Verilator Guide

Minsu Gong

gongms@postech.ac.kr

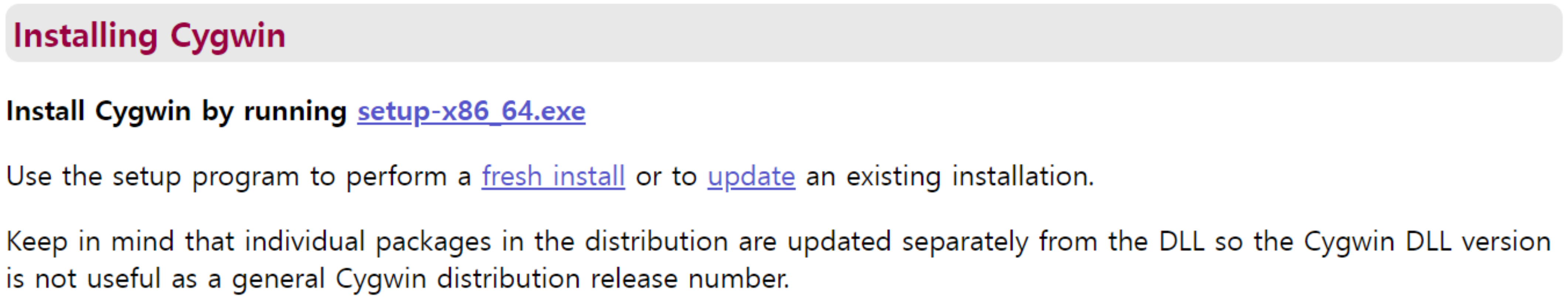
Installation Guide

What is Verilator

- Verilog/SystemVerilog simulator
- Compiles into multithreaded C++, or SystemC
- Widely used in industry and academy
- Developed on Linux & Mac OS
  - To use on Windows, you need WSL, Cygwin, or MinGW
- For this course, we use Windows(Cygwin) + Verilator.
  - You can use other OS or Linux supports for Windows (like WSL, MinGW) if you want
  - Required storage for Cygwin + Verilator is about 4 GB

0. Cygwin install

- For Windows only
  - Skip this if you are using Linux, MacOS, WSL, MinGW …
- Link : https://www.cygwin.com/
- Download setup file and execute



Installing Cygwin

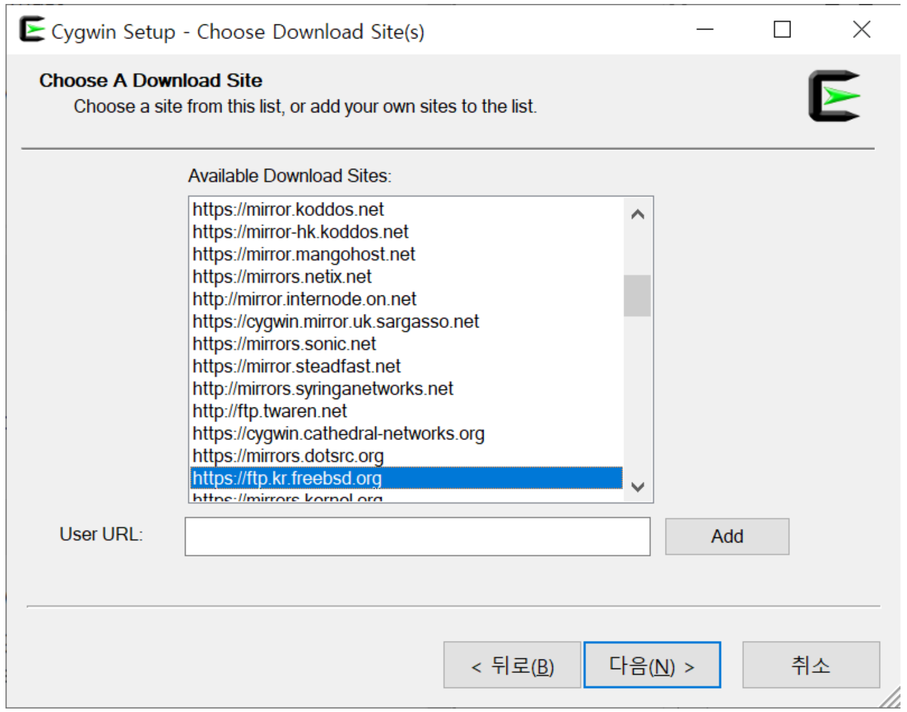
Install Cygwin by running setup-x86_64.exe

Use the setup program to perform a fresh install or to update an existing installation.

Keep in mind that individual packages in the distribution are updated separately from the DLL so the Cygwin DLL version is not useful as a general Cygwin distribution release number.

0. Cygwin install

- "Next" until you see this page
- Select https://ftp.kr.freebsd.org, which is located in Korea.
  - You can select anything else


Cygwin Setup - Choose Download Site(s)

Choose A Download Site

Choose a site from this list, or add your own sites to the list.

Available Download Sites:

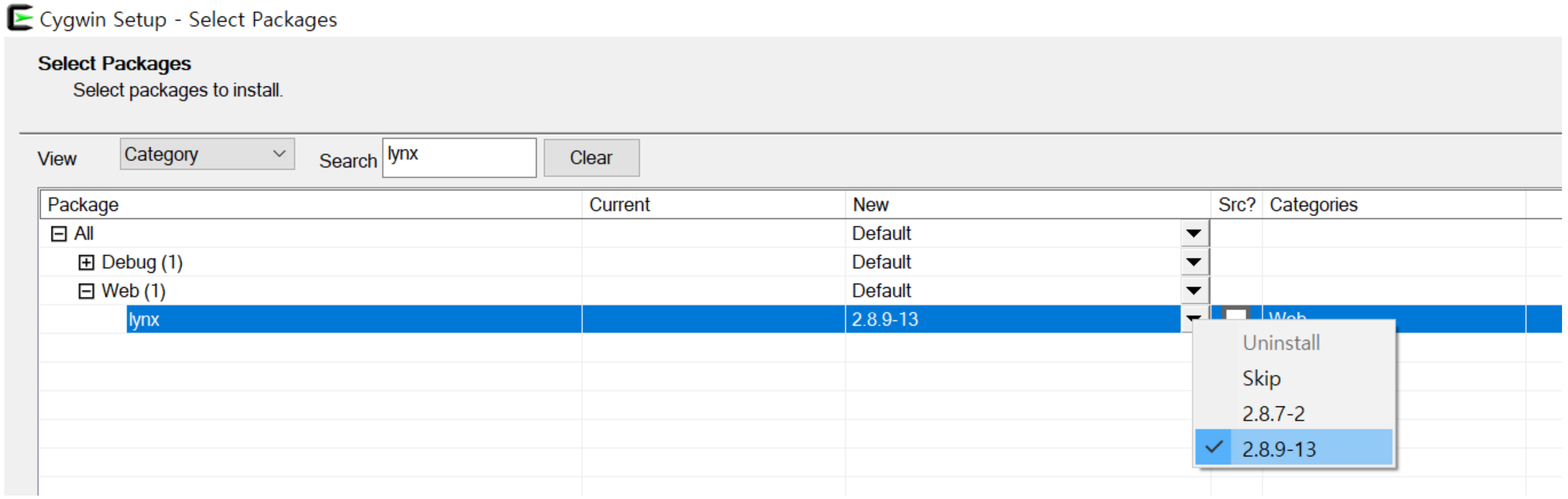
https://mirror.koddos.net
https://mirror-hk.koddos.net
https://mirror.mangohost.net
https://mirrors.netix.net
http://mirror.internode.on.net
https://cygwin.mirror.uk.sargasso.net
https://mirrors.sonic.net
https://mirror.steadfast.net
http://mirrors.syringanetworks.net
http://ftp.twaren.net
https://cygwin.cathedral-networks.org
https://mirrors.dotsrc.org
https://ftp.kr.freebsd.org

User URL:

POSTECH

0. Cygwin install

- Search “lynx” and select latest version
- Keep select “Next”



Cygwin Setup - Select Packages

Select Packages

Select packages to install.

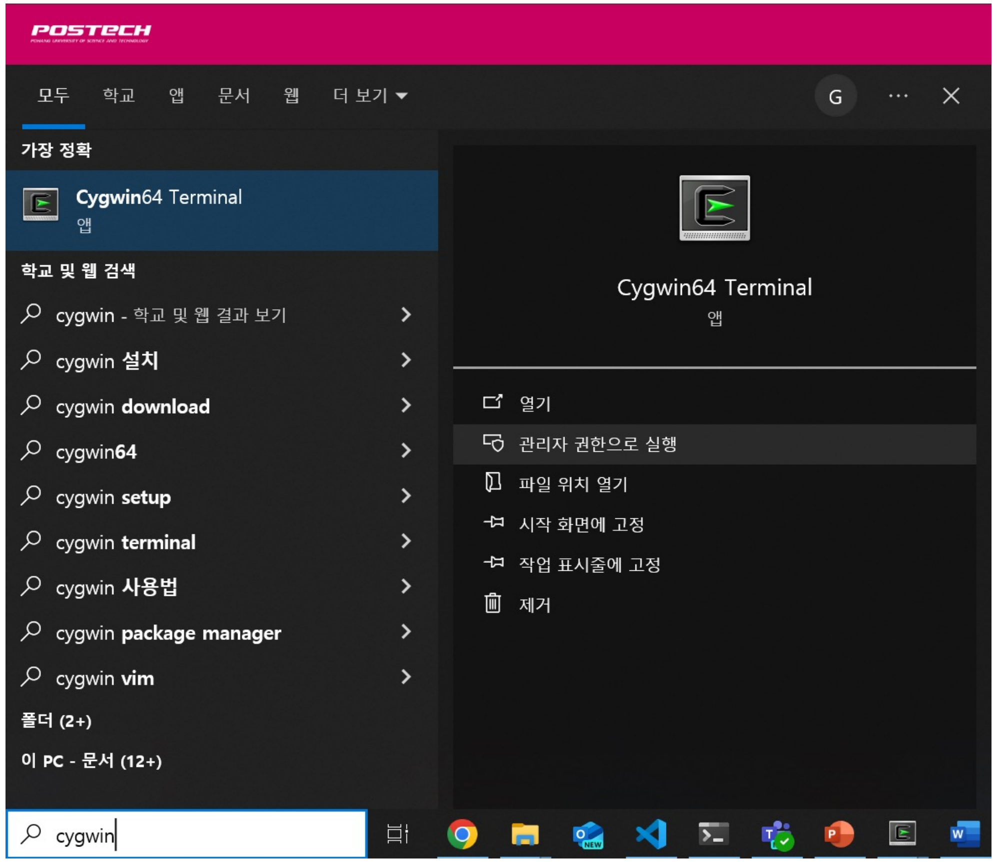
View: Category | Search: lynx | Clear

Package	Current	New	Src?	Categories
⊟ All		Default		
⊞ Debug (1)		Default		
⊟ Web (1)		Default		
lynx		2.8.9-13		Web

POSTECH

0. Cygwin install

- Run Cygwin as administrator



0. Cygwin install

- Run below

```
lynx -source rawgit.com/transcode-open/apt-cyg/master/apt-cyg > apt-cyg
install apt-cyg /bin
```

1. Verilator install

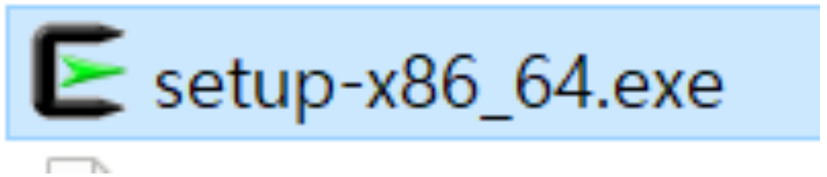
- This step follows https://verilator.org/guide/latest/install.html
- If you use Windows with Cygwin, follow this ppt.
  - We will use “apt-cyg” instead of “apt-get” to install packages.
- Or else, please check the link above.

1. Verilator install

- Run below to install prerequisites packages
  - Unlike the manual, install “gcc-g++” instead of “g++”

```
apt-cyg install git help2man perl python3 make autoconf gcc-g++ flex bison ccache
```

- Close Cygwin console and execute Cygwin setup file again.



setup-x86_64.exe

- Keep select “Next”, then new packages will be installed.

1. Verilator install

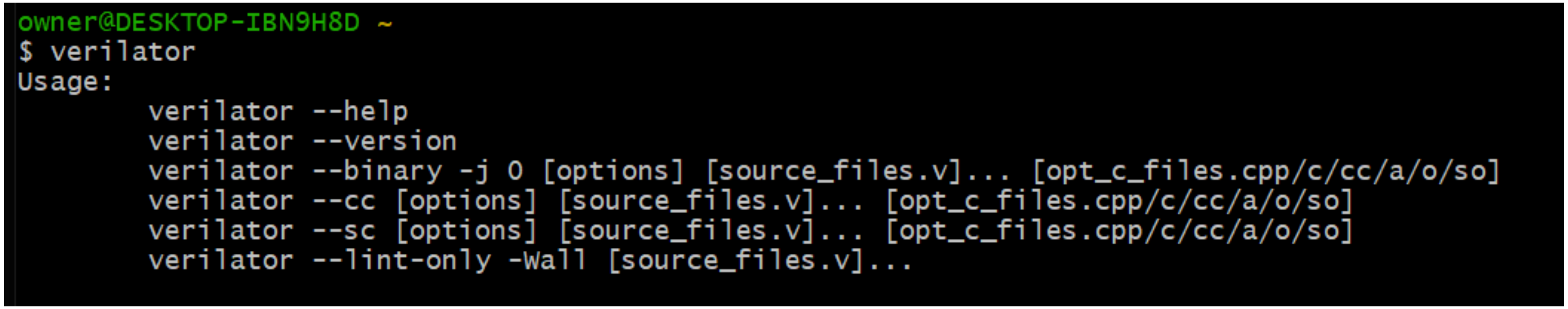
- Start Cygwin console and run below

```
git clone https://github.com/verilator/verilator
cd verilator
autoconf
./configure
make -j `nproc`
make install
cd ..
```

1. Verilator install

- Run below to check if verilator is installed well.

```
verilator
```



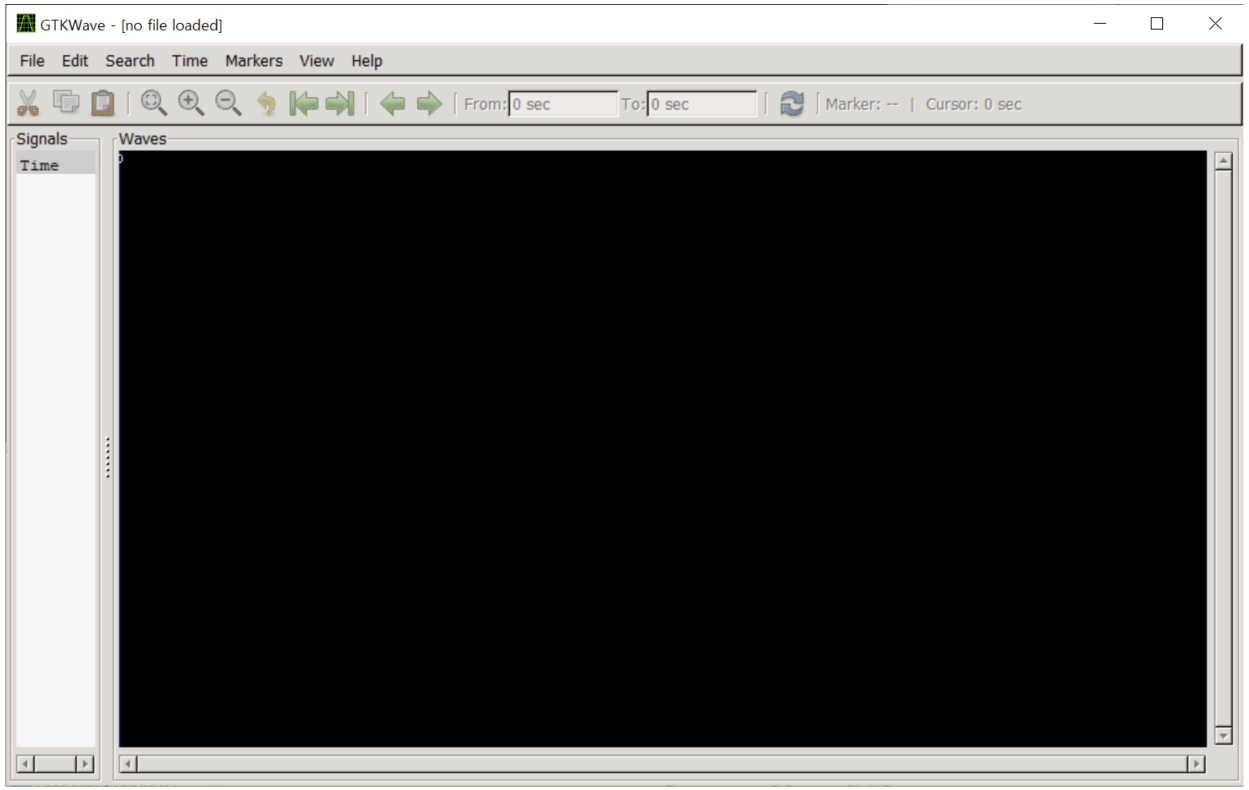
```
owner@DESKTOP-IBN9H8D ~
$ verilator
Usage:
        verilator --help
        verilator --version
        verilator --binary -j 0 [options] [source_files.v]... [opt_c_files.cpp/c/cc/a/o/so]
        verilator --cc [options] [source_files.v]... [opt_c_files.cpp/c/cc/a/o/so]
        verilator --sc [options] [source_files.v]... [opt_c_files.cpp/c/cc/a/o/so]
        verilator --lint-only -Wall [source_files.v]...
```

2. GTKWave install

- GTKWave is waveform viewer for Linux, MacOS and Windows
- For Windows, download https://sourceforge.net/projects/gtkwave/files/gtkwave-3.3.100-bin-win64/
- Or else, check https://gtkwave.sourceforge.net/

2. GTKWave install

- Under “bin” folder, run “gtkwave.exe”



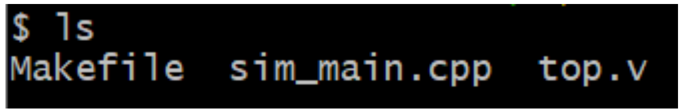
2. GTKWave install (macOS Sonoma)

- Currently, Gtkwave has an issue on macOS Sonnoma
- Run below commands in terminal
  - brew uninstall gtkwave
  - brew untap randomplum/gtkwave
  - brew install --HEAD randomplum/gtkwave/gtkwave
- Run “gtkwave” on terminal to run Gtkwave
- If “gtkwave” command not found, find all gtkwave files under /opt/Homebrew path and delete and rerun above commands
  - Find command : find /opt/Homebrew –name gtkwave
- Reference : https://github.com/gtkwave/gtkwave/issues/250

Verilator Simulation Guide

Simulation Directory

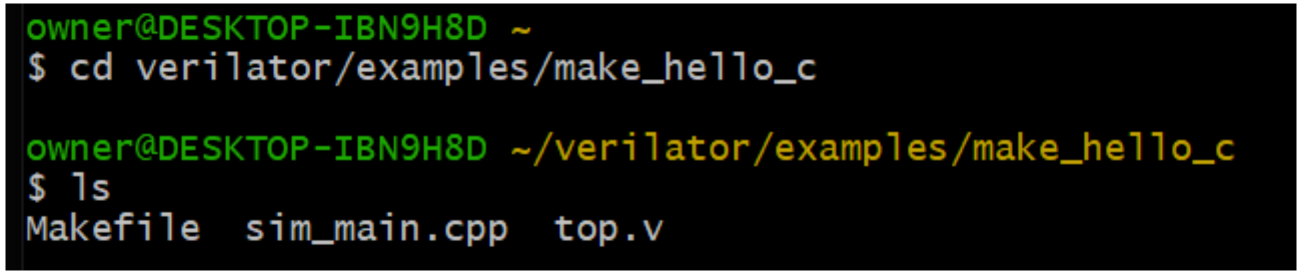
- .v : Verilog code
- sim_main.cpp : test bench code
- Makefile : build executable binary file



```
$ ls
Makefile  sim_main.cpp  top.v
```

Example : Hello World

- Go to "{workspace}/verilator/examples/make_hello_c"

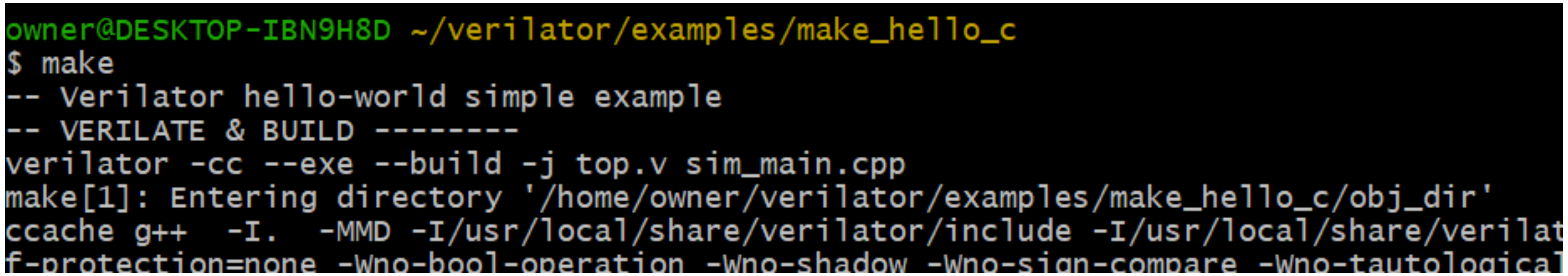


```
owner@DESKTOP-IBN9H8D ~
$ cd verilator/examples/make_hello_c

owner@DESKTOP-IBN9H8D ~/verilator/examples/make_hello_c
$ ls
Makefile  sim_main.cpp  top.v
```

- Open and check each codes
  - Use your favorite code editor(VSCode, Notepad…)
  - Cygwin users can locate {workspace} folder via "C:\cygwin64\home\{UserName}\"

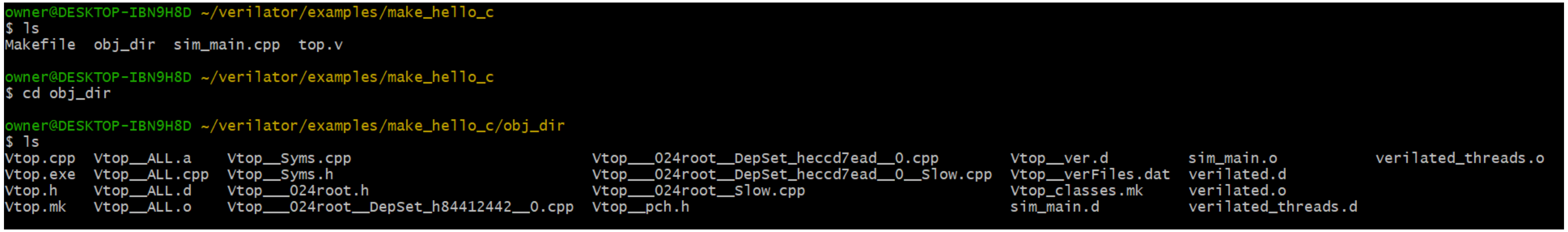
Example : Hello World

- Build by “make”



```
owner@DESKTOP-IBN9H8D ~/verilator/examples/make_hello_c
$ make
-- Verilator hello-world simple example
-- VERILATE & BUILD --------
verilator -cc --exe --build -j top.v sim_main.cpp
make[1]: Entering directory '/home/owner/verilator/examples/make_hello_c/obj_dir'
ccache g++  -I.  -MMD -I/usr/local/share/verilator/include -I/usr/local/share/verilat
f-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-compare -Wno-tautological
```

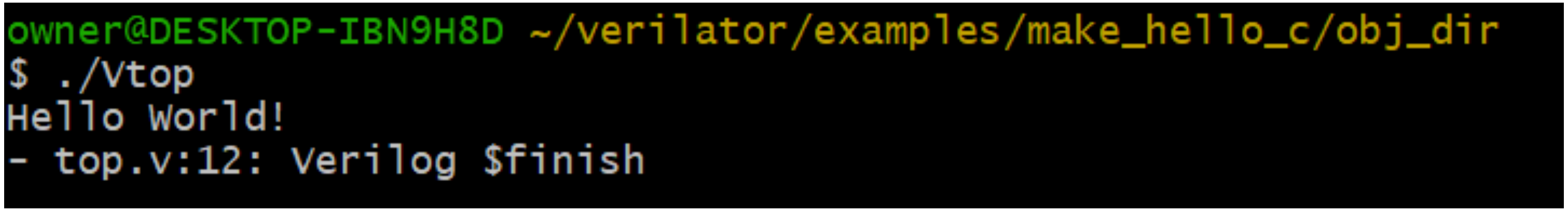
- Go to “obj_dir”



```
owner@DESKTOP-IBN9H8D ~/verilator/examples/make_hello_c
$ ls
Makefile  obj_dir  sim_main.cpp  top.v

owner@DESKTOP-IBN9H8D ~/verilator/examples/make_hello_c
$ cd obj_dir

owner@DESKTOP-IBN9H8D ~/verilator/examples/make_hello_c/obj_dir
$ ls
Vtop.cpp  Vtop__ALL.a    Vtop__Syms.cpp                           Vtop___024root__DepSet_heccd7ead__0.cpp        Vtop__ver.d         sim_main.o           verilated_threads.o
Vtop.exe  Vtop__ALL.cpp  Vtop__Syms.h                             Vtop___024root__DepSet_heccd7ead__0__Slow.cpp  Vtop__verFiles.dat  verilated.d
Vtop.h    Vtop__ALL.d    Vtop___024root.h                         Vtop___024root__Slow.cpp                       Vtop_classes.mk     verilated.o
Vtop.mk   Vtop__ALL.o    Vtop___024root__DepSet_h84412442__0.cpp  Vtop__pch.h                                    sim_main.d          verilated_threads.d
```

Example : Hello World

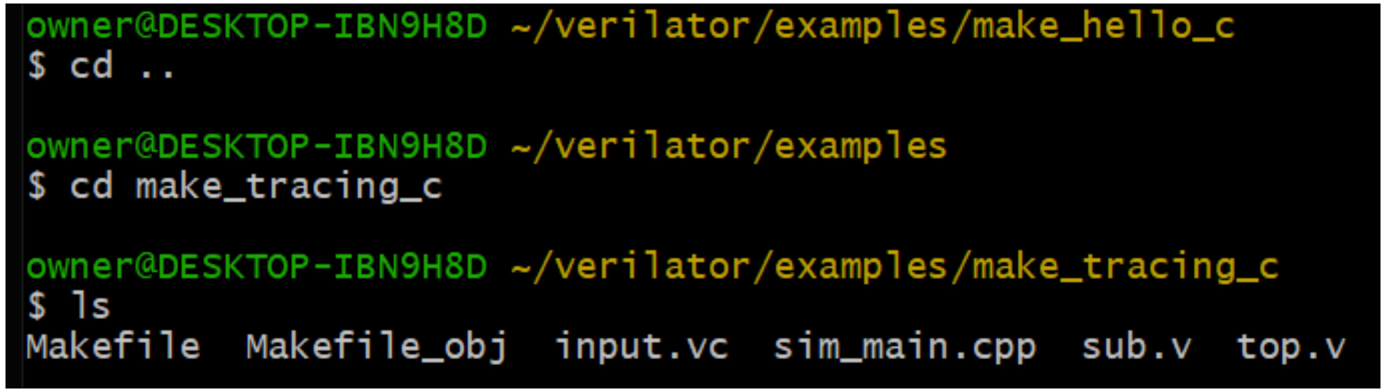
- Run Vtop



```
owner@DESKTOP-IBN9H8D ~/verilator/examples/make_hello_c/obj_dir
$ ./Vtop
Hello World!
- top.v:12: Verilog $finish
```

Example : Counter

- Go to “{workspace}/verilator/examples/make_tracing_c”

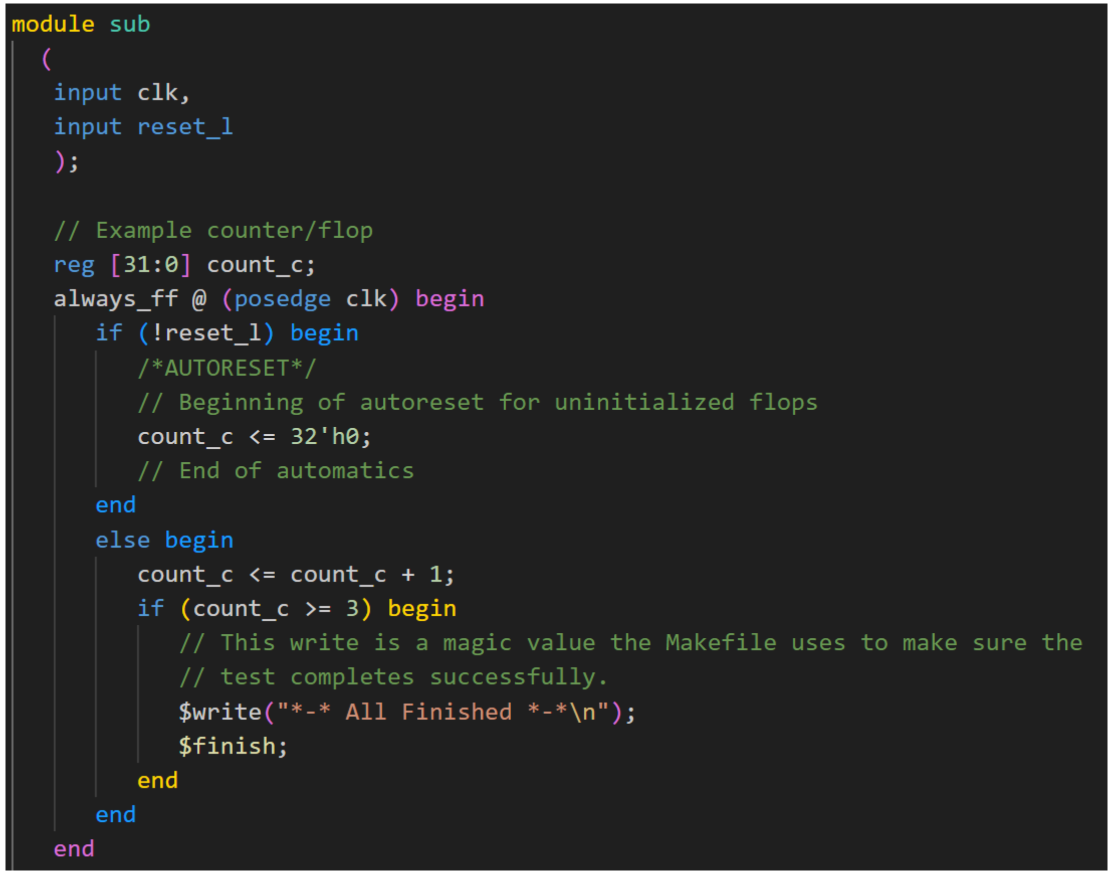


```
owner@DESKTOP-IBN9H8D ~/verilator/examples/make_hello_c
$ cd ..

owner@DESKTOP-IBN9H8D ~/verilator/examples
$ cd make_tracing_c

owner@DESKTOP-IBN9H8D ~/verilator/examples/make_tracing_c
$ ls
Makefile  Makefile_obj  input.vc  sim_main.cpp  sub.v  top.v
```

- Again, check each codes
- Run “make”

Example : Counter

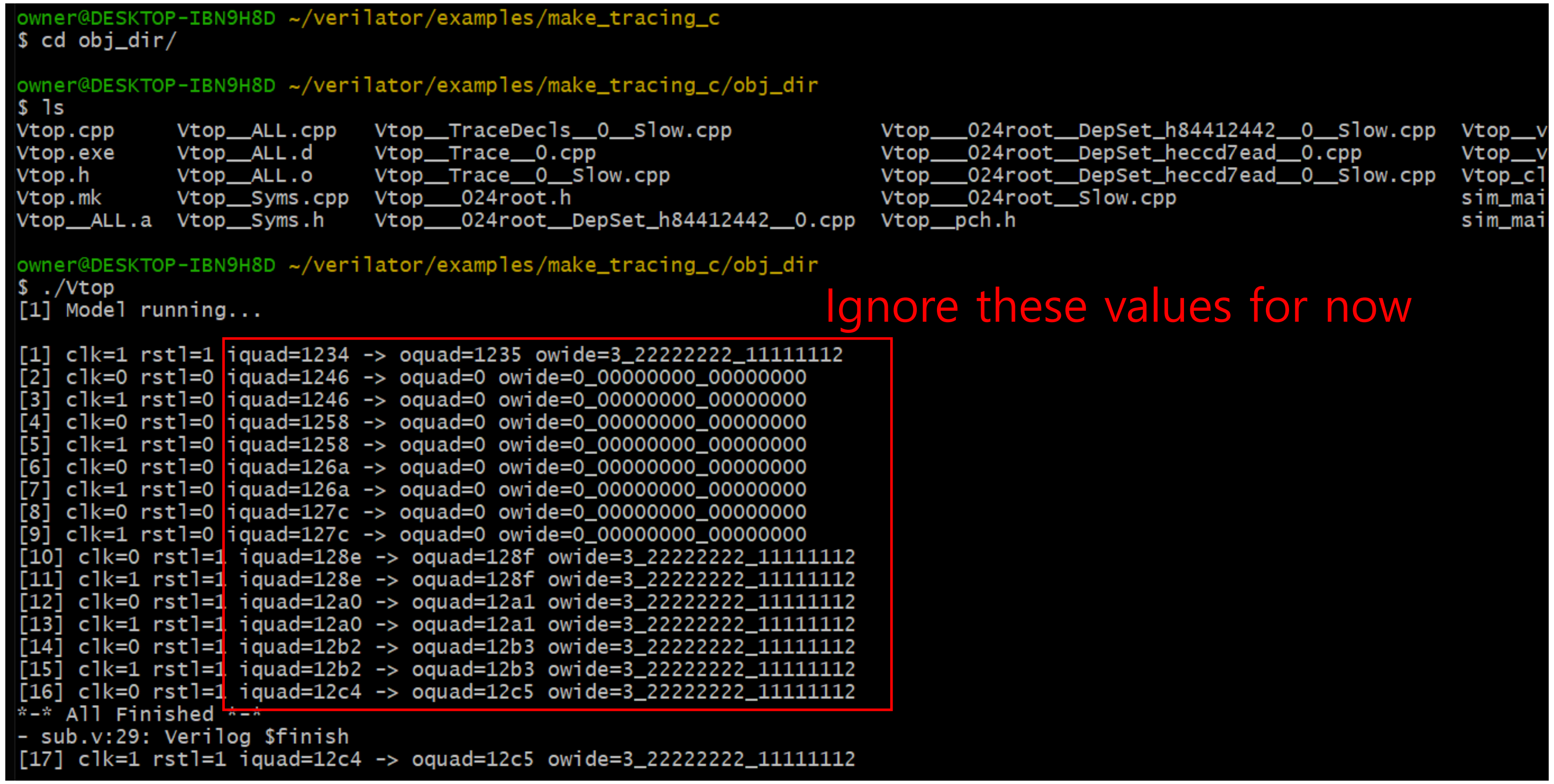
- Code implementing counter



```
module sub
 (
  input clk,
  input reset_l
  );

  // Example counter/flop
  reg [31:0] count_c;
  always_ff @ (posedge clk) begin
     if (!reset_l) begin
        /*AUTORESET*/
        // Beginning of autoreset for uninitialized flops
        count_c <= 32'h0;
        // End of automatics
     end
     else begin
        count_c <= count_c + 1;
        if (count_c >= 3) begin
           // This write is a magic value the Makefile uses to make sure the
           // test completes successfully.
           $write("*-* All Finished *-*\n");
           $finish;
        end
     end
  end
```

Example : Counter

- Again, run Vtop

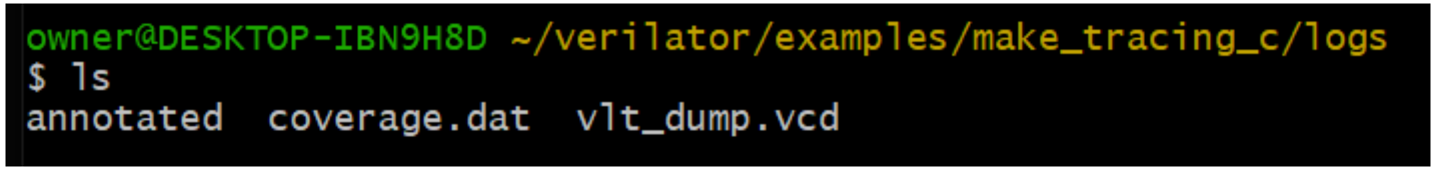


```
owner@DESKTOP-IBN9H8D ~/verilator/examples/make_tracing_c
$ cd obj_dir/

owner@DESKTOP-IBN9H8D ~/verilator/examples/make_tracing_c/obj_dir
$ ls
Vtop.cpp      Vtop__ALL.cpp   Vtop__TraceDecls__0__Slow.cpp               Vtop___024root__DepSet_h84412442__0__Slow.cpp  Vtop__v
Vtop.exe      Vtop__ALL.d     Vtop__Trace__0.cpp                          Vtop___024root__DepSet_heccd7ead__0.cpp        Vtop__v
Vtop.h        Vtop__ALL.o     Vtop__Trace__0__Slow.cpp                    Vtop___024root__DepSet_heccd7ead__0__Slow.cpp  Vtop_cl
Vtop.mk       Vtop__Syms.cpp  Vtop___024root.h                            Vtop___024root__Slow.cpp                       sim_mai
Vtop__ALL.a   Vtop__Syms.h    Vtop___024root__DepSet_h84412442__0.cpp     Vtop__pch.h                                    sim_mai

owner@DESKTOP-IBN9H8D ~/verilator/examples/make_tracing_c/obj_dir
$ ./Vtop
[1] Model running...

[1] clk=1 rstl=1 iquad=1234 -> oquad=1235 owide=3_22222222_11111112
[2] clk=0 rstl=0 iquad=1246 -> oquad=0 owide=0_00000000_00000000
[3] clk=1 rstl=0 iquad=1246 -> oquad=0 owide=0_00000000_00000000
[4] clk=0 rstl=0 iquad=1258 -> oquad=0 owide=0_00000000_00000000
[5] clk=1 rstl=0 iquad=1258 -> oquad=0 owide=0_00000000_00000000
[6] clk=0 rstl=0 iquad=126a -> oquad=0 owide=0_00000000_00000000
[7] clk=1 rstl=0 iquad=126a -> oquad=0 owide=0_00000000_00000000
[8] clk=0 rstl=0 iquad=127c -> oquad=0 owide=0_00000000_00000000
[9] clk=1 rstl=0 iquad=127c -> oquad=0 owide=0_00000000_00000000
[10] clk=0 rstl=1 iquad=128e -> oquad=128f owide=3_22222222_11111112
[11] clk=1 rstl=1 iquad=128e -> oquad=128f owide=3_22222222_11111112
[12] clk=0 rstl=1 iquad=12a0 -> oquad=12a1 owide=3_22222222_11111112
[13] clk=1 rstl=1 iquad=12a0 -> oquad=12a1 owide=3_22222222_11111112
[14] clk=0 rstl=1 iquad=12b2 -> oquad=12b3 owide=3_22222222_11111112
[15] clk=1 rstl=1 iquad=12b2 -> oquad=12b3 owide=3_22222222_11111112
[16] clk=0 rstl=1 iquad=12c4 -> oquad=12c5 owide=3_22222222_11111112
*-* All Finished *-*
- sub.v:29: Verilog $finish
[17] clk=1 rstl=1 iquad=12c4 -> oquad=12c5 owide=3_22222222_11111112
```

Ignore these values for now

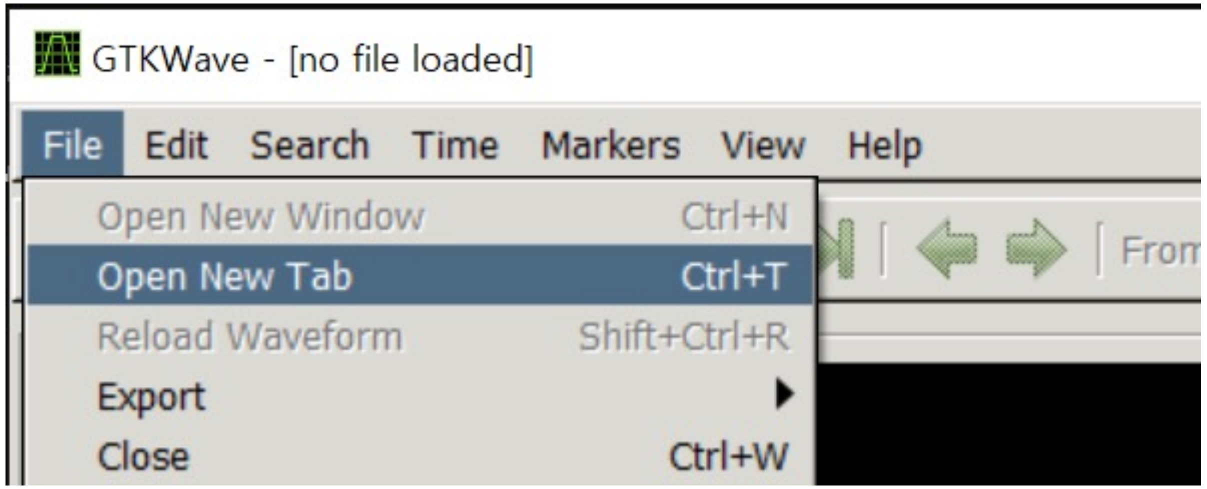
Example : Counter

- Go to “make_tracing_c/logs”



```
owner@DESKTOP-IBN9H8D ~/verilator/examples/make_tracing_c/logs
$ ls
annotated  coverage.dat  vlt_dump.vcd
```

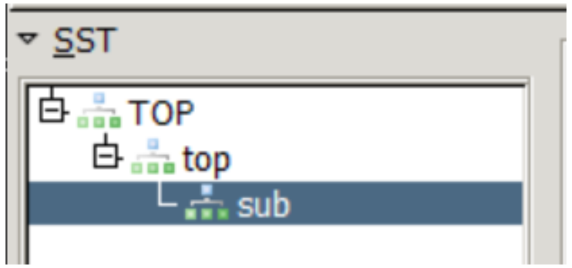
- Open “vlt_dump.vcd” with GTKWave



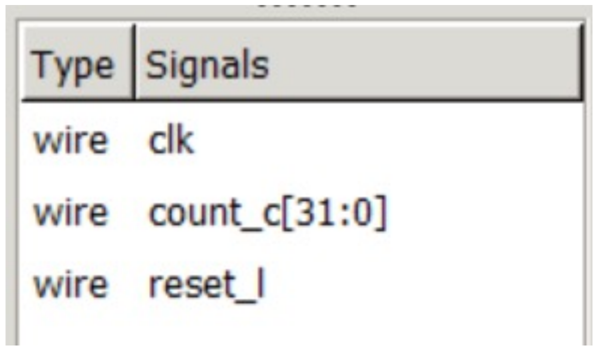
GTKWave - [no file loaded]

Example : Counter

- Open sub module



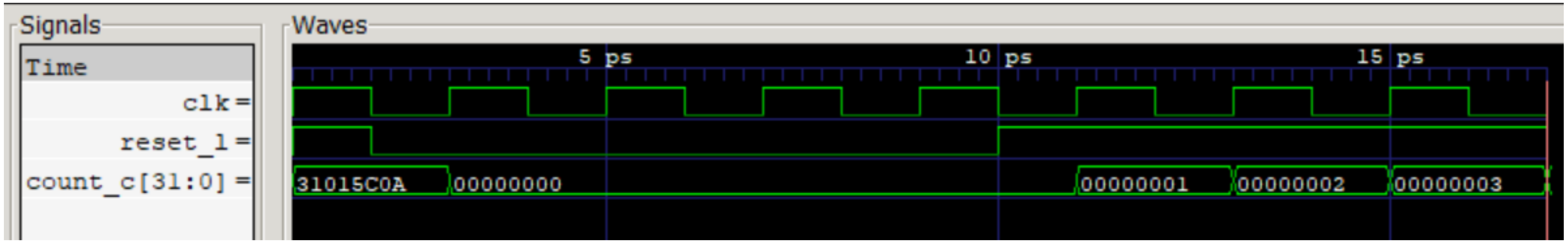
- Add clk, reset_l, count_c by double clicking



Type	Signals
wire	clk
wire	count_c[31:0]
wire	reset_l

Example : Counter

- Now you can see the result



Debug

- For debugging, you have several options
  - gdb
    - https://www.sourceware.org/gdb/
    - https://verilator.org/guide/latest/simulating.html?highlight=debug
    - https://verilator.org/guide/latest/exe_verilator.html?highlight=debug#cmdoption-debug
    - For VSCode users, https://code.visualstudio.com/docs/cpp/cpp-debug
  - printf
  - Waveform

Thank you

- Any questions?