Ho Chi Minh University of Technology
Department of Electronics

DIGITAL SYSTEM DESIGN AND VERIFICATION
(EE3213)

Basics of testbench and guide to using the Xcelium tool

Instructor: Hieu Nguyen

TA: Duy Huynh

October 7, 2025

Contents

1 Introduction

In this instruction, we learned about:

- Basic syntax for writing testbench
- Basic flow for a simple testbench
- Recommended testbench for RTL design
- Simulation tool manipulation

Specifically, we will see how to run a logic simulation to perform a direct test:

- How to drive stimulus to the RTL block by the testbench
- How to instantiate the DUT - Design Under Test (RTL block) in the testbench
- How to run the simulation and verify the design via waveform and logic checking

2 Direct test

2.1 Introduction to directed test

There are various verification strategies, including directed tests, random tests, and regression tests,... In this class, we focus on the most fundamental test style, the directed test.

In RTL verification, a directed test involves providing manual and explicit stimulus (inputs) to the DUT (RTL block) to verify its expected outputs. The directed test characteristics include:

- The manual input sequences
- Predictable and easy to check
- Low coverage as the testbench might miss corner cases - biased to the engineer's design mindset

The Figure 1 shows the flow of a basic testbench.

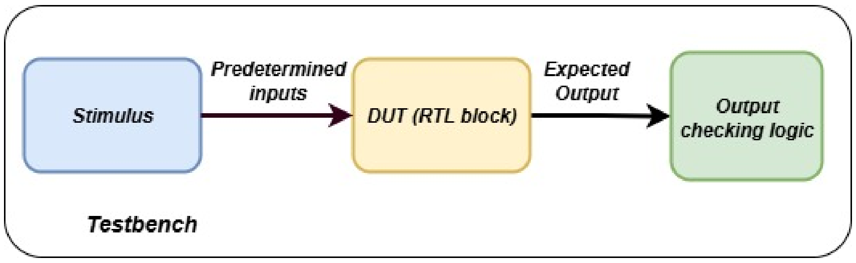


Figure 1: Basic testbench diagram

As for the inputs, the Stimulus component mainly includes:

- Clock and reset generator - In case the RTL design is sequential
- Input signal driver

As for the outputs, there are two main approaches to check their accuracy:

- Check on waveform
- Check by a logic check code written by System Verilog or Verilog

For example, the code below shows stimuli to a Full Adder design.

```
`timescale 1ns/1ps // <unit/precision>
module FullAdderTb();
//Stimulus inputs
  logic A_i;
  logic B_i;

//Expected outputs
  logic S_o;
  logic C_o;

  FullAdder DUT(
     .A_i  (A_i),
     .B_i  (B_i),
     .S_o  (S_o),
     .C_o  (C_o)
  );

/*#############################
 Stimulus generator
```

```
##############################*/
  initial begin
    //Initiate A_i and B_i
    A_i = 0;
    B_i = 0;
    #10; // Wait 10ns - 1 cycle
    // B_i change to 1, A_i stable
    A_i = 0;
    B_i = 1;
    #10; // Wait 10ns - 1 cycle
    // A_i change to 1, B_i change to 0
    A_i = 1;
    B_i = 0;
    #20; // Wait 20ns - 2 cycles
    // A_i and B_i both equal to 1
    A_i = 1;
    B_i = 1;
  end
endmodule
```

The input waveform of the stimulus generator above is shown in Figure 2:

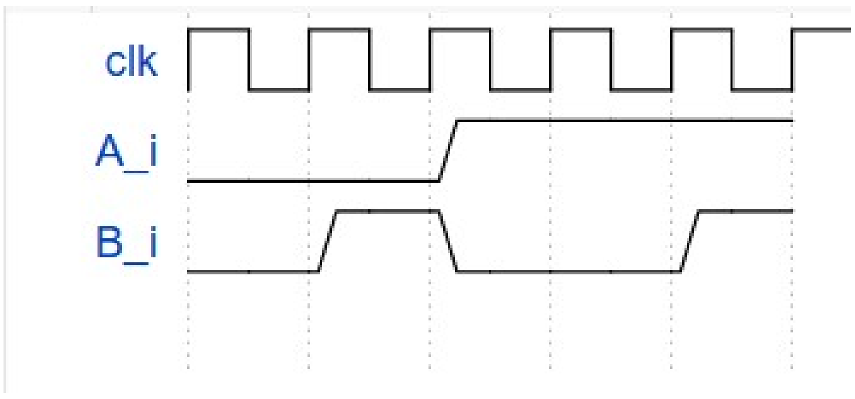


Figure 2: Stimulus input waveform

Remember:

- The testbench module has no inputs and outputs, just internal signals or drivers.
- The stimuli are driven by the initial block after a specific delay of times.

After generating stimuli, you can insert checking logic directly into the initial block to test the output to see if it satisfies the expected output. The code below demonstrates how to directly verify a Full Adder:

```
initial begin
    //Initiate A_i and B_i
    A_i = 0;
    B_i = 0;
    if({C_o,S_o} == A_i + B_i) $display(" Correct calculation");
    else $display(" Wrong calculation at %0d + %0d", A_i, B_i);
    #10; // Wait 10ns - 1 cycle
    // B_i change to 1, A_i stable
    A_i = 0;
    B_i = 1;
    if({C_o,S_o} == A_i + B_i) $display(" Correct calculation");
    else $display(" Wrong calculation at %0d + %0d", A_i, B_i);
    #10; // Wait 10ns - 1 cycle
    // A_i change to 1, B_i change to 0
    A_i = 1;
    B_i = 0;
    if({C_o,S_o} == A_i + B_i) $display(" Correct calculation");
    else $display(" Wrong calculation at %0d + %0d", A_i, B_i);
    #20; // Wait 20ns - 2 cycles
    // A_i and B_i both equal to 1
    A_i = 1;
    B_i = 1;
    if({C_o,S_o} == A_i + B_i) $display(" Correct calculation");
    else $display(" Wrong calculation at %0d + %0d", A_i, B_i);
  end
```

Explanation:

- $\{C_o, S_o\}$: bit concatenation that forms a 2-bit binary number (C_o is the high bit, S_o is the low bit).
- $A_i + B_i$: the sum of the two input bits.
- The code compares the adder output with the expected sum:
  - If equal → prints "Correct calculation".
  - If not → prints "Wrong calculation at $A_i + B_i$" (showing the input values).

More about $ display() command:
$ display(): is used to display the string to the screen. In order to print variables inside display functions, appropriate format specifiers have to be given for each variable. In addition, $ display() outputs values immediately.

Argument	Description
%h, %H	Display in hexadecimal format
%d, %D	Display in decimal format
%b, %B	Display in binary format
%m, %M	Display in hierarchical name
%s, %S	Display as a string
%t, %T	Display in time format
%f, %F	Display 'real' in decimal format
%e, %E	Display 'real' in an exponential format

In practice, we not only have combinational systems but also sequential systems. For these systems, we need to generate a clock signal before carrying out the verification process. If you want to create a clock period of 10, duty cycle 50%, follow this code:

```
always #5 Clk = !Clk;
initial begin
Clk = 0;
........// STIMULI DRIVER CODE HERE ......
end
```

The procedure for performing a direct test is shown in the Figure 3:

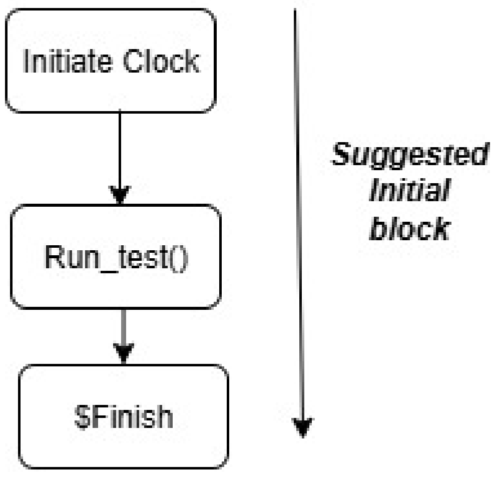


Figure 3: Suggested testbench flow

2.2 For loop in System Verilog

Take the example of a testbench module for the Full Adder with data width n of each signal (A, B, Cin), there are 2^{3n} additions in total. To check the output of all, we should use a for loop.

To avoid repetition of statements, we can use a for loop. Like all other procedural blocks, the for loop requires multiple statements within it to be enclosed by 'begin' and 'end' keywords.

Syntax:

```
for([initialization]; <condition> ; [modifier]) begin
    // Multiple statements
end
```

Since a Full Adder has three inputs (A, B, Cin), there are $2^3 = 8$ possible input combinations. The example testbench using a for loop:

```
module FullAdderTb;
logic A, B, Cin;
logic Sum, Cout;

// Instantiate the DUT (Device Under Test)
FullAdder dut (
    .A(A),
    .B(B),
    .Cin(Cin),
    .Sum(Sum),
    .Cout(Cout)
);

initial begin
    $display(" A B Cin | Cout Sum | Expected ");
    $display("-------------------------------");
    // Loop through all 8 combinations
    for (int i = 0; i < 8; i++) begin
        {A, B, Cin} = i;   // Assign binary pattern
        #5;                // Small delay for simulation
        if ({Cout, Sum} == (A + B + Cin))
            $display(" %0d %0d  %0d  |    %0d    %0d   | Correct",
                      A, B, Cin, Cout, Sum);
        else
```

```
                $display(" %0d %0d  %0d  |   %0d    %0d   | Wrong",
                          A, B, Cin, Cout, Sum);
        end
        $finish;
    end
endmodule
```

2.3 Task in System Verilog

Basic mindset of Task:

- A block of code that can be called from other parts of the testbench or RTL.
- Task can consume simulation time - such as @, #, wait
- Task can have inputs and outputs

Syntax of Task:

```
 // Style 1
task [name];
 input  [port_list];
 inout  [port_list];
 output [port_list];
 begin
  [statements]
 end
endtask

// Style 2
task [name] (input [port_list], inout [port_list], output [port_list]);
 begin
  [statements]
 end
endtask
```

If a task has no arguments:

```
task [name] ();
 begin
  [statements]
 end
endtask
```

We can use a task to create a run_test() task to generate stimuli and check logic. Take the previous testbench example, the Figure 4 shows the flow of the task run_test().

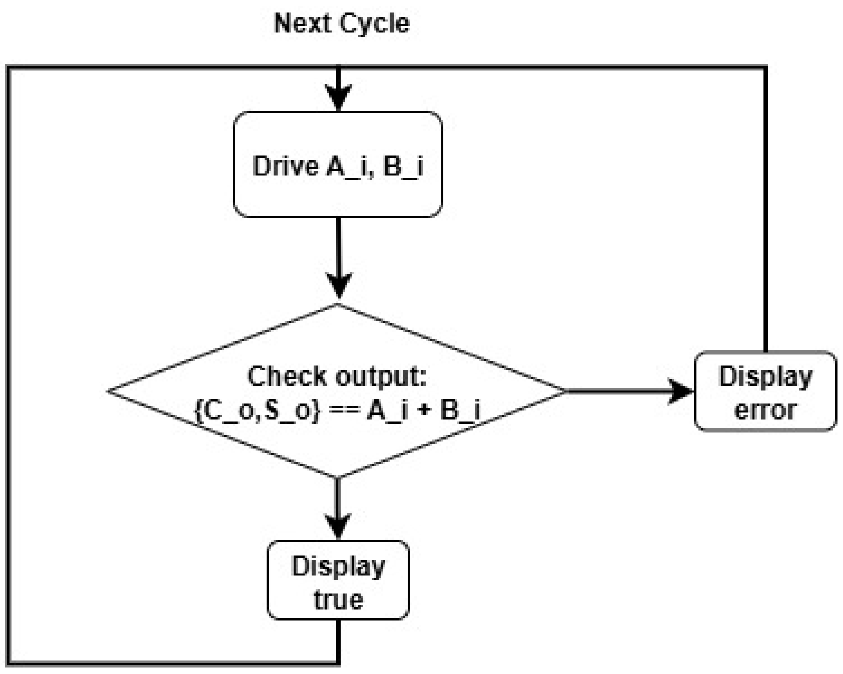


Figure 4: Run_test flow

The run_test task can be coded as below:

```
task run_test();
// In this case, the task has no input and output, just mainly update the input to DUT
    begin
        @(posedge Clk); // sync with clock
        for (int i = 0; i < 2**WIDTH; i++) begin
            for (int j = 0; j < 2**WIDTH; j++) begin
                for (int k = 0; k < 2**WIDTH; k++) begin
                    @(posedge Clk);
                    A_i   = i;
                    B_i   = j;
                    Cin_i = k;
                    #0.1; // allow propagation

                    if ({C_o, S_o} == (A_i + B_i + Cin_i)) begin
                    $display ("PASS: %0d + %0d + %0d = {C_o,S_o} = %0d",
                        A_i, B_i, Cin_i, {C_o, S_o});
                    end
                    else begin
                    $display ("FAIL: %0d + %0d + %0d -> got {C_o,S_o} = %0d (Expected = %0d)",
                        A_i, B_i, Cin_i, {C_o, S_o}, A_i + B_i + Cin_i);
                    end
                end
            end
        end
    end
endtask
```

The full testbench can be shown:

```
module FullAdderTb;
parameter WIDTH = 1;   // 1-bit full adder

// DUT signals
```

```
    logic Clk;
    logic [WIDTH-1:0] A_i, B_i;
    logic Cin_i;
    logic S_o, C_o;

    // Instantiate DUT (Full Adder)
    FullAdder dut (
        .A   (A_i),
        .B   (B_i),
        .Cin (Cin_i),
        .Sum (S_o),
        .Cout(C_o)
    );

    // Clock generation: 10ns period
    initial Clk = 0;
    always #5 Clk = ~Clk;

    // Task to run tests
    task run_test();
        //.....
    endtask

    // Run simulation
    initial begin
        $display("=== Full Adder Testbench Start ===");
        run_test();
        $display("=== Full Adder Testbench Done ===");
        $finish;
    end
endmodule
```

3 Remote Server Access

3.1 Overview

To access the server, you must have:

- Login credentials. For instance:
  - Username: eexxxx_yy
  - Password: eexxxx_yy
- VPN configuration file. This file is provided in the format "username.conf"
- Login node and Compute node

Upon logging in to the server, you will be directed to one of three distinct login nodes, each accessible via a unique login address:

- red – red.doelab.site
- blue – blue.doelab.site
- yellow – yellow.doelab.site

Within these login nodes, you will have access to file editing and PDF reading capabilities, but will be unable to access any tools. To be able to invoke Cadence, Synopsys, or open-source tools, you must execute a command to request a slot on one of the compute nodes. The server will allocate your request based on the current workload, allowing you to be transferred to one of the available compute nodes:

- black
- gray
- white

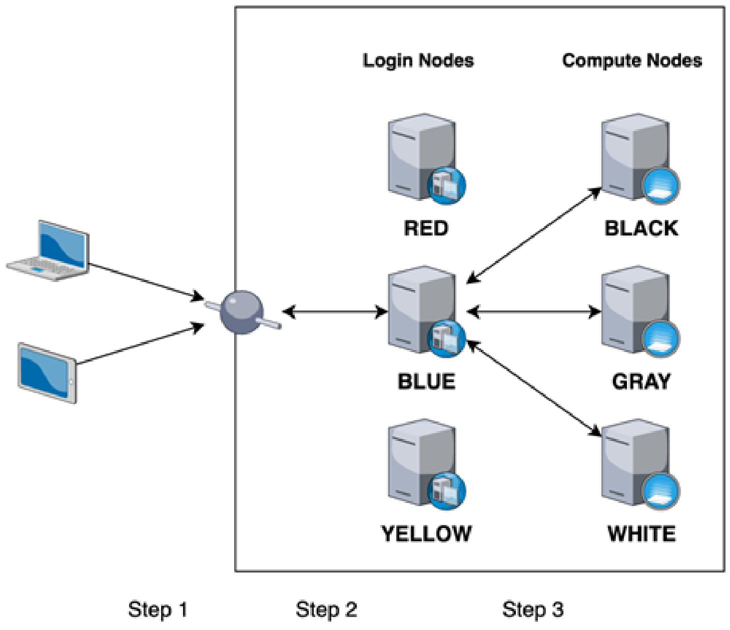


Figure 5: Login nodes and Compute nodes

There are three steps:

1. Establish a VPN Connection using Wireguard– Section 3.2
2. Initiate Remote Desktop Session to Access the Login Node– Section 3.3
3. Connect to a Compute Node for Tool Execution– Section 3.4

3.2 VPN Connection

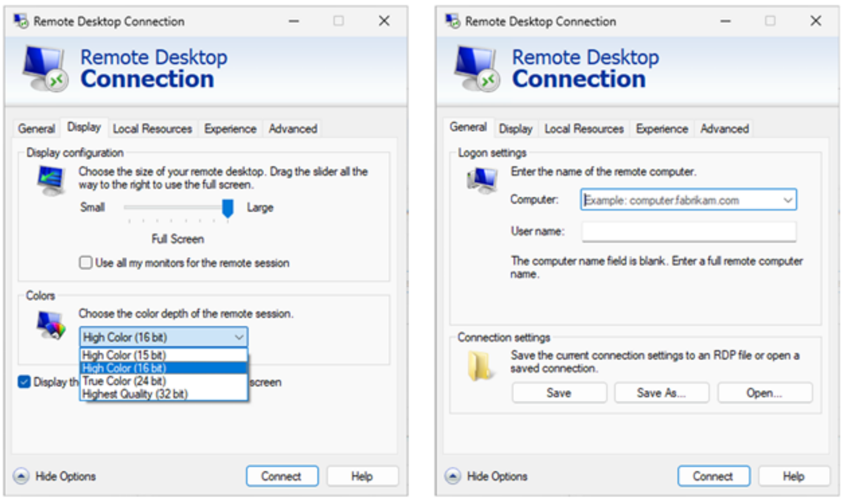
1. Installing WireGuard:
   Download and install the WireGuard client using this link: https://www.wireguard.com/install/
2. Configuring and Activating the VPN
   (a) Open the WireGuard application
   (b) Import the provided VPN configuration file
   (c) Activate the VPN connection
3. Verifying Connection
   (a) Open a PowerShell window or a Terminal
   (b) Run "ping" to verify connectivity to the server:

```
ping yellow.doelab.site
```

   If we connect to another login node, we use a similar "ping" command

3.3 Remote Desktop Access

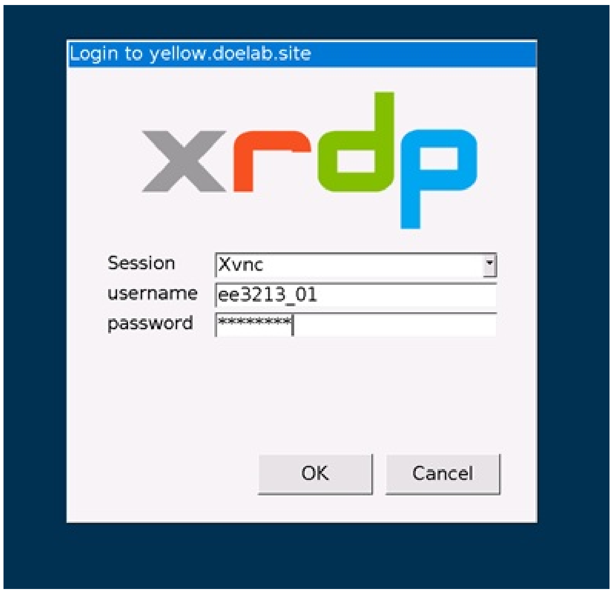
1. Open the Remote Desktop application
2. To have the color rendered properly, switch to the Display tab and in the Colors section, Choose "High Color (16 bit)"
3. Back to the General tab:
   (a) Enter the **login address** in "Computer" field
   (b) Enter your **username** in "User name" field
   (c) You may save the connection now
   (d) Click "Connect" then choose "Yes"



Choose the size of your remote desktop. Drag the slider all the way to the right to use the full screen.
Use all my monitors for the remote session
Choose the color depth of the remote session.
High Color (16 bit)
High Color (15 bit)
High Color (16 bit)
True Color (24 bit)
Highest Quality (32 bit)
Display th... screen

Enter the name of the remote computer.
Computer: Example: computer.fabrikam.com
User name:
The computer name field is blank. Enter a full remote computer name.
Save the current connection settings to an RDP file or open a saved connection.

Figure 6: Login nodes and Compute nodes


Login to yellow.doelab.site

Session Xvnc
username ee3213_01
password *********

Figure 7: Login window

4. A login server window display, as in Figure 7. Provide your **username** and **password** to access the remote desktop session
5. If you want to change your password, while in the login node, open Terminal and use the command "passwd"

Note: For MacOS users, please download Windows App from App Store. For Linux users, you can use Remmina to connect to the login node.

3.4 Compute Node Access

After login to server at login nodes, we open the terminal. To reserve a slot at compute nodes, utilize the following command:

```
    srun --x11 --pty bash
```

In the provided example, after executing the command, the user ee3213_01 is transferred from blue to black:

```
[ee3213_01@blue ~]$ srun --x11 --pty bash
[ee3213_01@black ~]$
```

The **module list** command indicates that nine tools are already loaded for your use:

```
[ee3213_01@black ~]$ module list
Currently Loaded Modulefiles:
1) gcc/13        4) assura/41            7) verdi/W-2024.09-SP1
2) ic/231        5) xcelium/2409         8) verilator/5.034(default)
3) spectre/241   6) vcs/W-2024.09-SP1    9) gtkwave/3.3lts
```

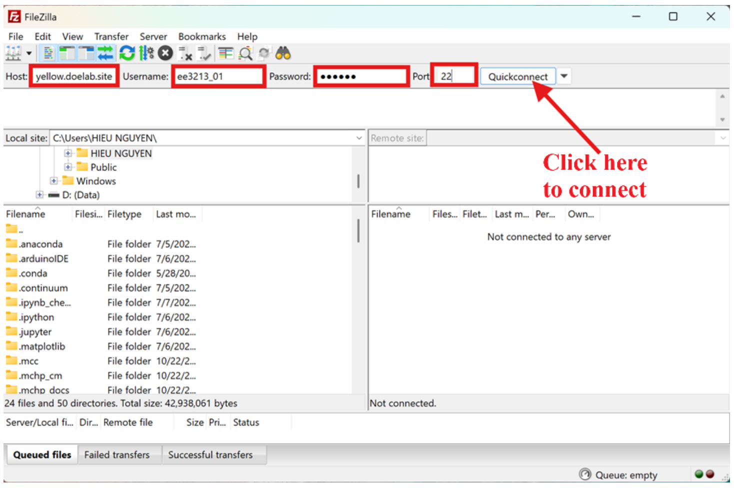
For other tools, you can execute the command "module avail" to ascertain their availability and subsequently load them.

```
[ee3213_01@black ~]$ module load fc
[ee3213_01@black ~]$ module list
Currently Loaded Modulefiles:
```

```
1) gcc/13        5) xcelium/2409            9) gtkwave/3.3lts
2) ic/231        6) vcs/W-2024.09-SP1       10) fc/W-2024.09-SP3
3) spectre/241   7) verdi/W-2024.09-SP1
4) assura/41     8) verilator/5.034(default)
```

3.5 File Transfer with FileZilla

1. Installation Download and install the appropriate FileZilla version for your operating system using this link: https://filezilla-project.org/download.php
2. Connect to the server
   (a) Open FileZilla
   (b) In the “Host” field, enter your login server address
   (c) Enter your username and password in the respective fields
   (d) Set the “Port” to 22
   (e) Click “Quickconnect” to establish the connection



Host: yellow.doelab.site Username: ee3213_01 Password: •••••• Port: 22

Local site: C:\Users\HIEU NGUYEN\

Filename	Filesi...	Filetype	Last mo...
..			
.anaconda		File folder	7/5/202...
.arduinoIDE		File folder	7/6/202...
.conda		File folder	5/28/20...
.continuum		File folder	7/5/202...
.ipynb_che...		File folder	7/7/202...
.ipython		File folder	7/6/202...
.jupyter		File folder	7/6/202...
.matplotlib		File folder	7/6/202...
.mcc		File folder	10/22/2...
.mchp_cm		File folder	10/22/2...
.mchp_docs		File folder	10/22/2...

24 files and 50 directories. Total size: 42,938,061 bytes

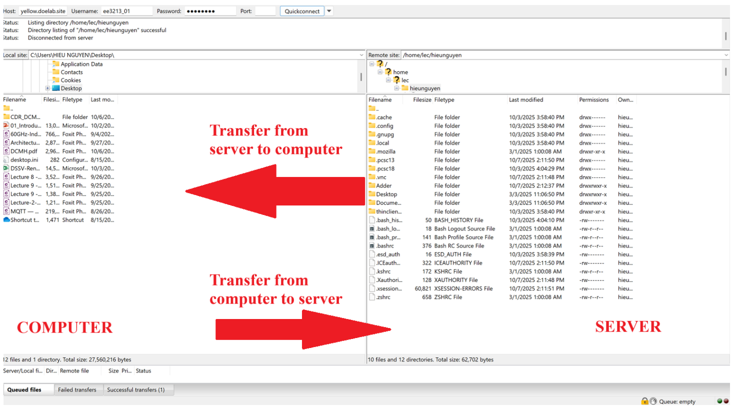
Not connected to any server

Not connected.

FileZilla
File Edit View Transfer Server Bookmarks Help
Quickconnect
HIEU NGUYEN
Public
Windows
D: (Data)
Remote site:
Click here
to connect
Filename Files... Filet... Last m... Per... Own...
Server/Local fi... Dir... Remote file Size Pri... Status
Queued files
Failed transfers
Successful transfers
Queue: empty

Figure 8: Use FileZilla to connect to server

3. After connecting to the server, the GUI is shown in Figure 9.
   (a) To transfer files from your computer to server, move files from left to right
   (b) To transfer files from server to your computer, move files from right to left



Host: yellow.doelab.site Username: ee3213_01 Password: ••••••••• Port:

```
Status:    Listing directory /home/lec/hieunguyen
Status:    Directory listing of "/home/lec/hieunguyen" successful
Status:    Disconnected from server
```

Local site: C:\Users\HIEU NGUYEN\Desktop\

Filename	Filesi...	Filetype	Last mo...
..			
CDR_DCM...		File folder	10/6/20...
01_Introdu...	13,0...	Microsof...	10/2/20...
60GHz-Ind...	766,...	Foxit Ph...	9/4/202...
Architectu...	2,87...	Foxit Ph...	9/27/20...
DCMH.pdf	2,96...	Foxit Ph...	10/6/20...
desktop.ini	282	Configur...	8/15/20...
DSSV-Ren...	14,5...	Microsof...	10/3/20...
Lecture 8 -...	3,52...	Foxit Ph...	9/26/20...
Lecture 9 -...	1,51...	Foxit Ph...	9/25/20...
Lecture 9 -...	1,38...	Foxit Ph...	9/25/20...
Lecture-2-...	1,21...	Foxit Ph...	9/25/20...
MQTT — ...	219,...	Foxit Ph...	8/26/20...
Shortcut t...	1,471	Shortcut	8/15/20...

12 files and 1 directory. Total size: 27,560,216 bytes

Remote site: /home/lec/hieunguyen

Filename	Filesize	Filetype	Last modified	Permissions	Own...
..					
.cache		File folder	10/3/2025 3:58:40 PM	drwx------	hieu...
.config		File folder	10/3/2025 3:58:40 PM	drwx------	hieu...
.gnupg		File folder	10/3/2025 3:58:40 PM	drwx------	hieu...
.local		File folder	10/3/2025 3:58:40 PM	drwx------	hieu...
.mozilla		File folder	3/1/2025 1:00:08 AM	drwxr-xr-x	hieu...
.pcsc13		File folder	10/7/2025 2:11:50 PM	drwx------	hieu...
.pcsc18		File folder	10/3/2025 4:04:29 PM	drwx------	hieu...
.vnc		File folder	10/7/2025 2:11:48 PM	drwx------	hieu...
Adder		File folder	10/7/2025 2:12:37 PM	drwxrwxr-x	hieu...
Desktop		File folder	3/3/2025 11:06:50 PM	drwxrwxr-x	hieu...
Docume...		File folder	3/3/2025 11:06:50 PM	drwxrwxr-x	hieu...
thinclien...		File folder	10/3/2025 3:58:40 PM	drwxr-xr-x	hieu...
.bash_his...	50	BASH_HISTORY File	10/3/2025 4:04:10 PM	-rw-------	hieu...
.bash_lo...	18	Bash Logout Source File	3/1/2025 1:00:08 AM	-rw-r--r--	hieu...
.bash_pr...	141	Bash Profile Source File	3/1/2025 1:00:08 AM	-rw-r--r--	hieu...
.bashrc	376	Bash RC Source File	3/1/2025 1:00:08 AM	-rw-r--r--	hieu...
.esd_auth	16	ESD_AUTH File	10/3/2025 3:58:39 PM	-rw-------	hieu...
.ICEauth...	322	ICEAUTHORITY File	10/7/2025 2:11:50 PM	-rw-------	hieu...
.kshrc	172	KSHRC File	3/1/2025 1:00:08 AM	-rw-r--r--	hieu...
.Xauthori...	128	XAUTHORITY File	10/7/2025 2:11:48 PM	-rw-------	hieu...
.xsession...	60,821	XSESSION-ERRORS File	10/7/2025 2:11:51 PM	-rw-------	hieu...
.zshrc	658	ZSHRC File	3/1/2025 1:00:08 AM	-rw-r--r--	hieu...

10 files and 12 directories. Total size: 62,702 bytes

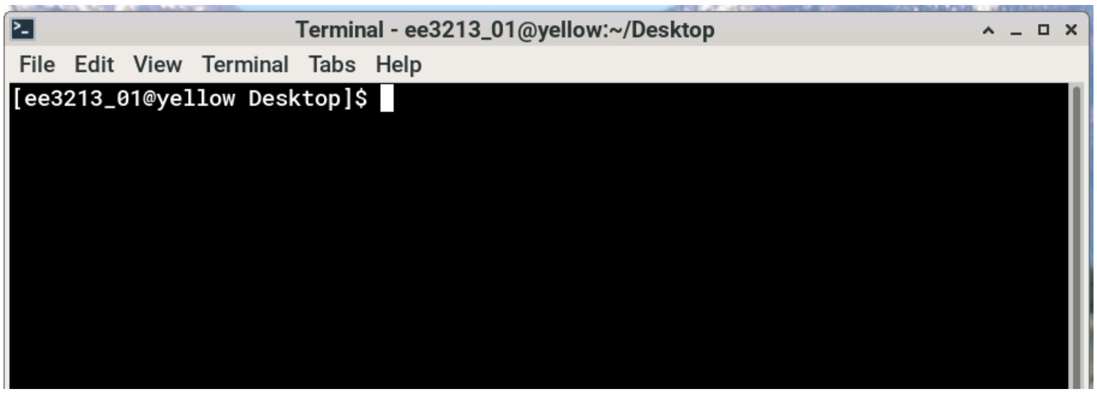
Figure 9: Use FileZilla to connect to server

4 Simulation Tool - Basic Xcelium Syntax

Xcelium from Cadence is the simulation tool for your design verification. It encompasses several tools in one executable, called **xrun**. In this section, we focus on some of the operations of the **xrun**:

- Compilation: Compile HDL code written in HDL language like System Verilog, Verilog and VHDL.
- Elaboration: Design after compilation will be elaborated into the internal database.
- Simulation: Simulate the logic behavior of DUT in the testbench.

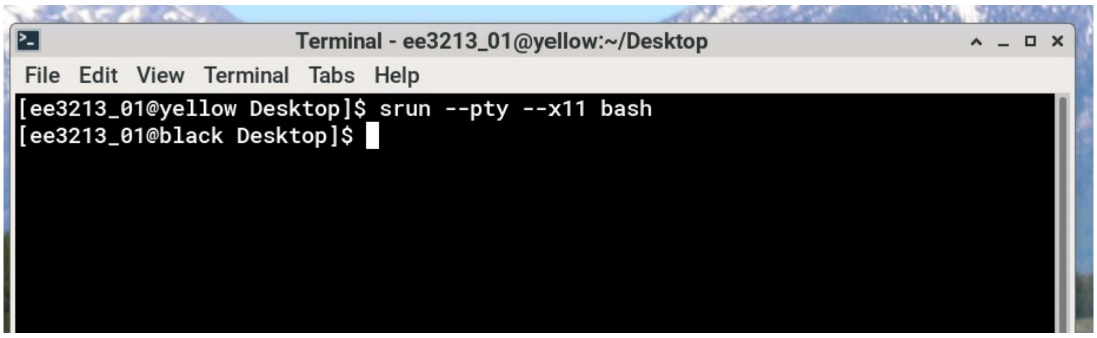
After login to server, right click and choose **Open Terminal Here** to open the terminal as shown in Figure 10. The terminal currently displays your username and login node.


Terminal - ee3213_01@yellow:~/Desktop
File Edit View Terminal Tabs Help

```
[ee3213_01@yellow Desktop]$ 
```

Figure 10: Terminal at yellow node

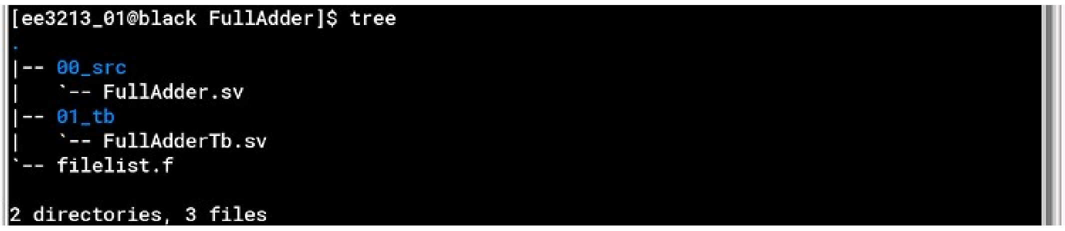
Before using the simulation tool, you have to transfer to the compute node (default is the black node) by using the command as in Figure 11.


Terminal - ee3213_01@yellow:~/Desktop
File Edit View Terminal Tabs Help

```
[ee3213_01@yellow Desktop]$ srun --pty --x11 bash
[ee3213_01@black Desktop]$ 
```

Figure 11: Change to compute note

Before going to the simulation tool, you can reference the suggested project structure in the Figure 12.

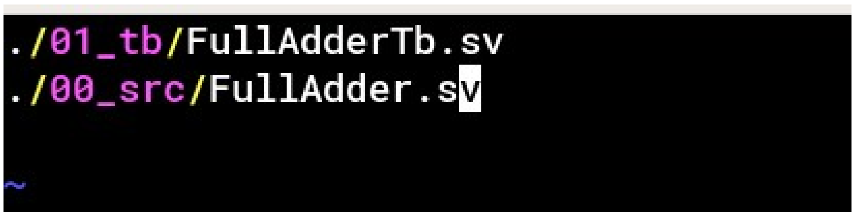


```
[ee3213_01@black FullAdder]$ tree
.
|-- 00_src
|   `-- FullAdder.sv
|-- 01_tb
|   `-- FullAdderTb.sv
`-- filelist.f

2 directories, 3 files
```

Figure 12: Suggested structure project

The file list contains the paths of all RTL/Testbench source files in the project that need to be compiled. The list of files should be organized in a top-down hierarchical manner as shown in Figure 13.

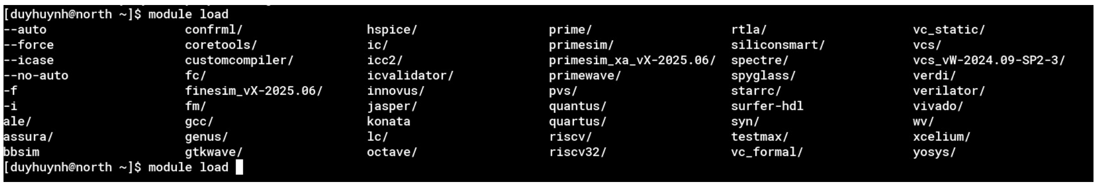


```
./01_tb/FullAdderTb.sv
./00_src/FullAdder.sv

~
```

Figure 13: Example of file list for Full Adder

Next, load the tool Xcelium by using the command **module load** like in Figure 14. Press ”Tab” to see the list of tools.



```
[duyhuynh@north ~]$ module load
--auto                 confrml/               hspice/                prime/                 rtla/                  vc_static/
--force                coretools/             ic/                    primesim/              siliconsmart/          vcs/
--icase                customcompiler/        icc2/                  primesim_xa_vX-2025.06/ spectre/              vcs_vW-2024.09-SP2-3/
--no-auto              fc/                    icvalidator/           primewave/             spyglass/              verdi/
-f                     finesim_vX-2025.06/    innovus/               pvs/                   starrc/                verilator/
-i                     fm/                    jasper/                quantus/               surfer-hdl             vivado/
ale/                   gcc/                   konata                 quartus/               syn/                   wv/
assura/                genus/                 lc/                    riscv/                 testmax/               xcelium/
bbsim                  gtkwave/               octave/                riscv32/               vc_formal/             yosys/
[duyhuynh@north ~]$ module load
```

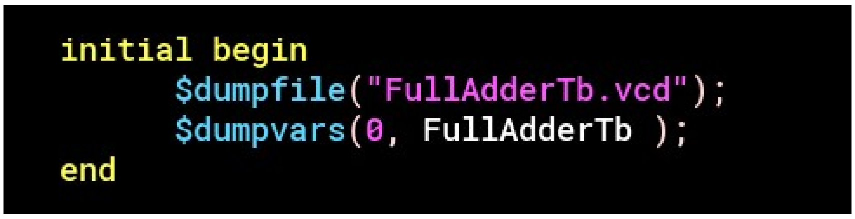
Figure 14: List of tools

After loading the tool, run the tool by using the following command:
xrun -f filelist.f +access+rw -sv

- xrun: executable command of Xcelium - do compilation, elaboration, and simulation
- -f filelist.f: read files from the filelist
- +access+rw: set access permission to simulation, so that you can read (r) or write(w) the signal simulation
- -sv: this notifies Xcelium to enable System Verilog mode - treat file as System Verilog

Before running this command, you should do those things:

- Add $finish in the testbench: Tell the simulator to end the simulation
- Add system task to generate waveform files as shown in Figure 15



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

Figure 15: Generate waveform files code

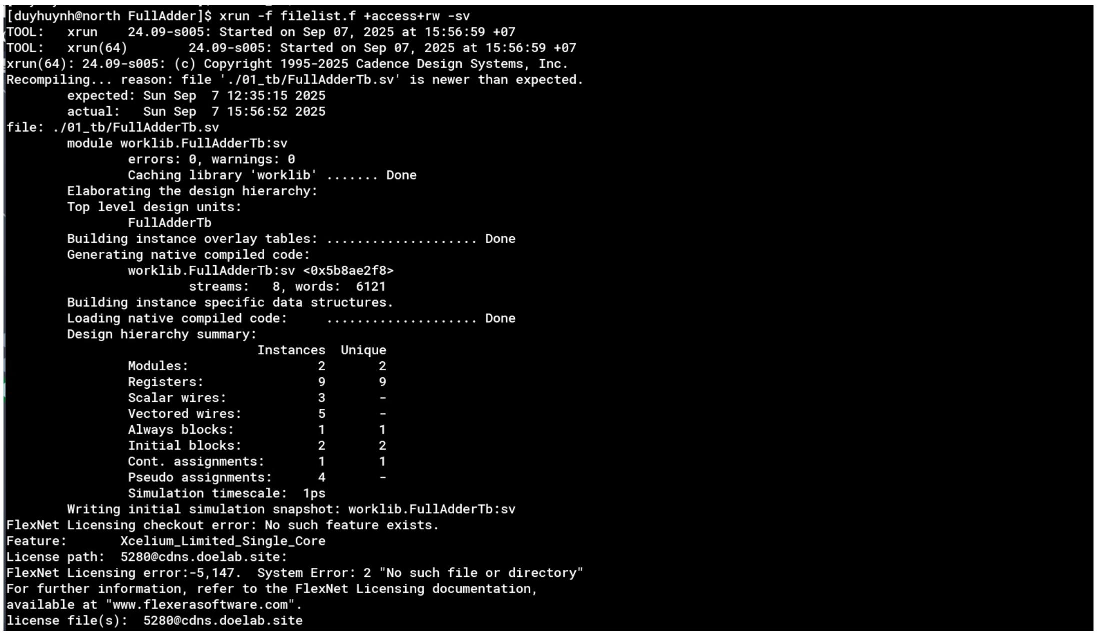
The **$dumpfile(”Name.vcd”)** specifies the output file where the waveform will be written. This creates a file Name.vcd

The **$dumpvars(0, Name)** tells the simulator which signals to dump into the VCD file. Name is the module or instance to dump, 0 means dump everything.

Some important files are created while running xrun:

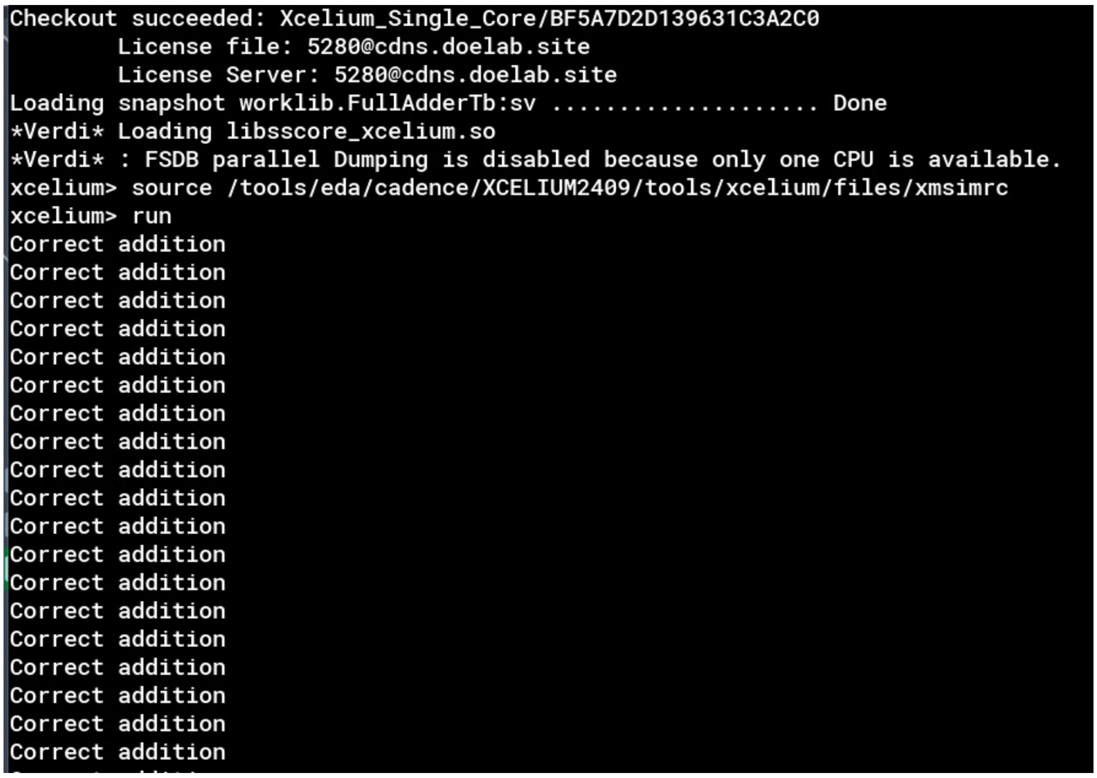
- xrun.log: The logfile with all the errors and warnings
- xrun.history: The last command line used to invoke Xcelium.
- worklib/: scratch directory with compiled files

The terminal when run xrun as shown in Figure 16 and Figure 17



```
[duyhuynh@north FullAdder]$ xrun -f filelist.f +access+rw -sv
TOOL:   xrun    24.09-s005: Started on Sep 07, 2025 at 15:56:59 +07
TOOL:   xrun(64)        24.09-s005: Started on Sep 07, 2025 at 15:56:59 +07
xrun(64): 24.09-s005: (c) Copyright 1995-2025 Cadence Design Systems, Inc.
Recompiling... reason: file './01_tb/FullAdderTb.sv' is newer than expected.
        expected: Sun Sep  7 12:35:15 2025
        actual:   Sun Sep  7 15:56:52 2025
file: ./01_tb/FullAdderTb.sv
        module worklib.FullAdderTb:sv
                errors: 0, warnings: 0
                Caching library 'worklib' ....... Done
        Elaborating the design hierarchy:
        Top level design units:
                FullAdderTb
        Building instance overlay tables: .................... Done
        Generating native compiled code:
                worklib.FullAdderTb:sv <0x5b8ae2f8>
                        streams:   8, words:  6121
        Building instance specific data structures.
        Loading native compiled code:     .................... Done
        Design hierarchy summary:
                                 Instances  Unique
                Modules:                 2       2
                Registers:               9       9
                Scalar wires:            3       -
                Vectored wires:          5       -
                Always blocks:           1       1
                Initial blocks:          2       2
                Cont. assignments:       1       1
                Pseudo assignments:      4       -
                Simulation timescale:  1ps
        Writing initial simulation snapshot: worklib.FullAdderTb:sv
FlexNet Licensing checkout error: No such feature exists.
Feature:       Xcelium_Limited_Single_Core
License path:  5280@cdns.doelab.site:
FlexNet Licensing error:-5,147.  System Error: 2 "No such file or directory"
For further information, refer to the FlexNet Licensing documentation,
available at "www.flexerasoftware.com".
license file(s):  5280@cdns.doelab.site
```

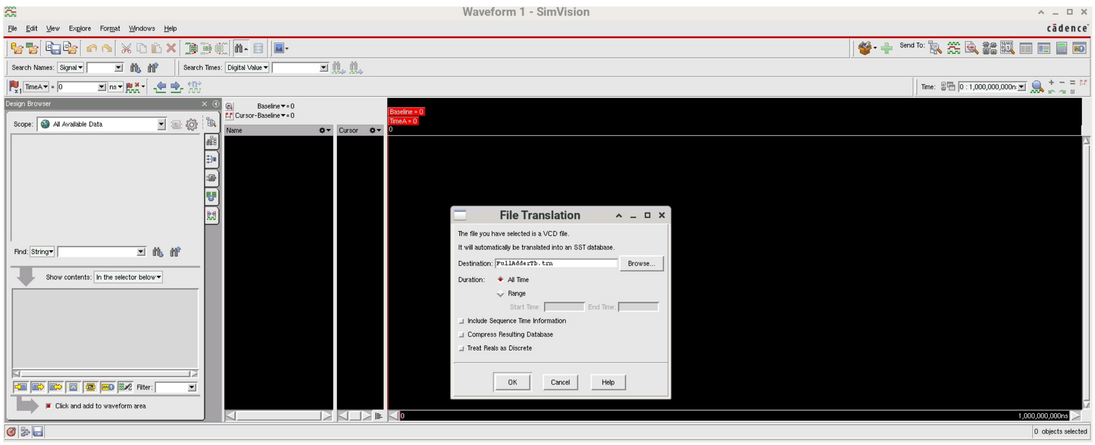
Figure 16: Compilation and elaboration result



```
Checkout succeeded: Xcelium_Single_Core/BF5A7D2D139631C3A2C0
        License file: 5280@cdns.doelab.site
        License Server: 5280@cdns.doelab.site
Loading snapshot worklib.FullAdderTb:sv .................... Done
*Verdi* Loading libsscore_xcelium.so
*Verdi* : FSDB parallel Dumping is disabled because only one CPU is available.
xcelium> source /tools/eda/cadence/XCELIUM2409/tools/xcelium/files/xmsimrc
xcelium> run
Correct addition
Correct addition
Correct addition
Correct addition
Correct addition
Correct addition
Correct addition
Correct addition
Correct addition
Correct addition
Correct addition
Correct addition
Correct addition
Correct addition
Correct addition
Correct addition
Correct addition
Correct addition
Correct addition
```

Figure 17: Simulation result

Then type this command on your terminal: **simvision Name.vcd &** take the FullAdder example it would be **simvision FullAdderTb.vcd &** to laund the Simvision Gui so that you can see the waveform of your testbench like in Figure 18. Next just click OK at the File Transition stage.



File Translation

The file you have selected is a VCD file.

It will automatically be translated into an SST database.

Destination: FullAdderTb.trn

Duration: [✓ All Time] Range

Start Time: End Time:

Include Sequence Time Information

Compress Resulting Database

Treat Reals as Discrete

Figure 18: Simvision first look

Then you will see the display like in Figure 19. The project structure is on your left side at the Design Browser, take a right click to your testbench module, select ”Send to Waveform Window” to see the Waveform, also you can choose ”Send to Schematic Tracer” to see the schematic of your RTL code.

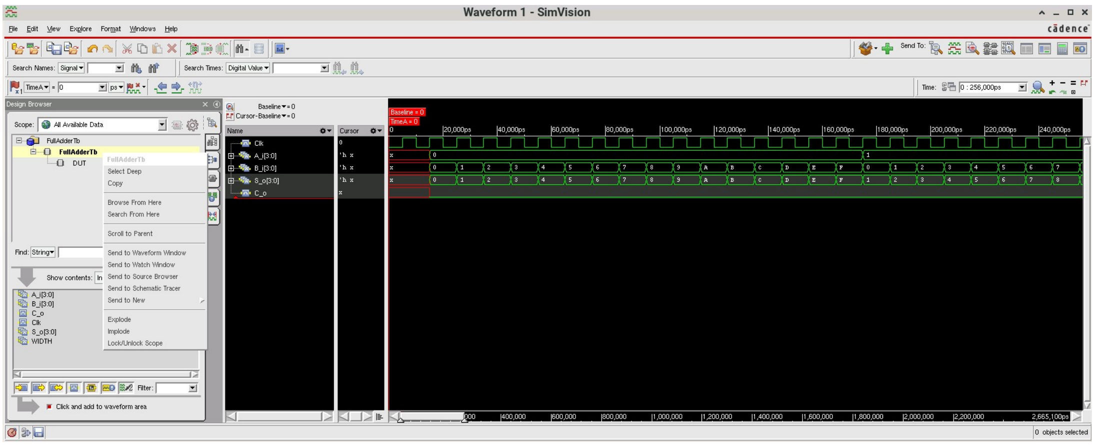


Figure 19: Simvision waveform window