VLSI Design Flow: RTL to GDS (NPTEL Course) Tutorial 4

1 Objective

To demonstrate the installation of simulation and the code coverage tool. Also, the working of the tool is explained with an example Verilog code.

2 Choice of Tools

- You can use the open-source tools: (ICARUS for Verilog Simulation and COVERED for computing code coverage).
- You can use any open-source tool for this lab activity.

3 Steps to install ICARUS Verilog [1]

It is expected that you already have a Linux distribution installed on your system.

- Open your Linux distribution and run the following command to clone the github repository of ICARUS.

  ```
  $ git clone https://github.com/steveicarus/iverilog.git
  ```

  once the cloning is completed, the iverilog directory is formed. You can check by doing

  ```
  $ ls
  ```

- Change directory to iverilog

  ```
  $ cd iverilog
  ```

- Build the configuration files

  ```
  $ sh autoconf.sh
  ```

  At this step if an error pops up `autoconf.sh: line 10: autoconf: command not found` or `autoconf.sh: line 16: gperf: not found` install the packages

  ```
  $ sudo apt-get update
  $ sudo apt-get install gperf
  $ sudo apt-get install autoconf
  ```

once its done, again, build the configuration files.

- Configure

  ```
  $ ./configure
  ```

  Install the required packages if the error pops up `configure: error: no acceptable C compiler found in $PATH`. It also requires `flex` and `bison` packages, so add them too.

  ```
  $ sudo apt-get install gcc g++
  $ sudo apt-get install flex
  $ sudo apt-get install bison
  ```

  Again, run the `./configure` command.

- Compile the source code

  ```
  $ make
  ```

  If the error pops up `Command 'make' not found`. Install the required package

  ```
  $ sudo apt-get install make
  ```

  and again run the compile source code command.

- Move all the application files to the appropriate system directories

  ```
  $ sudo make install
  ```

  This ends the installation process of the ICARUS tool. You can check the same by exiting the iverilog directory

  ```
  $ cd
  $ iverilog
  ```

  It will show `iverilog: no source files` and would suggest `-c` , `-y` etc.

3.1 Install GTKWave

GTKWave is an analysis tool used to perform debugging on Verilog or VHDL simulation models. Here, we will be using one of the dump file formats, Value Change Dump (VCD). To install gtkwave, execute the following commands

```
$ cd
$ sudo apt install gtkwave
```

Later, we will see how the signals are plotted using an example.

4 Steps to install COVERED Verilog Code Coverage Analyzer [2]

- Clone the gihub repository

  ```
  $ git clone https://github.com/chiphackers/covered
  ```

- Check the contents of the working directory. It should have covered directory.
- Change directory to covered
- Configure

  ```
  $ ./configure
  ```

- Next, compile the source code

  ```
  $ make
  ```

- If you encounter this error: 'Tcl_Interp′ has no member named 'result′, go to section 4.1
- If the required dependencies are not already installed, install them as shown below

  ```
  $ sudo apt update
  $ sudo apt-get install zlib1g-dev
  $ git clone https://git.savannah.gnu.org/git/libiconv.git
  $ sudo apt-get install tcl8.6
  $ sudo apt-get install tcl8.6-dev
  $ sudo apt-get install tk8.6
  $ sudo apt-get install tk8.6-dev
  $ sudo apt-get install doxygen
  ```

- Again, configure

  ```
  $ ./configure
  ```

  and compile the source code

  ```
  $ make
  ```

- For moving all the application files to the appropriate system directories, execute

  ```
  $ sudo make install
  ```

This ends the installation process of the COVERED tool.

4.1 Steps to resolve the error: 'Tcl_Interp' has no member named 'result'

If this error comes while running the make command, follow the steps given below:

```
$ cd
$ cd covered
$ cd src
$ gedit report.c
```

- In the report.c file you will see a list of #include commands. Find #include <tcl.h> and add the following command #define USE_INTERP_RESULT 1 before #include <tcl.h>.

- Your report will look like this:

  #ifdef HAVE_TCLTK
  #define USE_INTERP_RESULT 1
  #include <tcl.h>

- Save the `report.c` file.
- Run the `make` command again and proceed with the installation process as shown above.

5 Example code [3]

5.1 Design Under Verification

```
/*
* 4 bit synchronous counter
*/

module Mycounter(CLK,RST,OUT);
    input CLK, RST;
    output [3:0]OUT;
    reg [3:0]OUT;

    always @(posedge CLK)
    begin
        if (RST==1'b1)
            OUT<=4'b0000;
        else
            OUT<=OUT+1;
    end
endmodule
```

5.2 Testbench

```
/*
*  Testbench for a 4 bit synchronous counter
*/

module Testbench();
    reg Clock, Reset;
    wire [3:0]Count;

    //instantiate the DUV and make connections
    Mycounter I1(.CLK(Clock),.RST(Reset),.OUT(Count));

    //initialize the Testbench
    initial begin
        $display("Starting simulation...");
        Clock=1'b0;
        Reset=1'b1;         //reset the counter at t=0
        #100 Reset=1'b0;  //remove reset at t=100
        #2000 Reset=1'b1;  //remove reset at t=2100
        #400 $finish;   //end the simulation after t=2500
    end

    //generate stimulus (in this case clock signal)
    always #50 Clock=~Clock;//clock period=100

    //monitor the response and save it in a file

    initial begin
        $dumpfile("count.vcd"); // specifies the VCD file
        $dumpvars; //dump all the variables
        $monitor("%d,%b,%b,%d",$time,Clock,Reset,Count);
    end

endmodule
```

6 Simulation and viewing the output waveform

- Launch the Linux distribution and make a directory

  ```
  $ mkdir icarus_codes
  ```

  where icarus_codes is the name of the directory.

- Change directory to icarus_codes. For simulation, you require a Verilog code for the implemented functionality and a test bench in .v format. Include those in the working directory. Let's say my verilog code is named Mycounter.v and the test bench is Testbench.v. To simulate:

  ```
  $ iverilog -o Mycounter Mycounter.v Testbench.v
  $ vvp Mycounter
  ```

  A dump file, which I have named as count.vcd in the test bench, is created, and you can see the output on the terminal as well. You can also view the output in the GTKWave

  ```
  $  gtkwave count.vcd
  ```

  This command launches the GTKWave analyzer application. In the left panel, expand the Testbench and click on the subfolder. It will expand to show input Clock, Reset and output waveform OUT[3:0]. Drag those to the Signals panel and analyze the results.

7 Generate the code coverage report

To estimate the percentage of RTL design tested by the test bench, COVERED Verilog Code Coverage Analyzer tool is used.

- Generate the code coverage report in the same directory i.e., icarus_codes by executing the following command

  ```
  $ covered score -t Testbench -v Testbench.v -v Mycounter.v
     -vcd count.vcd -o Mycounter.cdd
  ```

- To view the coverage report, execute

  ```
  $ covered report -d v Mycounter.cdd
  ```

  The coverage report is displayed in the terminal.

References

[1] Installation steps for The ICARUS Verilog Compilation System. [Online]. Available: https://github.com/steveicarus/iverilog

[2] Installation steps for Covered - Verilog Code Coverage Analyzer. [Online]. Available: https://github.com/chiphackers/covered/blob/master/INSTALL

[3] S. Saurabh, *Introduction to VLSI Design Flow.* Cambridge: Cambridge University Press, 2023.