# Verilator Basic Usage Guide

Author: Ryan Cramer

## 1. What is Verilator?

Verilator converts Verilog and SystemVerilog HDL designs into C++ or SystemC, which is then compiled and executed. Verilator is more of a compiler than a simulator.

## 2. Typical Usage:

Verilator can be used in 5 different ways:

1. **Compilation Modes**
    1. **--binary** option, which Verilator makes your design executable by generating C++ and compiling it
    2. **-cc** or **-sc** options, which Verilator will translate the design into C++ (cc) or SystemC (sc)
    3. **--lint-only**, which Verilator will lint the design to check for warnings (no output files)
2. **Utility Modes**
    4. **--json-only**, which Verilator creates a JSON output to feed to other design tools
    5. **-E**, which Verilator preprocesses the code according to IEEE preprocessing rules and write the output to stdout. This is useful to feed other tools and debug `define statements.

## 3. Generating Models

--binary just makes a C++ model and compiles it

--cc = C++ output mode, Verilator generates a simple C++ class for the model

- User must write C++ wrapper and main loop for simulation
- Call eval() to evaluate the model
- Call final() when simulation is complete

--sc will turn SystemVerilog into SystemC

- User must write SystemC wrapper and main loop for simulation

--top-module determines the top module (otherwise MULTITOP errors can occur)

Verilator writes C++/SystemC code to output files into the --Mdir option-specified directory, or defaults to /obj_dir/. The prefix is set with --prefix, which defaults to the name of the top module

If --binary or --main is used, Verilator creates a C++ top wrapper to read CLI arguments, creates the model, then executes the model.

If --binary or --exe is used, Verilator creates makefiles to generate a simulation executable, otherwise, it creates makefiles to generate an archive (.a) containing the objects

If --binary or --build is used, it calls GNU Make or CMake to build the model
(once a model is built, the next step is to run it)

# Please see Cycle-Driven vs Event-Driven Simulation

| Mode | How it Works | Pros | Cons |
|---|---|---|---|
| CYCLE-Driven | Evaluate whole design per clock tick in C++ | Extremely fast, deterministic | Ignores #delays, not valid for gate-level timing. |
| EVENT-Driven | Scheduler processes events, delta cycles | Supports precise timing, 2-state simulation | Slower, heavier<br>Made for Verilog TBs |

You can create a wrapper that will call the Verilated model of your design, and essentially act as a testbench.

Example: Using a 32-bit comparator, here is an example of a combinational testbench wrapper:

verilator_wrappers > sim_main.cpp

```cpp
#include "Vcmp_32.h"
#include "verilated.h"
#include <iostream>


int main(int argc, char** argv) {
    // Create a simulation context
    VerilatedContext* contextp = new VerilatedContext;
    contextp->commandArgs(argc, argv);

    // Instantiate DUT inside this context
    Vcmp_32* top = new Vcmp_32{contextp};

    if (argc < 3) {
        std::cerr << "Usage: " << argv[0] << " <A> <B>\n";
        delete top;
        delete contextp;
        return 1;
    }

    // Parse inputs (decimal or hex)
    uint32_t A = std::strtol(argv[1], nullptr, 0);
    uint32_t B = std::strtol(argv[2], nullptr, 0);

    // Drive DUT
    top->A = A;
    top->B = B;
    top->eval();

    // Print results
    std::cout << "A = " << A << ", B = " << B << "\n";
    std::cout << "GT = " << (top->GT ? 1 : 0) << "\n";
    std::cout << "LT = " << (top->LT ? 1 : 0) << "\n";
    std::cout << "EQ = " << (top->EQ ? 1 : 0) << "\n";

    delete top;
    delete contextp;
    return 0;
}
```

## SystemVerilog to Executable (no-wave trace):

1. Create SystemVerilog File <design>.sv

```
(base) ryanc@rcramer:~/brogramming/NVHDL$ ls
demo.sv
```

```
module demo;
     initial begin $display("Hello World"); $finish; end
endmodule
```

2. Run Verilator on the example (this creates obj_dir)

   `ubuntu@asic$ verilator --binary -j 0 -Wall <design>.sv`

```
(base) ryanc@rcramer:~/brogramming/NVHDL$ verilator --binary -j 0 -Wall demo.sv
make: Entering directory '/home/ryanc/brogramming/NVHDL/obj_dir'
g++ -Os  -I.  -MMD -I/usr/share/verilator/include -I/usr/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0 -DVM_T
RACE=0 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=0 -faligned-new -fcf-protection=none -Wno-bool-operation -Wno-overloaded-virtual
-Wno-shadow -Wno-sign-compare -Wno-uninitialized -Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-
parameter -Wno-unused-variable     -DVL_TIME_CONTEXT   -c -o verilated.o /usr/share/verilator/include/verilated.cpp
g++ -Os  -I.  -MMD -I/usr/share/verilator/include -I/usr/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0 -DVM_T
RACE=0 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=0 -faligned-new -fcf-protection=none -Wno-bool-operation -Wno-overloaded-virtual
-Wno-shadow -Wno-sign-compare -Wno-uninitialized -Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-
parameter -Wno-unused-variable     -DVL_TIME_CONTEXT   -c -o verilated_threads.o /usr/share/verilator/include/verilated_
threads.cpp
/usr/bin/python3 /usr/share/verilator/bin/verilator_includer -DVL_INCLUDE_OPT=include Vdemo.cpp Vdemo___024root__DepSet_
h3df3bbc7__0.cpp Vdemo___024root__DepSet_hd2c5ae8f__0.cpp Vdemo__main.cpp Vdemo___024root__Slow.cpp Vdemo___024root__Dep
Set_hd2c5ae8f__0__Slow.cpp Vdemo__Syms.cpp > Vdemo__ALL.cpp
echo "" > Vdemo__ALL.verilator_deplist.tmp
g++ -Os  -I.  -MMD -I/usr/share/verilator/include -I/usr/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0 -DVM_T
RACE=0 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=0 -faligned-new -fcf-protection=none -Wno-bool-operation -Wno-overloaded-virtual
-Wno-shadow -Wno-sign-compare -Wno-uninitialized -Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-
parameter -Wno-unused-variable     -DVL_TIME_CONTEXT   -c -o Vdemo__ALL.o Vdemo__ALL.cpp
Archive ar -rcs Vdemo__ALL.a Vdemo__ALL.o
g++     verilated.o verilated_threads.o Vdemo__ALL.a    -pthread -lpthread -latomic   -o Vdemo
rm Vdemo__ALL.verilator_deplist.tmp
make: Leaving directory '/home/ryanc/brogramming/NVHDL/obj_dir'
(base) ryanc@rcramer:~/brogramming/NVHDL$ ls
demo.sv  obj_dir
```

| | |
|---|---|
| **--binary** | Verilator will translate the design into an executable |
| **-j 0** | simulate using as many threads as possible |
| **-Wall** | stronger lint warnings |
| **demo.sv** | our example SystemVerilog file |

3. Run your Verilated model (your model inside obj_dir - has prefix V)

   `ubuntu@asic$ ./obj_dir/V<design>`

```
(base) ryanc@rcramer:~/brogramming/NVHDL$ ./obj_dir/Vdemo
Hello World
- demo.sv:2: Verilog $finish
```

# SystemVerilog to Waveform

Note: Please be sure to use **$dumpfile()** and **$dumpvars()** in your testbench to generate a .vcd

```
initial begin
    $dumpfile("tb_cmp_32.vcd");
    $dumpvars(0, tb_cmp_32);
end
```

verilator --timing --trace --binary <testname>.sv -I"<rtl folder>"

./obj_dir/V<testname>

```
ubuntu@asic:~/workspace/cmp_32$ verilator --timing --trace --binary tests/tb_cmp_32/tb_cmp_32.sv -I"rtl"
make: Entering directory '/home/asic/workspace/cmp_32/obj_dir'
ccache g++ -Os  -I.  -MMD -I/foss/tools/verilator/share/verilator/include -I/foss/tools/verilator/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0 -DVM_TIMING=1
 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-compare -Wno-subobject-linkage -Wno-tautolo
gical-compare -Wno-uninitialized -Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter -Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutin
es -c -o verilated.o /foss/tools/verilator/share/verilator/include/verilated.cpp
ccache g++ -Os  -I.  -MMD -I/foss/tools/verilator/share/verilator/include -I/foss/tools/verilator/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0 -DVM_TIMING=1
 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-compare -Wno-subobject-linkage -Wno-tautolo
gical-compare -Wno-uninitialized -Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter -Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutin
es -c -o verilated_vcd_c.o /foss/tools/verilator/share/verilator/include/verilated_vcd_c.cpp
ccache g++ -Os  -I.  -MMD -I/foss/tools/verilator/share/verilator/include -I/foss/tools/verilator/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0 -DVM_TIMING=1
 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-compare -Wno-subobject-linkage -Wno-tautolo
gical-compare -Wno-uninitialized -Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter -Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutin
es -c -o verilated_timing.o /foss/tools/verilator/share/verilator/include/verilated_timing.cpp
ccache g++ -Os  -I.  -MMD -I/foss/tools/verilator/share/verilator/include -I/foss/tools/verilator/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0 -DVM_TIMING=1
 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-compare -Wno-subobject-linkage -Wno-tautolo
gical-compare -Wno-uninitialized -Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter -Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutin
es -c -o verilated_threads.o /foss/tools/verilator/share/verilator/include/verilated_threads.cpp
python3 /foss/tools/verilator/share/verilator/bin/verilator_includer -DVL_INCLUDE_OPT=include Vtb_cmp_32.cpp Vtb_cmp_32___024root__DepSet_h138cccfe__0.cpp Vtb_cmp_32___
024root__DepSet_h4b90a73b__0.cpp Vtb_cmp_32__main.cpp Vtb_cmp_32__Trace__0.cpp Vtb_cmp_32___024root__Slow.cpp Vtb_cmp_32___024root__DepSet_h138cccfe__0__Slow.cpp Vtb_cm
p_32___024root__DepSet_h4b90a73b__0__Slow.cpp Vtb_cmp_32___024unit__Slow.cpp Vtb_cmp_32___024unit__DepSet_h94f38ef6__0__Slow.cpp Vtb_cmp_32__Syms.cpp Vtb_cmp_32__Trace_
_0__Slow.cpp Vtb_cmp_32__TraceDecls__0__Slow.cpp > Vtb_cmp_32__ALL.cpp
ccache g++ -Os  -I.  -MMD -I/foss/tools/verilator/share/verilator/include -I/foss/tools/verilator/share/verilator/include/vltstd -DVM_COVERAGE=0 -DVM_SC=0 -DVM_TIMING=1
 -DVM_TRACE=1 -DVM_TRACE_FST=0 -DVM_TRACE_VCD=1 -faligned-new -fcf-protection=none -Wno-bool-operation -Wno-shadow -Wno-sign-compare -Wno-subobject-linkage -Wno-tautolo
gical-compare -Wno-uninitialized -Wno-unused-but-set-parameter -Wno-unused-but-set-variable -Wno-unused-parameter -Wno-unused-variable    -DVL_TIME_CONTEXT   -fcoroutin
es -c -o Vtb_cmp_32__ALL.o Vtb_cmp_32__ALL.cpp
echo "" > Vtb_cmp_32__ALL.verilator_deplist.tmp
g++ -fuse-ld=mold   verilated.o verilated_vcd_c.o verilated_timing.o verilated_threads.o Vtb_cmp_32__ALL.a    -pthread -lpthread -latomic   -o Vtb_cmp_32
rm Vtb_cmp_32__ALL.verilator_deplist.tmp
make: Leaving directory '/home/asic/workspace/cmp_32/obj_dir'
- V e r i l a t i o n   R e p o r t: Verilator 5.034 2025-02-24 rev v5.034
- Verilator: Built from 0.025 MB sources in 3 modules, into 0.045 MB in 13 C++ files needing 0.000 MB
- Verilator: Walltime 8.777 s (elab=0.000, cvt=0.005, bld=8.764); cpu 0.012 s on 1 threads; alloced 9.113 MB
ubuntu@asic:~/workspace/cmp_32$
```

Use Surfer or GTKWave to view the .vcd file

```
ubuntu@asic:~/workspace/cmp_32$ gtkwave tests/tb_cmp_32/tb_cmp_32.vcd

GTKWave Analyzer v3.4.0 (w)1999-2022 BSI

[0] start time.
[40000] end time.
```