilog with YaST Software Management; OpenSUSE (Ver. Leap 15.6):

OpenSUSE offers GUI software management with console-like capabilities. So, I will try to guide you through this process if the console is not your preferable method of installation.

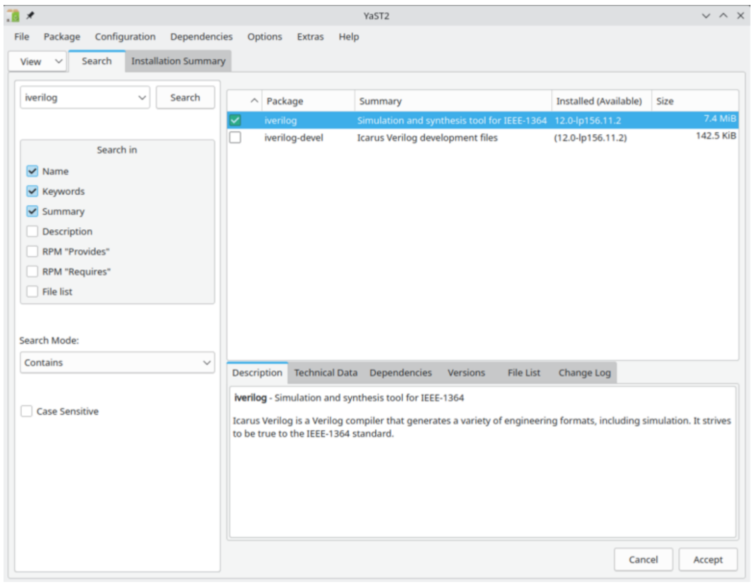
3.1.5 Installing Verilog through YaST:

1) First, on the application launcher look for "YaST Software Management"

2) Once there, you'll see the following window:



3) On YaST, on the top left's "Search" tab search for "iverilog"



	Package	Summary	Installed (Available)	Size
✓	iverilog	Simulation and synthesis tool for IEEE-1364	12.0-lp156.11.2	7.4 MiB
☐	iverilog-devel	Icarus Verilog development files	(12.0-lp156.11.2)	142.5 KiB

iverilog - Simulation and synthesis tool for IEEE-1364

Icarus Verilog is a Verilog compiler that generates a variety of engineering formats, including simulation. It strives to be true to the IEEE-1364 standard.

4) Checkmark “iverilog” and click Accept on the bottom right.

5) Wait for the installation to finish, and Verilog is now installed!

3.1.6 Installing GTKWave with YaST Software Management; OpenSUSE (Ver. Leap 15.6):

To install GTKWave through YaST Software Management follow the same previous steps when you installed Verilog.

Once Verilog and GTK wave are installed through YaST, all that’s left is to set-up the GNU Toolchain for the A-profile Architecture (in Section 3.3), and compile and create the Verilog simulation workspace (in Section 4.0). The rest of the “Getting Started” guide is OK to follow. The instructions are not distro-dependent, just make sure you’re carefully following the steps on the specified “Linux” sections.

3.2 Setting up Icarus Verilog and GTKWave (MacOS)

In this tutorial, homebrew will need to be installed as the package manager will be used to download icarus-verilog and gtkwave. Installation information can be found here. Additionally, the reference for the following instructions can be found here (note: they are outdated but have a general scheme of what to do).

1. Download Icarus-Verilog (documentation) using homebrew, you can use the following homebrew command below:

```
(base) user@User-MBP ~ % brew install icarus-verilog
```

2. Download GTK wave:

```
(base) user@User-MBP ~ % brew --cask install gtkwave
```

3. Install Perl Switch:

```
(base) user@User-MBP ~ % cpan install Switch
```

4. Check the installation location of Switch:

```
(base) user@User-MBP ~ % perl -V:'installsitelib'
```

5. If the installation location is not the following:

```
/usr/local/Cellar/perl/...
```

6. Then, copy Switch to the following location using this command:

```
(base) user@User-MBP ~ % sudo cp /usr/local/Cellar/perl/5.*/lib/perl5/site_perl/5.*/Switch.pm
/Library/Perl/5.*/
```

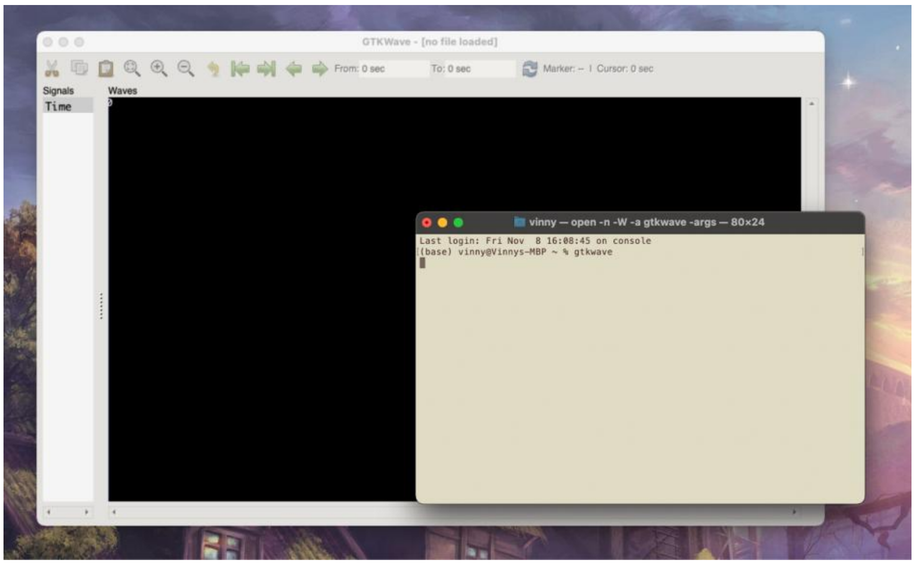
7. Add the following command to your bash profile (note: you will need to add a .bash_profile if it is not created already to alias):

```
(base) user@User-MBP ~ % alias gtkwave = /Applications/gtkwave.app/Contents/Resources/bin/gtkwave

OR

(base) user@User-MBP ~ % export PATH= /Applications/gtkwave.app/Contents/Resources/bin/:$PATH
```

8. Verify gtkwave will launch using the previously created alias:

```
(base) user@User-MBP ~ % gtkwave
```

The following window should pop up if successfully installed:


GTKWave - [no file loaded]
From: 0 sec
To: 0 sec
Marker: -- | Cursor: 0 sec
Signals
Waves
Time
vinny — open -n -W -a gtkwave -args — 80×24

```
Last login: Fri Nov  8 16:08:45 on console
(base) vinny@Vinnys-MBP ~ % gtkwave
```

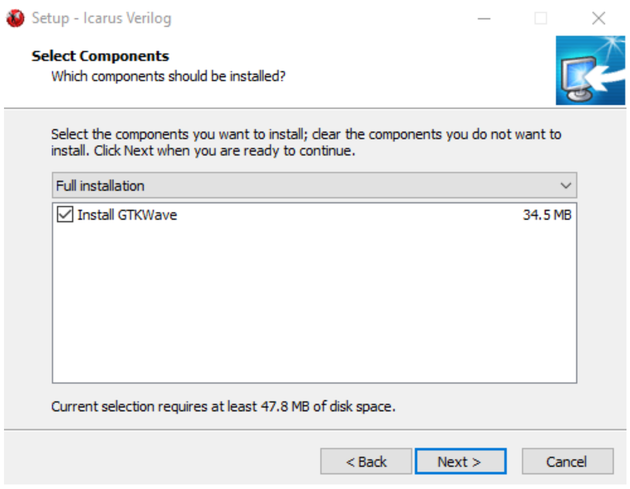
3.3 Setting up Icarus Verilog (Windows)

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 https://steveicarus.github.io/iverilog/.

Commented [VA1]: This link does not work

To set up Icarus Verilog in a Windows OS, follow these steps:

1. Download the Icarus Verilog setup.exe listed in Software requirements.
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:


Setup - Icarus Verilog

Select Components

Which components should be installed?

Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

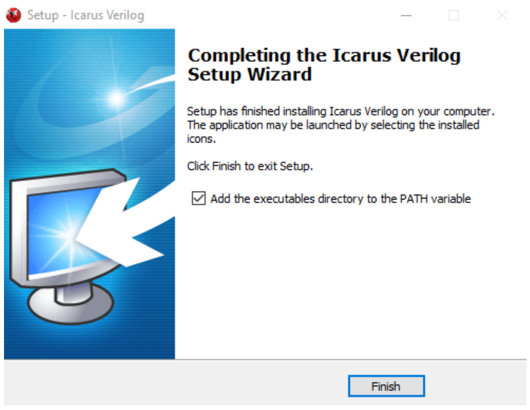
Full installation

☑ Install GTKWave 34.5 MB

Current selection requires at least 47.8 MB of disk space.

Figure 1: Install GTKWave during installation

4. 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.



Completing the Icarus Verilog Setup Wizard

Setup has finished installing Icarus Verilog on your computer. The application may be launched by selecting the installed icons.

Click Finish to exit Setup.

☑ Add the executables directory to the PATH variable

Figure 2: Add Icarus to the PATH variable

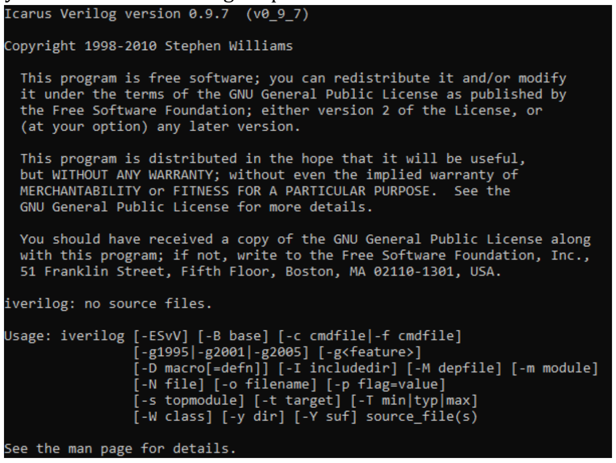
3.3.1 Verifying successful Icarus Verilog installation

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
```

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



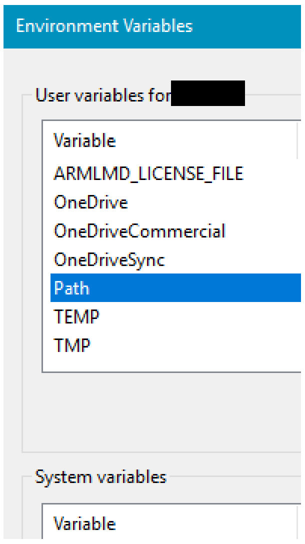
```
Icarus Verilog version 0.9.7  (v0_9_7)

Copyright 1998-2010 Stephen Williams

  This program is free software; you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation; either version 2 of the License, or
  (at your option) any later version.

  This program is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License for more details.

  You should have received a copy of the GNU General Public License along
  with this program; if not, write to the Free Software Foundation, Inc.,
  51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

iverilog: no source files.

Usage: iverilog [-ESvV] [-B base] [-c cmdfile|-f cmdfile]
                [-g1995|-g2001|-g2005] [-g<feature>]
                [-D macro[=defn]] [-I includedir] [-M depfile] [-m module]
                [-N file] [-o filename] [-p flag=value]
                [-s topmodule] [-t target] [-T min|typ|max]
                [-W class] [-y dir] [-Y suf] source_file(s)

See the man page for details.
```

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.



3. 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:

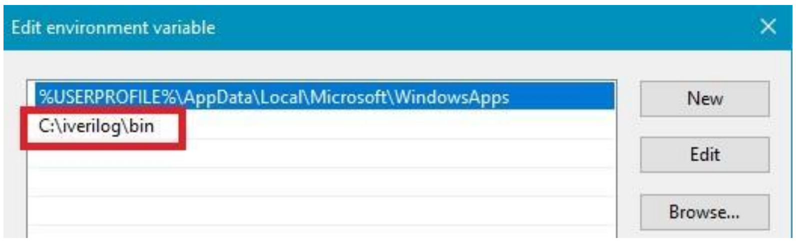

Edit environment variable
%USERPROFILE%\AppData\Local\Microsoft\WindowsApps
C:\iverilog\bin
New
Edit
Browse...

Figure 4: Example of added Iverilog path in Environment Variables

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

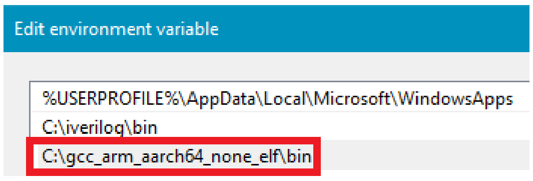
The lab exercises will require you to write software programs in Arm AArch64 Assembly and run these programs on the Arm Education Core. These programs are assembled and converted to hex code using the GNU Toolchain for the A-profile Architecture.

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:



```
%USERPROFILE%\AppData\Local\Microsoft\WindowsApps
C:\iverilog\bin
C:\gcc_arm_aarch64_none_elf\bin
```

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

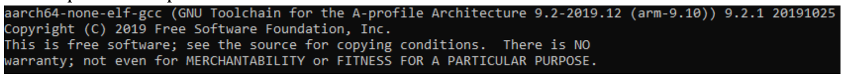
3.4.1 Verifying successful toolchain installation

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:

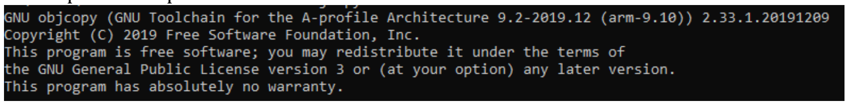


```
aarch64-none-elf-gcc (GNU Toolchain for the A-profile Architecture 9.2-2019.12 (arm-9.10)) 9.2.1 20191025
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

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

```
aarch64-none-elf-objcopy --version
```

The expected output is as follows:

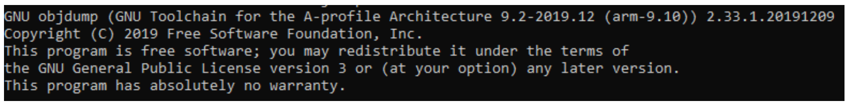


```
GNU objcopy (GNU Toolchain for the A-profile Architecture 9.2-2019.12 (arm-9.10)) 2.33.1.20191209
Copyright (C) 2019 Free Software Foundation, Inc.
This program is free software; you may redistribute it under the terms of
the GNU General Public License version 3 or (at your option) any later version.
This program has absolutely no warranty.
```

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

```
aarch64-none-elf-objdump --version
```

The expected output is as follows:



```
GNU objdump (GNU Toolchain for the A-profile Architecture 9.2-2019.12 (arm-9.10)) 2.33.1.20191209
Copyright (C) 2019 Free Software Foundation, Inc.
This program is free software; you may redistribute it under the terms of
the GNU General Public License version 3 or (at your option) any later version.
This program has absolutely no warranty.
```

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.

3.5 Setting up GNU Toolchain for the A-profile Architecture (Linux & MacOS)

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.

3.6 Setting up GTKWave (Windows)

GTKWave is a wave viewer for standard Verilog VCD files. If you have installed Icarus Verilog following the procedures in Setting up Icarus Verilog, 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**.

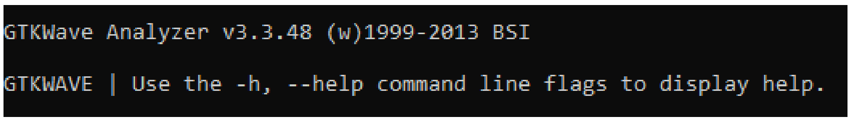
3. Double-click on **Path** in the user variable windows, select **New**, and add the following path according to where the GTKWave folder in the Icarus Verilog file location. For example:

   ```
   C:\iverilog\share\gtkwave
   ```

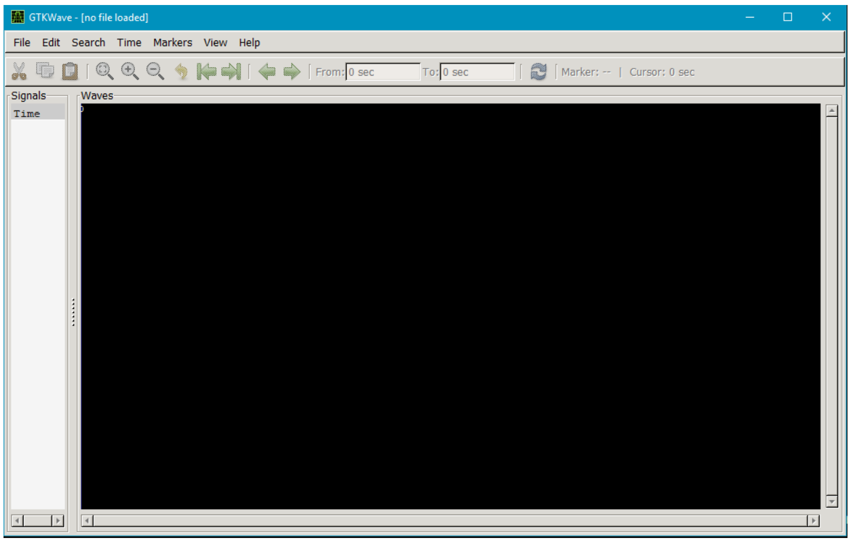
4. To ensure that GTKWave can be used in command-line, open a windows command terminal and enter the gtkwave command:

   ```
   C:\> gtkwave
   ```

   You should be able to observe the following outputs:



```
GTKWave Analyzer v3.3.48 (w)1999-2013 BSI

GTKWAVE | Use the -h, --help command line flags to display help.
```


GTKWave - [no file loaded]
File Edit Search Time Markers View Help
From: 0 sec
To: 0 sec
Marker: -- | Cursor: 0 sec
Signals
Time
Waves

3.7 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
   ```

2. 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.

4 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
```

5. Create a new file called FIFO_test.v and save it in the FIFO_TEST folder.
6. 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.

7. Save FIFO_test.v and open a Windows command terminal.
8. 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
```

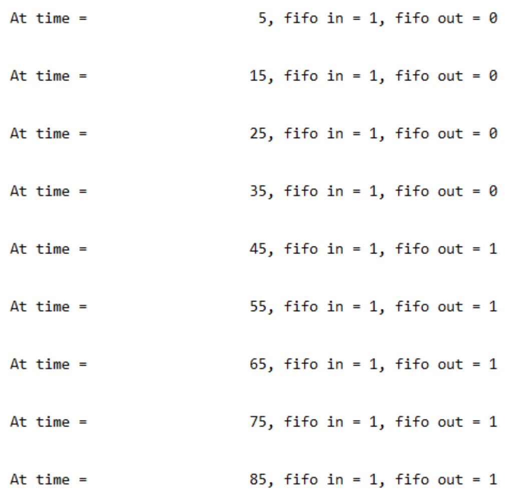
9. In Window Explorer click the “file name extensions” under view. Now rename each file by removing the “.txt” that now appears as part of the file name.
10. Run the following command so that Icarus Verilog compiles FIFO_test.v and generate an output called fifo.vvp:

```
iverilog -o fifo.vvp FIFO_test.v
```

11. 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.
12. Open run.log and check you have the following content:



```
At time =                     5, fifo in = 1, fifo out = 0

At time =                    15, fifo in = 1, fifo out = 0

At time =                    25, fifo in = 1, fifo out = 0

At time =                    35, fifo in = 1, fifo out = 0

At time =                    45, fifo in = 1, fifo out = 1

At time =                    55, fifo in = 1, fifo out = 1

At time =                    65, fifo in = 1, fifo out = 1

At time =                    75, fifo in = 1, fifo out = 1

At time =                    85, fifo in = 1, fifo out = 1
```

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);
```

4.3 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.
2. 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
```

3. 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.
4. Launch GTKWave to view the VCD dump file by running the following command:

```
gtkwave dump.lx2
```

5. 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.

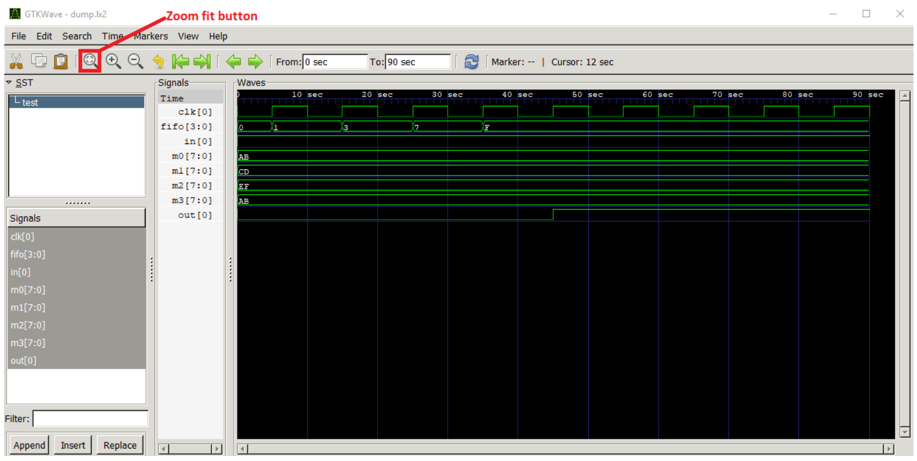


Figure 7: Snapshot of GTKWave and test module signals

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.

5 Troubleshooting

1. **GTKWave has error launching**

   Solution:
   - Ensure that you have followed instruction in Setting up GTKWave 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.

6 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