Documentation

Verilator & GTKWave Installation with Half Adder Simulation on Ubuntu

Prepared By:
Name: Muhammad Taha
Department: Electrical Engineering

Date of Submission: July 2025

Overview:

This documentation provides a step-by-step guide to installing **Verilator** & **GTKWaves** on **Ubuntu**, along with the **challenges faced** and how they were resolved during the installation process. This will help other beginners avoid common issues and get **Verilator** with **GTKwaves** working smoothly.

□**What is verilator?**
Verilator converts Verilog code (which describes how a digital circuit works) into C++ or SystemC code, which can then be compiled and run as a fast simulation. This lets engineers test and debug how their circuit behaves before building it in hardware (like FPGA or ASIC).

□What is GTKWave?
GTKWave is an open-source waveform viewer used to visualize digital signals from simulation outputs, typically generated by Verilog or VHDL simulators like Verilator or ModelSim.

System Requirement:

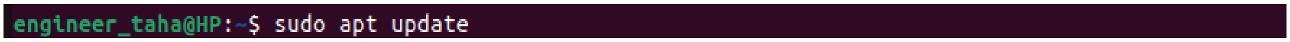
- OS: Ubuntu 20.04 or newer
- Internet connection
- Terminal access with **sudo** command

INSTALLATION OF VERILATOR

Step-by-Step Installation:

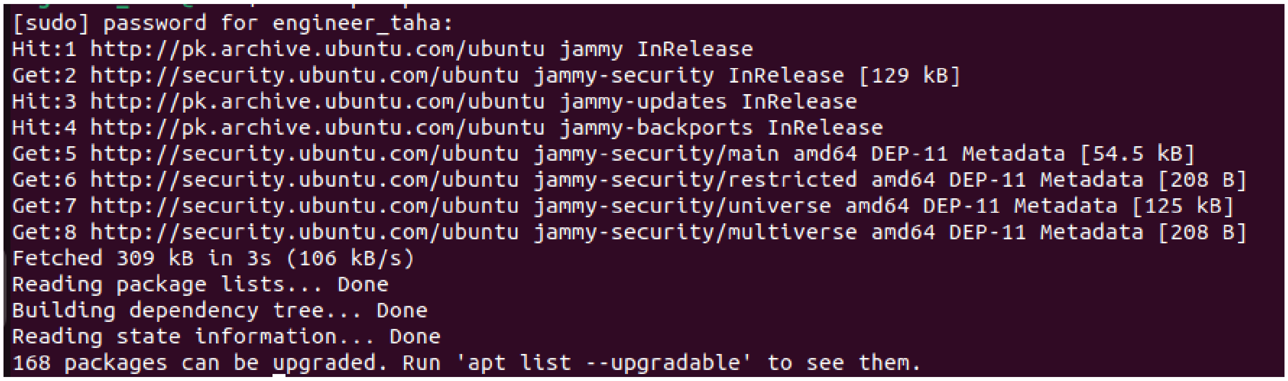
1. Update the System:

Input:



```
engineer_taha@HP:~$ sudo apt update
```

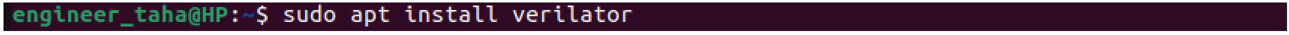
Output:



```
[sudo] password for engineer_taha:
Hit:1 http://pk.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Hit:3 http://pk.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:4 http://pk.archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:5 http://security.ubuntu.com/ubuntu jammy-security/main amd64 DEP-11 Metadata [54.5 kB]
Get:6 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 DEP-11 Metadata [208 B]
Get:7 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 DEP-11 Metadata [125 kB]
Get:8 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 DEP-11 Metadata [208 B]
Fetched 309 kB in 3s (106 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
168 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

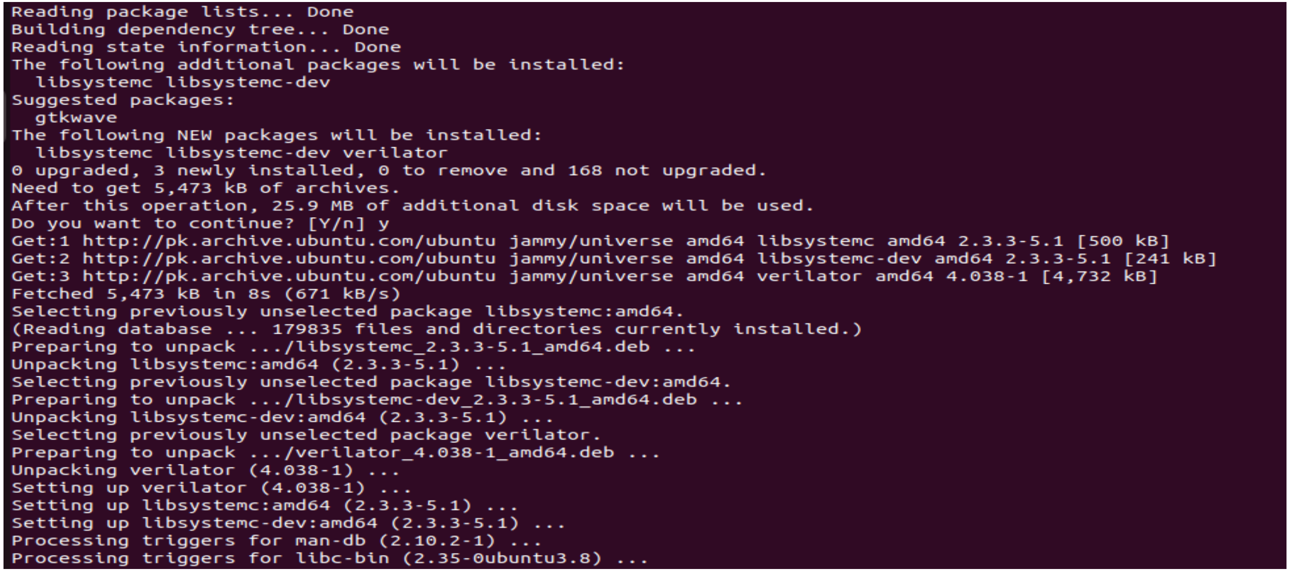
2. Install Verilator:

Input:



```
engineer_taha@HP:~$ sudo apt install verilator
```

Output:

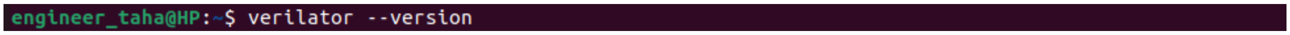


```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libsystemc libsystemc-dev
Suggested packages:
  gtkwave
The following NEW packages will be installed:
  libsystemc libsystemc-dev verilator
0 upgraded, 3 newly installed, 0 to remove and 168 not upgraded.
Need to get 5,473 kB of archives.
After this operation, 25.9 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://pk.archive.ubuntu.com/ubuntu jammy/universe amd64 libsystemc amd64 2.3.3-5.1 [500 kB]
Get:2 http://pk.archive.ubuntu.com/ubuntu jammy/universe amd64 libsystemc-dev amd64 2.3.3-5.1 [241 kB]
Get:3 http://pk.archive.ubuntu.com/ubuntu jammy/universe amd64 verilator amd64 4.038-1 [4,732 kB]
Fetched 5,473 kB in 8s (671 kB/s)
Selecting previously unselected package libsystemc:amd64.
(Reading database ... 179835 files and directories currently installed.)
Preparing to unpack .../libsystemc_2.3.3-5.1_amd64.deb ...
Unpacking libsystemc:amd64 (2.3.3-5.1) ...
Selecting previously unselected package libsystemc-dev:amd64.
Preparing to unpack .../libsystemc-dev_2.3.3-5.1_amd64.deb ...
Unpacking libsystemc-dev:amd64 (2.3.3-5.1) ...
Selecting previously unselected package verilator.
Preparing to unpack .../verilator_4.038-1_amd64.deb ...
Unpacking verilator (4.038-1) ...
Setting up verilator (4.038-1) ...
Setting up libsystemc:amd64 (2.3.3-5.1) ...
Setting up libsystemc-dev:amd64 (2.3.3-5.1) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.8) ...
```

3. **Verification:**

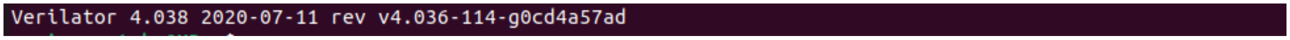
   After installing Verilator, it's important to confirm whether the installation was successful. This can be done using the following command;

   Input:



```
engineer_taha@HP:~$ verilator --version
```

Output:



```
Verilator 4.038 2020-07-11 rev v4.036-114-g0cd4a57ad
```

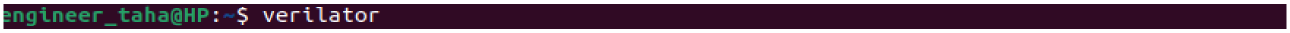
4. **Result:**

   Verilator is Successfully installed.

Problem faced by Installing the Verilator:

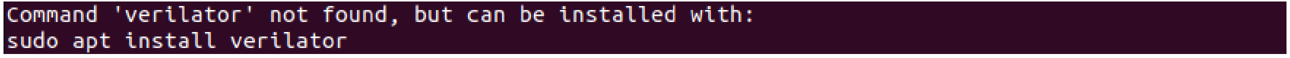
1. **Command 'Verilator' not found:**

   Input:



```
engineer_taha@HP:~$ verilator
```

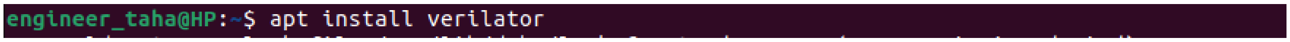
Output:



```
Command 'verilator' not found, but can be installed with:
sudo apt install verilator
```

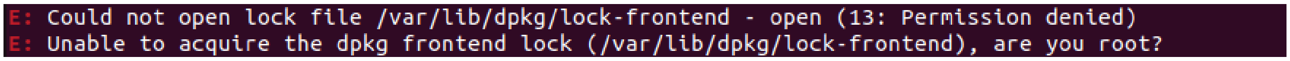
2. **Permission Error:**

   Input:



```
engineer_taha@HP:~$ apt install verilator
```

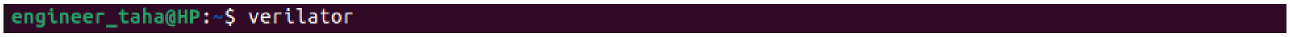
Output:



```
E: Could not open lock file /var/lib/dpkg/lock-frontend - open (13: Permission denied)
E: Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontend), are you root?
```

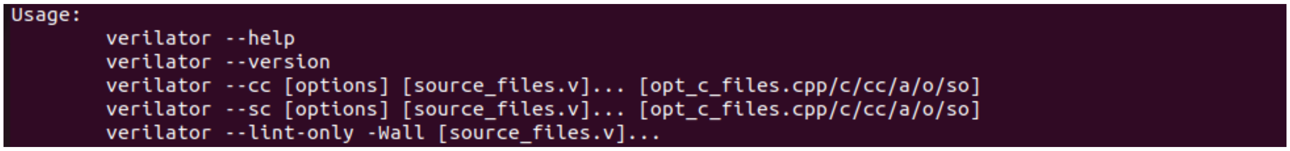
3. **Verification Error:**

   Input:



```
engineer_taha@HP:~$ verilator
```

Output:



```
Usage:
        verilator --help
        verilator --version
        verilator --cc [options] [source_files.v]... [opt_c_files.cpp/c/cc/a/o/so]
        verilator --sc [options] [source_files.v]... [opt_c_files.cpp/c/cc/a/o/so]
        verilator --lint-only -Wall [source_files.v]...
```

Explanation:

To verify whether Verilator is installed or to check its version, you can use the command-line options shown here. Simply copy and run the appropriate command, and you'll reach the desired result.

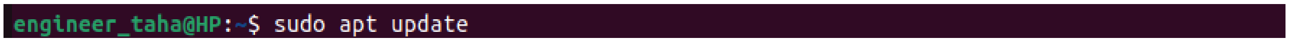
INSTALLATION OF GTKWave

Step-by-Step Installation:

1. **Update the System:**

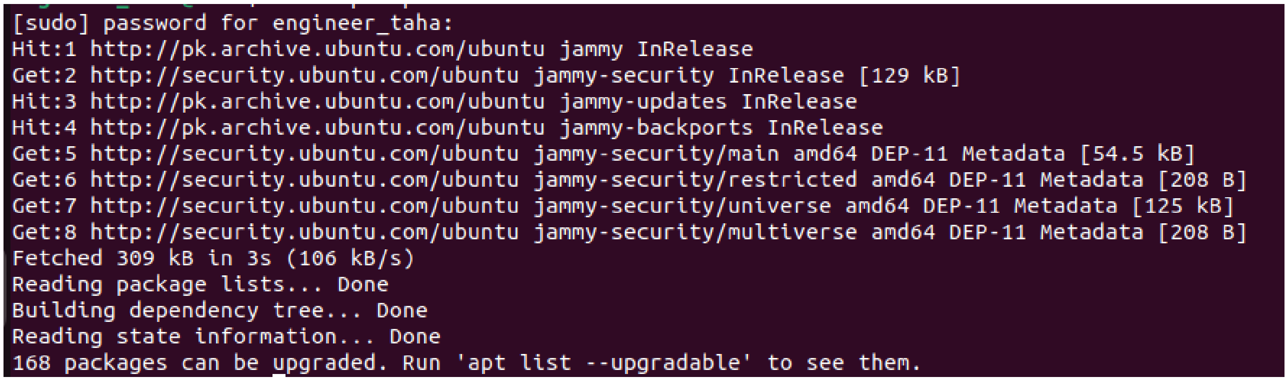
   Although we already updated the system during the Verilator installation, it's good practice to ensure your package list is current.

   Input:



```
engineer_taha@HP:~$ sudo apt update
```

   Output:

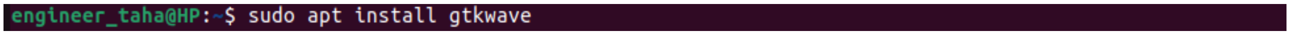


```
[sudo] password for engineer_taha:
Hit:1 http://pk.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Hit:3 http://pk.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:4 http://pk.archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:5 http://security.ubuntu.com/ubuntu jammy-security/main amd64 DEP-11 Metadata [54.5 kB]
Get:6 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 DEP-11 Metadata [208 B]
Get:7 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 DEP-11 Metadata [125 kB]
Get:8 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 DEP-11 Metadata [208 B]
Fetched 309 kB in 3s (106 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
168 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

   Note: *You can skip this step if you've already recently updated.*

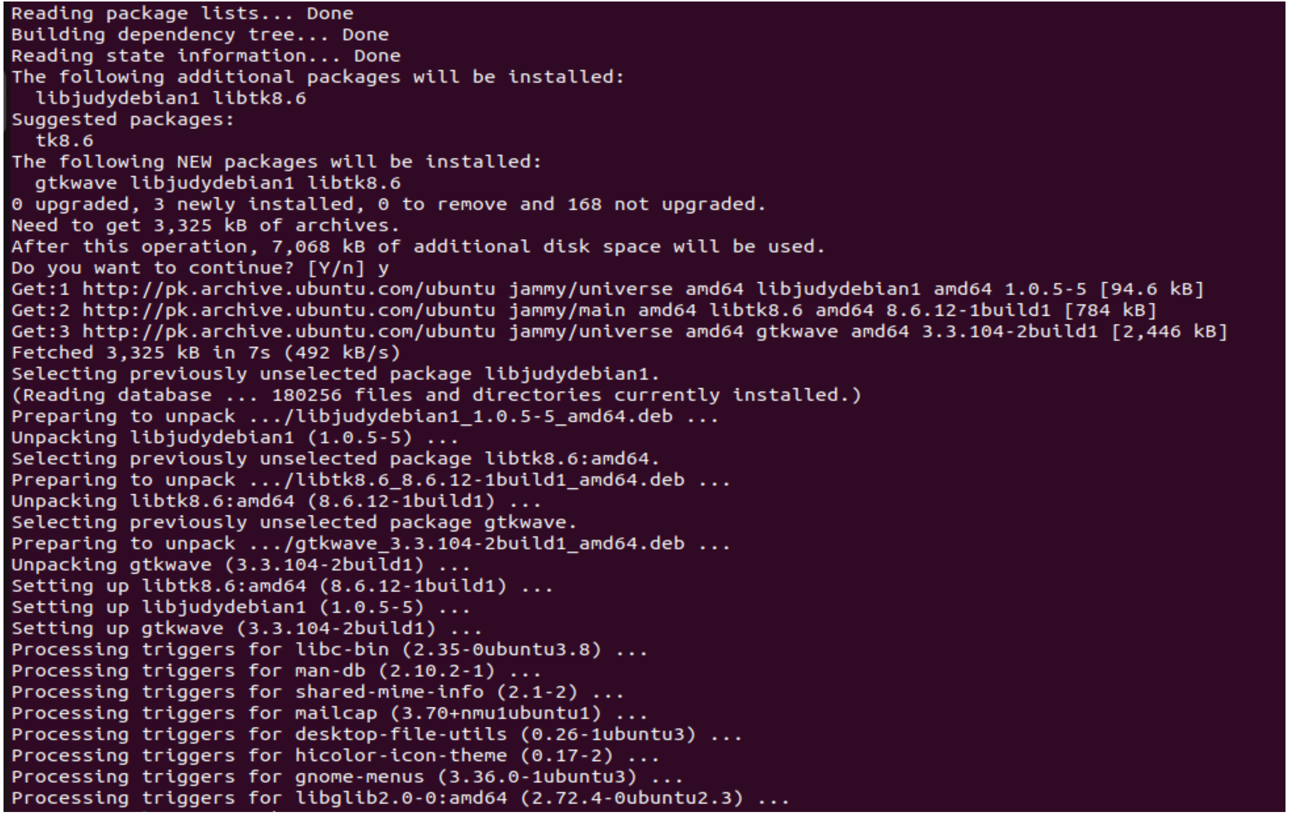
2. **Install GTKWaves:**

   Input:



```
engineer_taha@HP:~$ sudo apt install gtkwave
```

   Output:

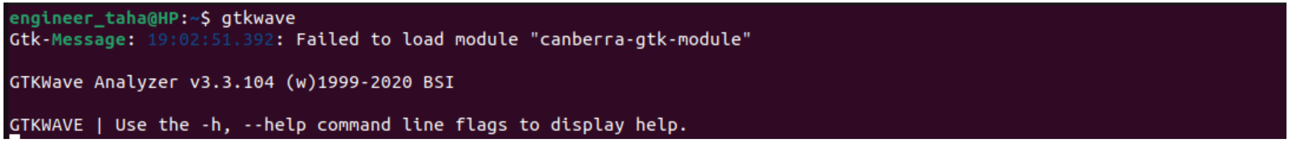


```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libjudydebian1 libtk8.6
Suggested packages:
  tk8.6
The following NEW packages will be installed:
  gtkwave libjudydebian1 libtk8.6
0 upgraded, 3 newly installed, 0 to remove and 168 not upgraded.
Need to get 3,325 kB of archives.
After this operation, 7,068 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://pk.archive.ubuntu.com/ubuntu jammy/universe amd64 libjudydebian1 amd64 1.0.5-5 [94.6 kB]
Get:2 http://pk.archive.ubuntu.com/ubuntu jammy/main amd64 libtk8.6 amd64 8.6.12-1build1 [784 kB]
Get:3 http://pk.archive.ubuntu.com/ubuntu jammy/universe amd64 gtkwave amd64 3.3.104-2build1 [2,446 kB]
Fetched 3,325 kB in 7s (492 kB/s)
Selecting previously unselected package libjudydebian1.
(Reading database ... 180256 files and directories currently installed.)
Preparing to unpack .../libjudydebian1_1.0.5-5_amd64.deb ...
Unpacking libjudydebian1 (1.0.5-5) ...
Selecting previously unselected package libtk8.6:amd64.
Preparing to unpack .../libtk8.6_8.6.12-1build1_amd64.deb ...
Unpacking libtk8.6:amd64 (8.6.12-1build1) ...
Selecting previously unselected package gtkwave.
Preparing to unpack .../gtkwave_3.3.104-2build1_amd64.deb ...
Unpacking gtkwave (3.3.104-2build1) ...
Setting up libtk8.6:amd64 (8.6.12-1build1) ...
Setting up libjudydebian1 (1.0.5-5) ...
Setting up gtkwave (3.3.104-2build1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.8) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for shared-mime-info (2.1-2) ...
Processing triggers for mailcap (3.70+nmu1ubuntu1) ...
Processing triggers for desktop-file-utils (0.26-1ubuntu3) ...
Processing triggers for hicolor-icon-theme (0.17-2) ...
Processing triggers for gnome-menus (3.36.0-1ubuntu3) ...
Processing triggers for libglib2.0-0:amd64 (2.72.4-0ubuntu2.3) ...
```

3. **Verification:**

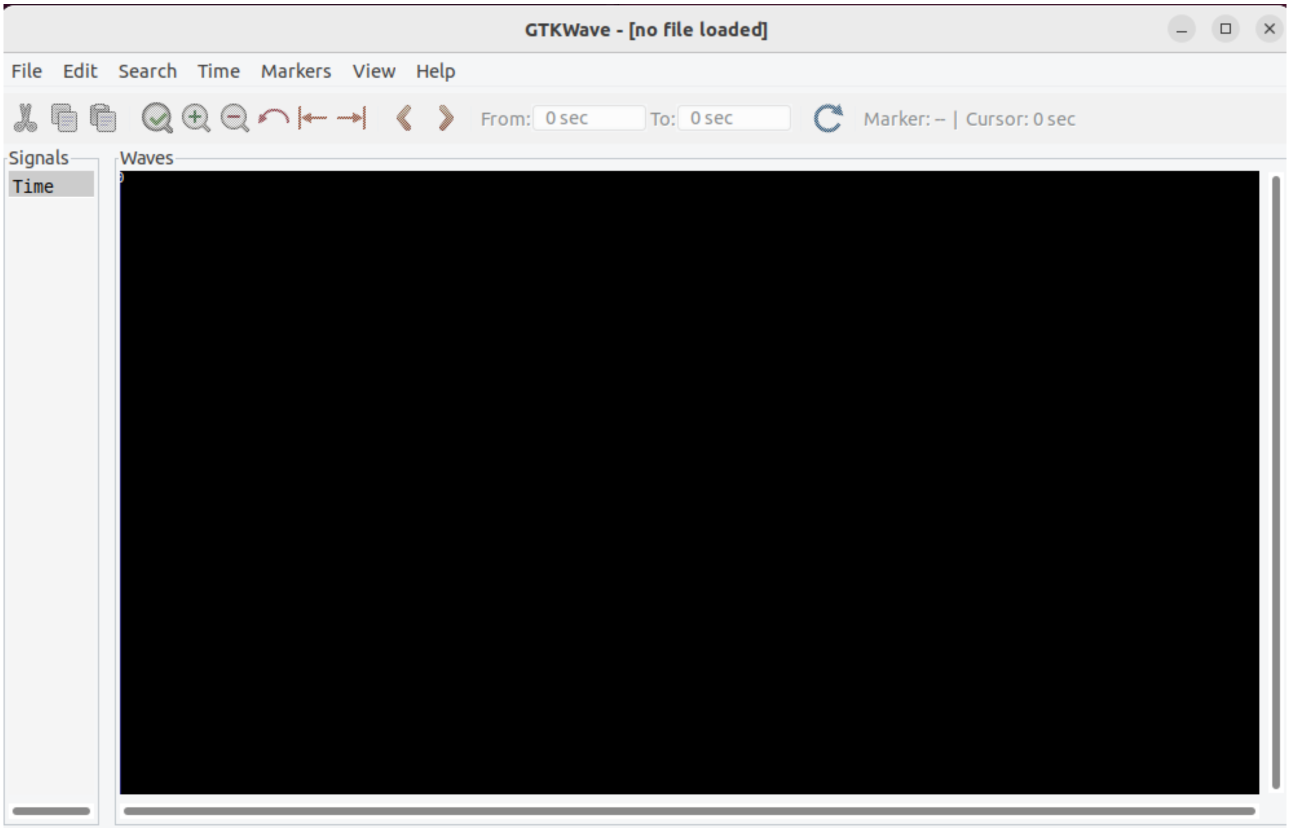
   To confirm that GTKWave was installed correctly, run:

   Input:



```
engineer_taha@HP:~$ gtkwave
Gtk-Message: 19:02:51.392: Failed to load module "canberra-gtk-module"

GTKWave Analyzer v3.3.104 (w)1999-2020 BSI

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

   Output:



   Note: *GTKWave is open.*

Problem faced by Installing the GTKWave:

During the installation of GTKWave, I didn't **encounter any issues**. The process was smooth, and the software was installed successfully using the standard **apt** command without requiring any additional configurations or fixes.

Half Adder Simulation using Verilator and GTKWave

Introduction:

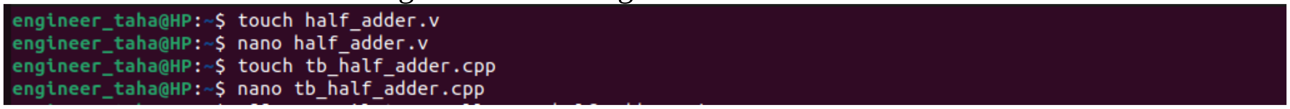
The objective of this section is to simulate a simple **Half Adder circuit** using **Verilator** and visualize its waveform using **GTKWave**.

Tools Required:

- **Verilator** – For compiling and simulating Verilog HDL code
- **GTKWave** – For waveform visualization

Step 1: Create Project Folder:

A new folder was created to organize the Verilog and testbench files:



```
engineer_taha@HP:~$ touch half_adder.v
engineer_taha@HP:~$ nano half_adder.v
engineer_taha@HP:~$ touch tb_half_adder.cpp
engineer_taha@HP:~$ nano tb_half_adder.cpp
```

Explanation: The **touch** command is used to create a new file, while **nano** is a text editor used to open and edit the contents of that file.

Verilog code of half_adder:

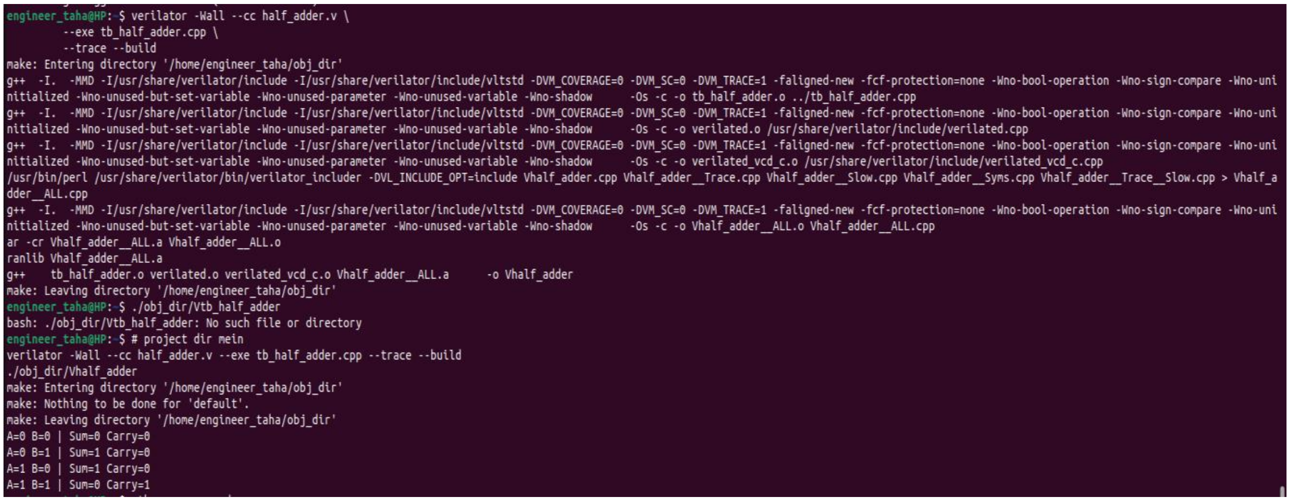
```
module half_adder(A, B, Carry, Sum);

  input A, B;

  output Carry, Sum;

  assign Carry = A & B;

  assign Sum = A ^ B;

endmodule
```

Testbench code of half_adder:

```
#include "Vhalf_adder.h"

#include "verilated.h"

#include "verilated_vcd_c.h"

int main(int argc, char** argv, char** env) {

   Verilated::commandArgs(argc, argv);

   Vhalf_adder* top = new Vhalf_adder;

   Verilated::traceEverOn(true);

   VerilatedVcdC* tfp = new VerilatedVcdC;

   top->trace(tfp, 99);

   tfp->open("wave.vcd");

   for (int A = 0; A <= 1; A++) {

      for (int B = 0; B <= 1; B++) {

         top->A = A;

         top->B = B;

         top->eval();

         tfp->dump(10 * (2 * A + B));

         printf("A=%d B=%d | Sum=%d Carry=%d\n", A, B, top>Sum, top->Carry);

      }

   }

   tfp->close();

   delete top;

   delete tfp;

   return 0;

}
```

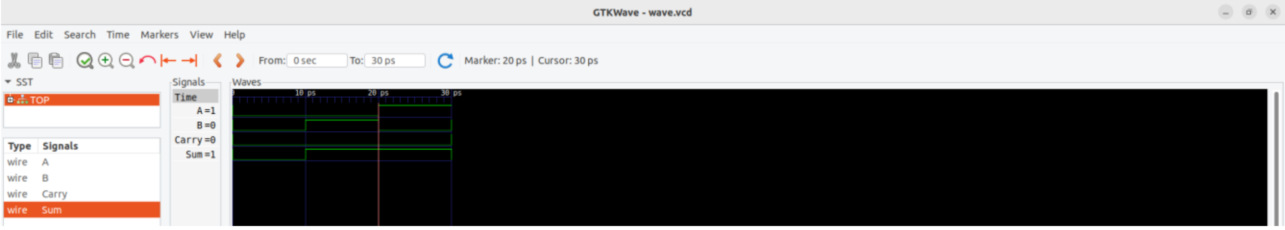
Step 2: Compilation and Simulation:

The design and testbench were compiled and simulated using the following Verilator command:



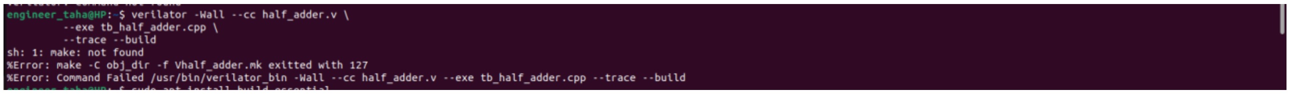
```
engineer_taha@HP:~$ verilator -Wall --cc half_adder.v \
        --exe tb_half_adder.cpp \
        --trace --build
make: Entering directory '/home/engineer_taha/obj_dir'
g++  -I.  -MMD -I/usr/share/verilator/include -I/usr/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0 -DVM_TRACE=1 -faligned-new -fcf-protection=none -Wno-bool-operation -Wno-sign-compare -Wno-uni
nitialized -Wno-unused-but-set-variable -Wno-unused-parameter -Wno-unused-variable -Wno-shadow     -Os -c -o tb_half_adder.o ../tb_half_adder.cpp
g++  -I.  -MMD -I/usr/share/verilator/include -I/usr/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0 -DVM_TRACE=1 -faligned-new -fcf-protection=none -Wno-bool-operation -Wno-sign-compare -Wno-uni
nitialized -Wno-unused-but-set-variable -Wno-unused-parameter -Wno-unused-variable -Wno-shadow     -Os -c -o verilated.o /usr/share/verilator/include/verilated.cpp
g++  -I.  -MMD -I/usr/share/verilator/include -I/usr/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0 -DVM_TRACE=1 -faligned-new -fcf-protection=none -Wno-bool-operation -Wno-sign-compare -Wno-uni
nitialized -Wno-unused-but-set-variable -Wno-unused-parameter -Wno-unused-variable -Wno-shadow     -Os -c -o verilated_vcd_c.o /usr/share/verilator/include/verilated_vcd_c.cpp
/usr/bin/perl /usr/share/verilator/bin/verilator_includer -DVL_INCLUDE_OPT=include Vhalf_adder.cpp Vhalf_adder__Trace.cpp Vhalf_adder__Slow.cpp Vhalf_adder__Syms.cpp Vhalf_adder__Trace__Slow.cpp > Vhalf_a
dder__ALL.cpp
g++  -I.  -MMD -I/usr/share/verilator/include -I/usr/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0 -DVM_TRACE=1 -faligned-new -fcf-protection=none -Wno-bool-operation -Wno-sign-compare -Wno-uni
nitialized -Wno-unused-but-set-variable -Wno-unused-parameter -Wno-unused-variable -Wno-shadow     -Os -c -o Vhalf_adder__ALL.o Vhalf_adder__ALL.cpp
ar -cr Vhalf_adder__ALL.a Vhalf_adder__ALL.o
ranlib Vhalf_adder__ALL.a
g++    tb_half_adder.o verilated.o verilated_vcd_c.o Vhalf_adder__ALL.a      -o Vhalf_adder
make: Leaving directory '/home/engineer_taha/obj_dir'
engineer_taha@HP:~$ ./obj_dir/Vtb_half_adder
bash: ./obj_dir/Vtb_half_adder: No such file or directory
engineer_taha@HP:~$ # project dir mein
verilator -Wall --cc half_adder.v --exe tb_half_adder.cpp --trace --build
./obj_dir/Vhalf_adder
make: Entering directory '/home/engineer_taha/obj_dir'
make: Nothing to be done for 'default'.
make: Leaving directory '/home/engineer_taha/obj_dir'
A=0 B=0 | Sum=0 Carry=0
A=0 B=1 | Sum=1 Carry=0
A=1 B=0 | Sum=1 Carry=0
A=1 B=1 | Sum=0 Carry=1
```

Step 3: Waveform in GTKWave:



Problem faced in Compiling the code:

While compiling the Verilog code using Verilator, the following error occurred:

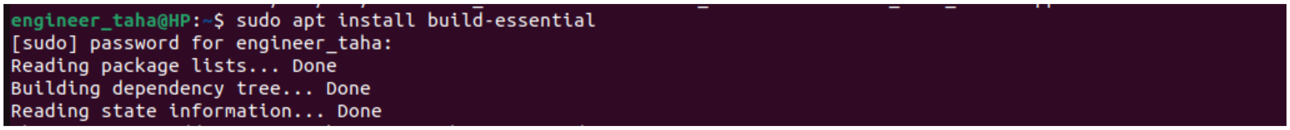


```
engineer_taha@HP:~$ verilator -Wall --cc half_adder.v \
        --exe tb_half_adder.cpp \
        --trace --build
sh: 1: make: not found
%Error: make -C obj_dir -f Vhalf_adder.mk exitted with 127
%Error: Command Failed /usr/bin/verilator_bin -Wall --cc half_adder.v --exe tb_half_adder.cpp --trace --build
```

Note: *This error indicated that the make utility, which is required to build the C++ simulation executable generated by Verilator, was not installed on the system.*

Solution:

This error indicated that the make utility, which is required to build the C++ simulation executable generated by Verilator, was not installed on the system.



```
engineer_taha@HP:~$ sudo apt install build-essential
[sudo] password for engineer_taha:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

This package includes essential development tools like **make**, **gcc**, and other utilities required for compiling C/C++ programs. After installing it, the Verilator build process completed successfully, and the simulation executable was generated without errors.

Conclusion:

Through this documentation, Verilator and GTKWave were successfully installed and used to simulate a Half Adder circuit on Ubuntu. The simulation verified the correct functionality of the Half Adder, and GTKWave provided a visual representation of the signal transitions.

This experience improved understanding of Verilog simulation, waveform analysis, and troubleshooting installation issues. It serves as a helpful reference for anyone setting up a similar simulation workflow for the first time.