

Training on Functional Verification Methodology *Using UVM*

LAB Verilog Basic

Objectives

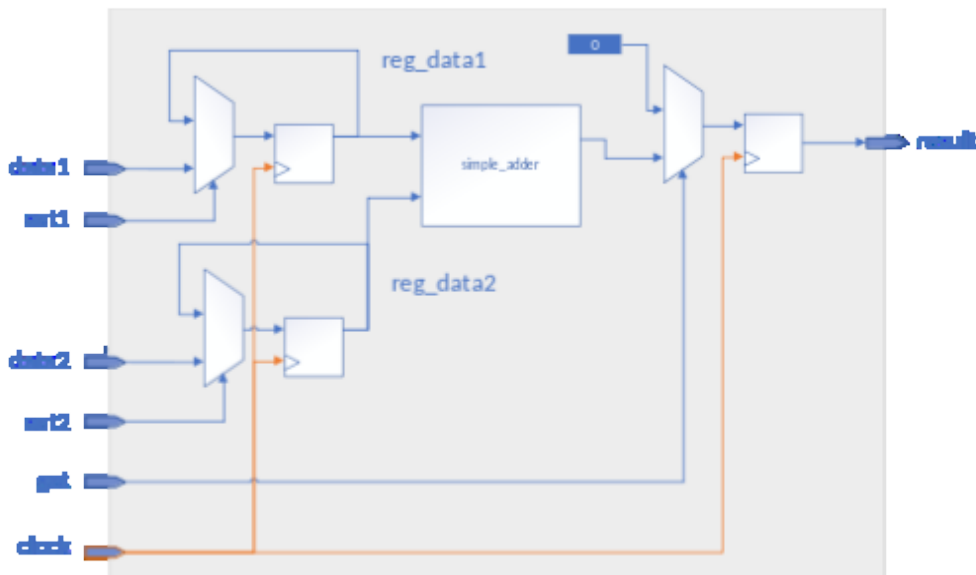
Verilog is a Hardware Description Language (HDL) used to describe electronic circuits at the register transfer level (RTL).

The objective of this lab is to present and use the key Verilog features seen in the training presentations. This practical exercise will improve the student skills in relation to Verilog module creation and instantiation, concept of sequential and combinational circuits in Verilog and the simple testbench construction.

Design Specification

The design you have to implement is a adder with the following characteristics:

- Two input data signals (**i_data1** and **i_data2**) of a parametrized width “N”;
- Two input signals **i_set1** and **i_set2** which, when set to 1, will register the data (synchronous process) ;
- One output **o_result** signal which will be driven to the computed data when **i_get** is set (synchronous process) ;
- One input **i_get** signal that will control when the output result is “outputted”;
- One **o_carry** signal that will indicate if there is an overflow;
- Clocks and Reset.



For the purpose of the lab, we will implement a simple adder sub-module.



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

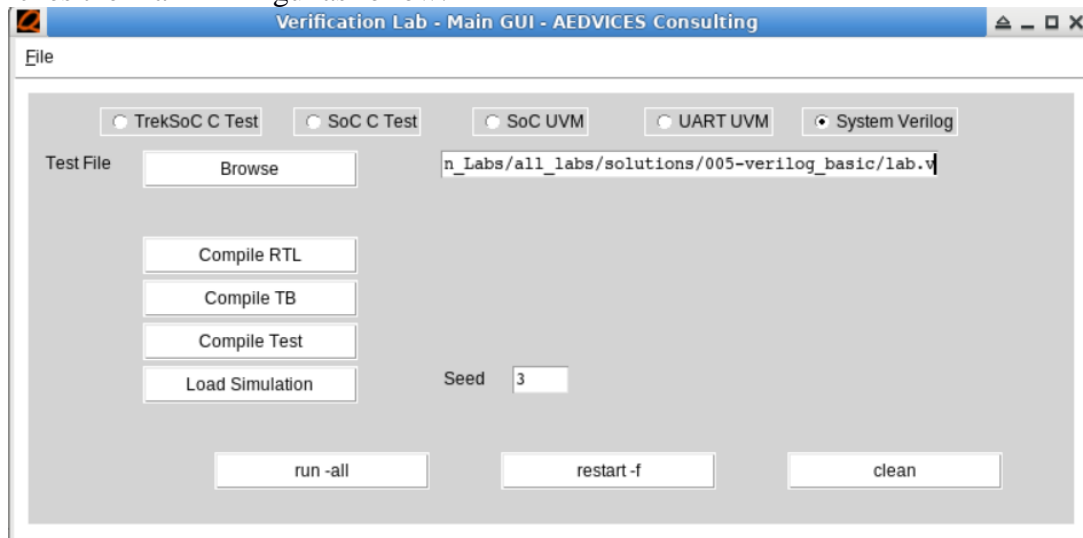
```
setenv LAB_LAUNCHER 'bsub -I -q gui -P mcdverif -R "select[rh60]"'
```

Where SIMULATOR is:

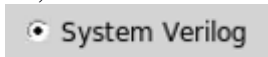
ius for Cadence Incisive / Xcelium
questa for Mentor Questa

For Questa under Windows, click on runquesta.bat

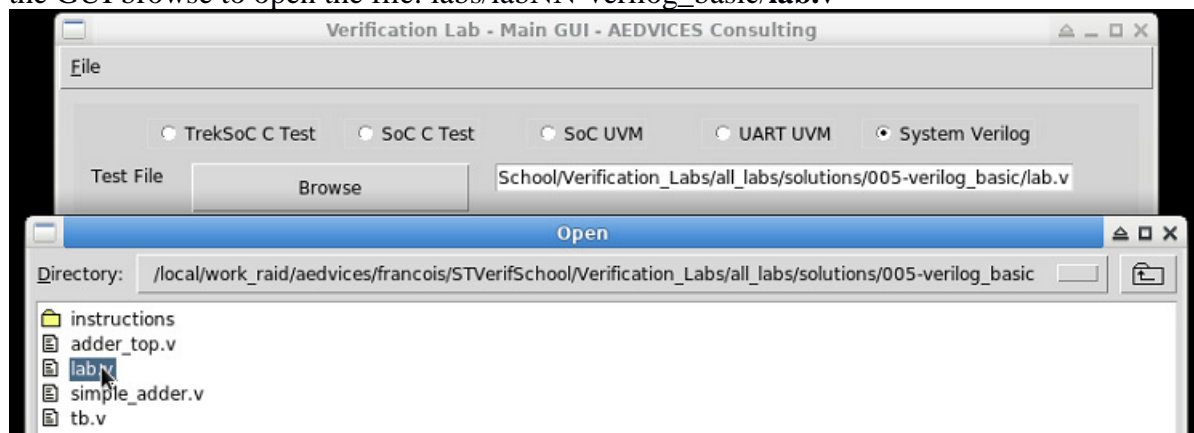
This launches the main LAB gui as follow:



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



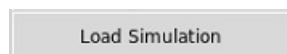
Use the GUI browse to open the file: labs/labNN-verilog_basic/**lab.v**



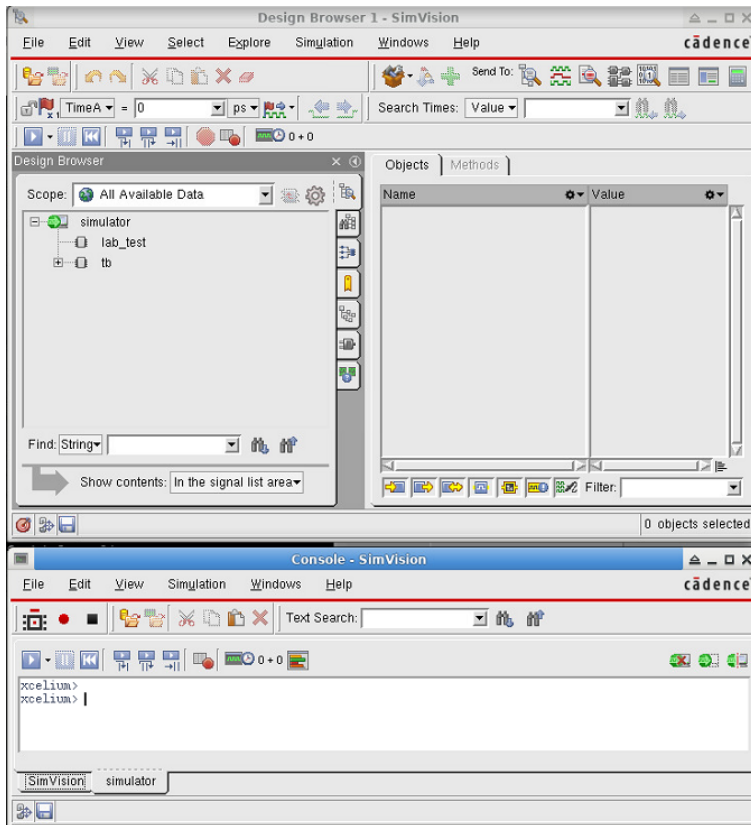
Click on “Compile RTL”



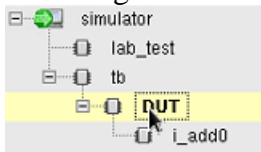
Click on “Load Simulation”



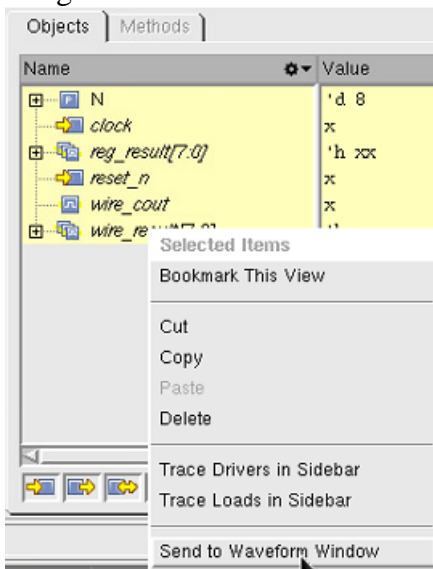
This should open your simulator with the loaded design.




Browse the design under tb.DUT

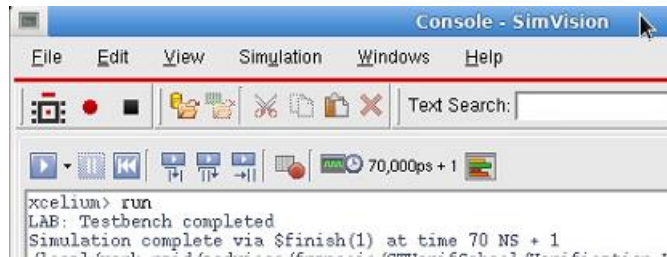


Select all signals and add them to the waveform viewer:

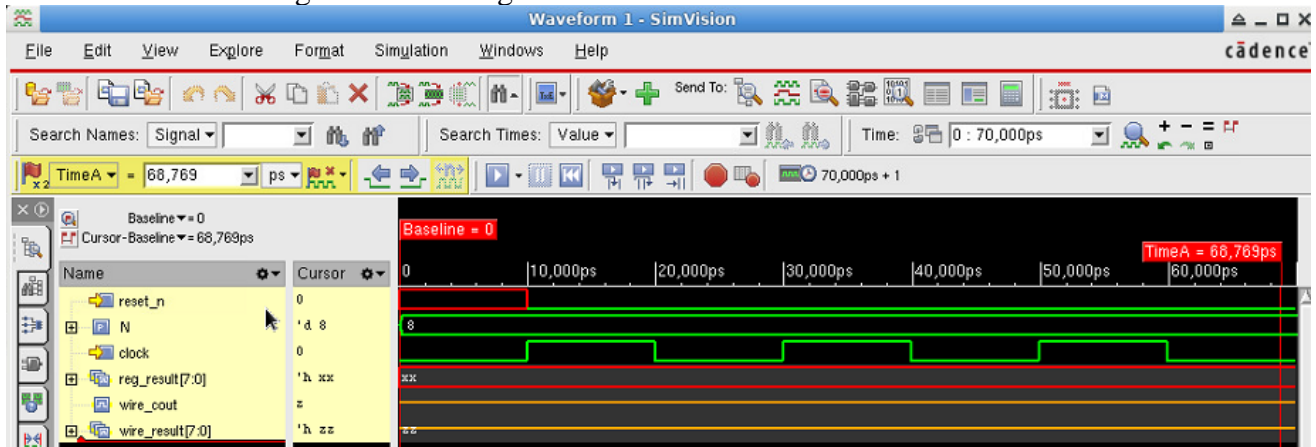


Run the simulation clicking on : 
Or by typing the command “run”

Simulation should stop after printing “LAB: Testbench completed”



You should be able to get the following waveforms:



Lab Instructions

Go into the lab directory : labs/NN-verilog_basic/

You will find the following files:

lab.v	Which includes all others
simple_adder.v	To implement a simple adder module
adder_top.v	To implement the top adder design
tb.v	To implement a basic testbench

These files contain basic empty designs and testbenches that you now have to complete.

Step 1: Simple Full Adder.

- Open the file : *simple_adder.v* in your favourite editor.

```
module simple_adder #(parameter N = 4) (  
    //LAB-TODO-STEP1-a: Define the module interface  
);  
  
    //LAB-TODO-STEP1-b: Describe the module behavioral  
endmodule
```

- Search for **LAB-TODO-STEP-1-a**
- Define the module interface (Inputs and Outputs):

Name	Width	Direction
o_result	N	Output
o_carry	1	Output
i_data1	N	Input
i_data2	N	Input

- Search for **LAB-TODO-STEP-1-b**
- Implement a combinational module behavior using “assign”

Step 2: Top Full Adder

- Open the file : *adder_top.v* in your favourite editor.
- Search for **LAB-TODO-STEP-2-a**
- Define the module interface (Inputs and Outputs) .

Name	Width	Direction
o_result	N	Output
o_carry	1	Output
i_data1	N	Input
i_data2	N	Input
i_set1	1	Input
i_set2	1	Input
i_get	1	Input

- Search for **LAB-TODO-STEP-2-b**
- Declare the module internal registers .

- Search for **LAB-TODO-STEP-2-c**
- Describe the clock synchronous behaviour to set the input registers.
 - o Q: Which Verilog statement did you use?
 - o Q: How did you check the select signals? (Which control flow construct?)

- Search for **LAB-TODO-STEP-2-d**
- Describe the clock and i_get synchronous behaviour to get the output signals.
 - o Q: Which Verilog statement did you use?

- Search for **LAB-TODO-STEP-2-e**
- Instantiate and connect the adder sub-module signals.

Step 3: Basic Testbench

- Open the file : *tb.v* in your favourite editor.

- Search for **LAB-TODO-STEP-3-a**
- Declare the testbench signals to be connected to the DUT
 - o Q: What type did you chose for the inputs and for the outputs? Why?

- Search for **LAB-TODO-STEP-3-b**
- Connect the signals to the DUT (you may connect the ports by name or by order).

- Search for **LAB-TODO-STEP-3-c**
- Describe a sequential testbench behaviour as you wish, do two or three assignments. Tip: use the **#n** statement to control the time flow, where n is the number of steps.
 - o Q: Why do we use an **initial** block?

- Search for **LAB-TODO-STEP-3-d**
- Examine the clock generation.

Step 4: Compile and Run simulation

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