

Training on Functional Verification Methodology Using UVM

LAB Verilog Testbench

Objectives

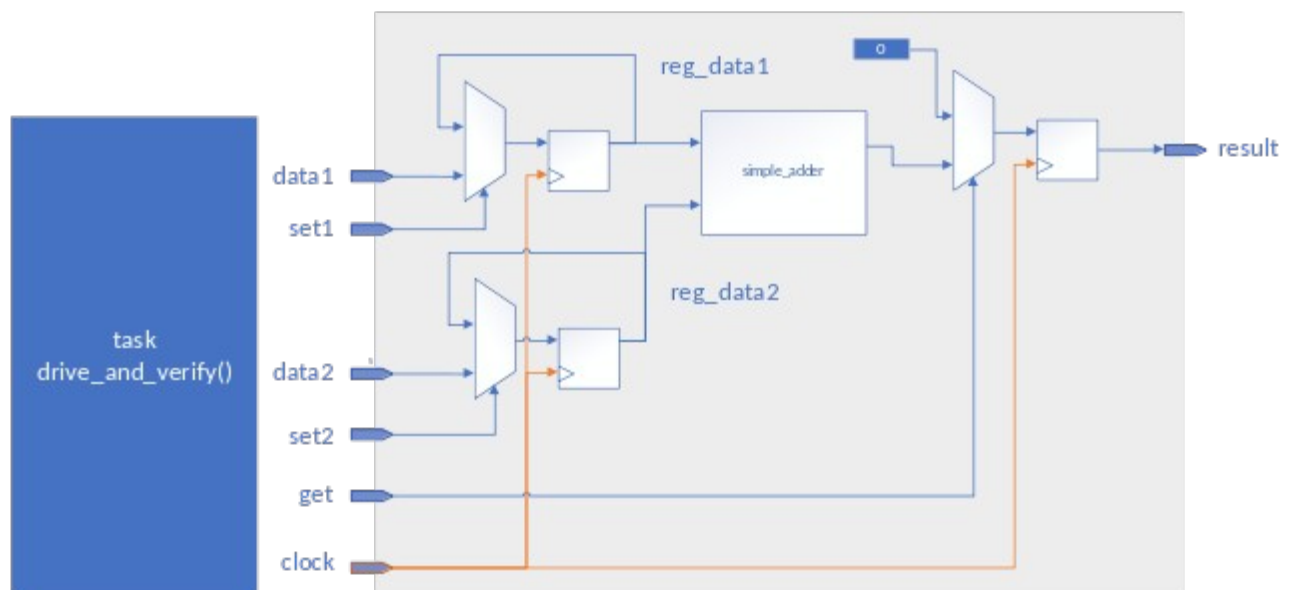
Verilog is a Hardware Description Language (HDL) used to describe electronic circuits at the register transfer level (RTL).

This lab has as objective to improve the Verilog testbench students' skill, using features as functions and tasks to perform the test of their previous design. We will create a testbench that will call a task responsible to drive all the possible values of entry and verify the output.

Test Specification

The design under test (DUT) is the adder of the previous lab. For convenience a full design is given in the lab directory, but you can also switch to your own design.

In this lab we will implement a task `drive_and_verify()` and function `sum` that we will use in the testbench to actually verify the design.



Get Started

Untar the training tarball:

```
mkdir <TRAINING_WORKAREA>  
cd <TRAINING_WORKAREA>  
tar zxvf <TRAINING_NAME>.tgz
```

Lab files can be found under:

<TRAINING_NAME>/labs/<LABNAME>

To launch the simulation,

```
cd simulation/<SIMULATOR>  
./runsim.sh
```

Note: in order to launch using LSF, you must first setup the variable LAB_LAUNCHER

```
setenv LAB_LAUNCHER 'bsub -I -q gui -P mcdverif -R "select[rh60]"'
```

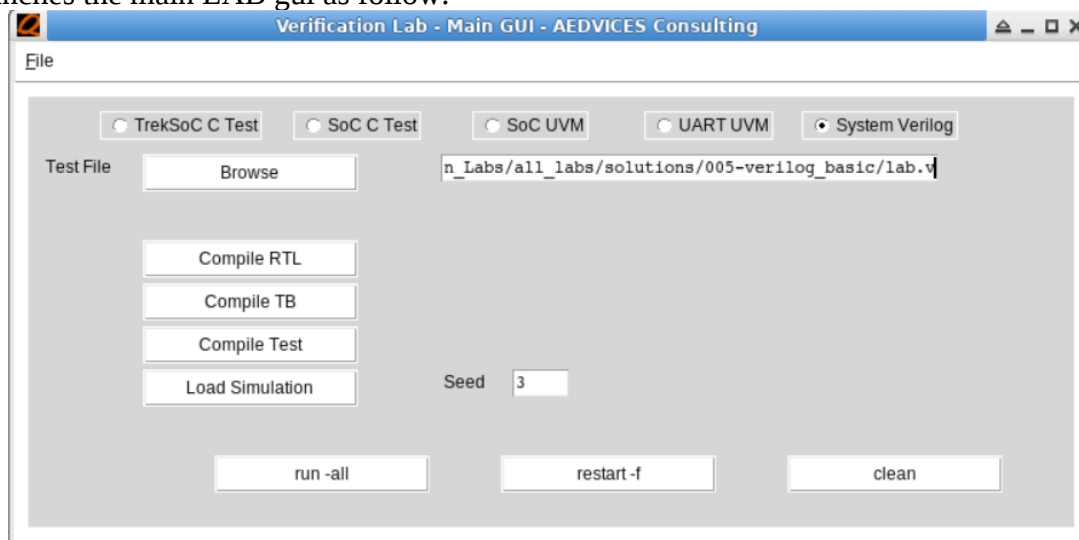
Where SIMULATOR is:

ius for Cadence Incisive / Xcelium

questa for Mentor Questa

For Questa under Windows, click on runquesta.bat

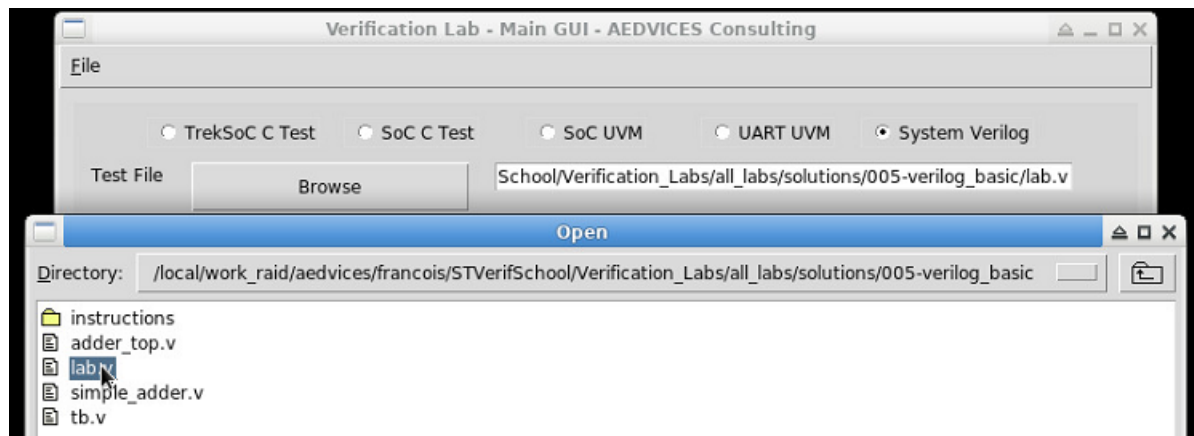
This launches the main LAB gui as follow:



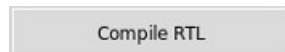
In this lab, we will use the SystemVerilog mode. So the following checkbox should be set:

☒ System Verilog

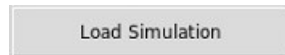
Use the GUI browse to open the file: labs/labNN-verilog_testbench/**lab.v**



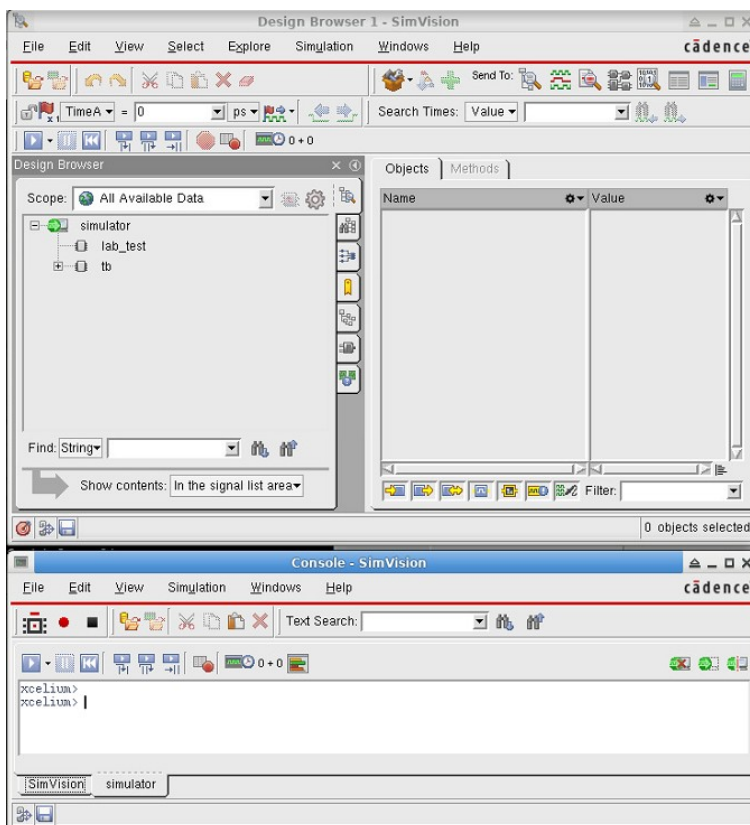
Click on “Compile RTL”



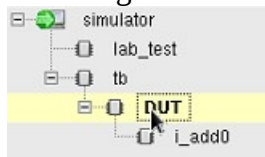
Click on “Load Simulation”



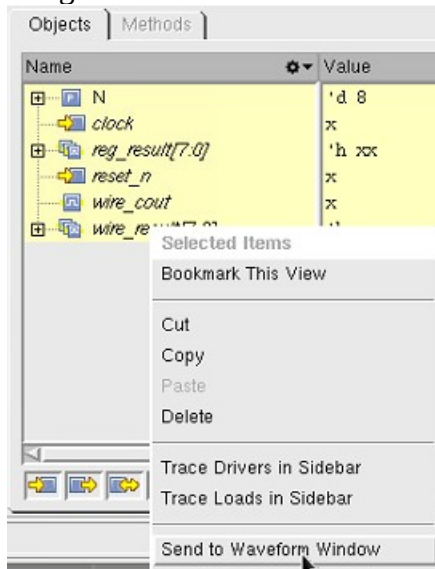
This should open your simulator with the loaded design.




Browse the design under tb.DUT

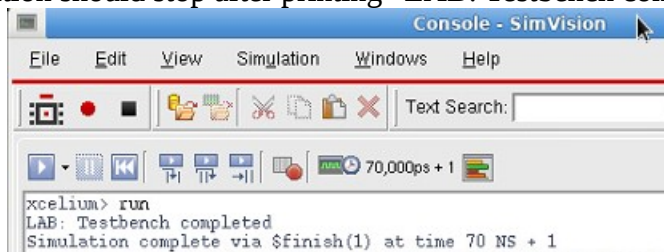


Select all signals and add them to the waveform viewer:

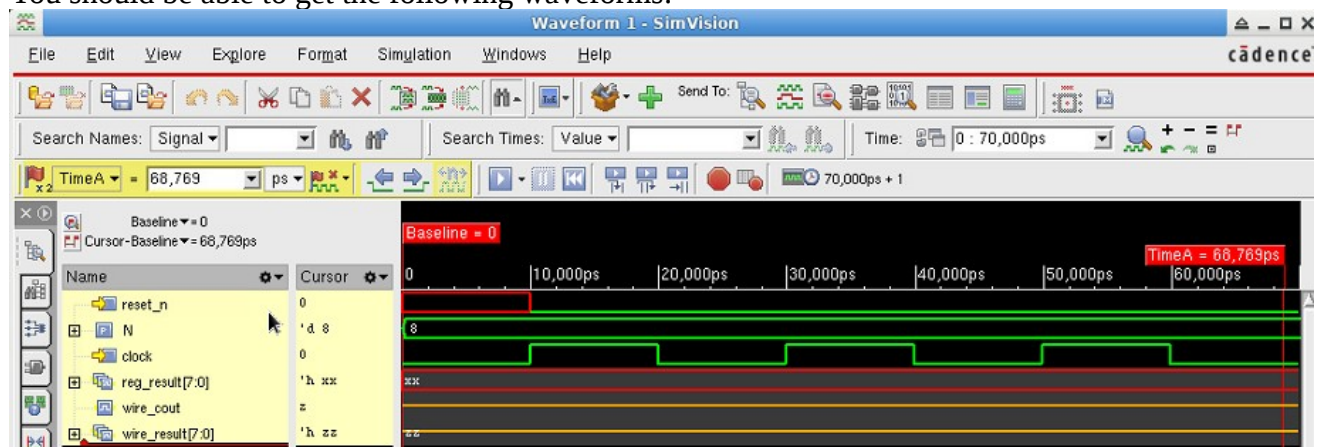


Run the simulation clicking on : 
Or by typing the command “run”

Simulation should stop after printing “LAB: Testbench completed”



You should be able to get the following waveforms:



Instructions

Step 1: Sum Function.

- Open the file: <SANDBOX>/labs-Xdays/labNN-verilog_testbench /tb.sv
- Search for **LAB-TODO-STEP-1**
- Create a function called *sum()* to add two N size values and return the result of N+1 size. (Carry out)
- Consult the presentation slides and the cheat sheet if needed.

Step 2: *drive_verify()* task

- Open the file: <SANDBOX>/labs-Xdays/labNN-verilog_testbench /tb.sv
- Search for **LAB-TODO-STEP-2a**
- Uncomment the task content.
- This task must drive all possible input values to the DUT. So, we will use the *for loop* to generate these inputs.

```
for(ii=0;ii<((2**N)-1);ii=ii+1) begin
    for(jj=0;jj<((2**N)-1);jj=jj+1) begin
        for(kk=0;kk<((2**3));kk=kk+1) begin
            {set1, set2, get} <= kk;

            data1 <= ii;
            data2 <= jj;

            @(posedge clk);
        end
    end
end
```

- To verify the design, we must use our model, that is, in this case, the *sum* function. To use the model correctly we must consider the configuration signals(*set1*, *set2* and *get*), therefore we may use the following code within the loop:

```
if(set1 === 1'b1) begin
    aux_data1 <= ii;
end
if(set2 === 1'b1) begin
    aux_data2 <= jj;
end

if(get === 1'b1) begin
    sum_result <= sum(aux_data1, aux_data2);
end
else begin
    sum_result <= 'h0;
end
```

```
#2;  
  
if({cout,result} !== sum_result) begin  
    $display("**ERROR: data1 = %d, data2 = %d", data1, data2);  
end
```

- Search for **LAB-TODO-STEP-2b**
- Uncomment to wait for a positive clock edge.
- Search for **LAB-TODO-STEP-2c**
- Uncomment to call the sum function.
 - o Q: Could we declare the *drive_and_verify()* task as a function?
 - o Q: We called the *sum()* function inside the task, the opposite is permitted? Why?

Step 3: Call the de *drive_and_verify()* task

- Search for **LAB-TODO-STEP-3**
- In the initial block, uncomment the task call.

Step 4: Compile and Run simulation

- After the load phase, add the testbench signals to the waveform and run. Observe the signals behaviour.