# Project-2: Lab Manual

## Section-I: Initial Setup

1. The tools setup and the necessary files have been set up inside docker. This can be thought of as a container, much simpler than a virtual machine. Invoke docker by typing the following on the terminal.

**docker run --privileged -e DISPLAY=$DISPLAY -v /tmp/.X11-unix/:/tmp/.X11-unix/ -it docker_ee619:labv2**

Important Note

Any changes you make inside the docker container will be lost once you exit docker unlike in a virtual machine where the changes get saved. You will need to commit the changes that you wish to save using the following commands. In another terminal, type

**docker ps -a**

This will list down all the active dockers. Find the container ID (alpha numerical code) for your container. For e.g. say the container ID is 896c03d23c77 and your team name is Team1. Then type the following to save the changes.

**docker commit 896c03d23c77 docker_ee619:Team1**

To check the saved images, run

**docker images**

If you accidently exit the docker, or are starting session 2, you can load the saved docker image by typing,

**docker run --privileged -e DISPLAY=$DISPLAY -v /tmp/.X11-unix/:/tmp/.X11-unix/ -it docker_ee619:Team1**

Please save your container periodically.

2. Go to the location /home/ee619 by typing the following command.
   **cd /home/ee619**
   (Please use the command **cd** whenever you want to go into a folder. Think of this as double-clicking the folder in Windows. If you want to see the contents of current directory, type **ls** in the terminal.)

3. **source set_var**
   (set_var file contains commands to set all the environment variables required to run the tools. You are executing the contents of the file with this statement.)

4. **iverilog -help**
   This should display the following, to indicate that icarus Verilog is configured correctly.
   **Usage: iverilog [-EiRSuvV] [-B base] [-c cmdfile|-f cmdfile]**
   **[-g1995|-g2001|-g2005|-g2005-sv|-g2009|-g2012] [-g<feature>]**
   **[-D macro[=defn]] [-I includedir] [-L moduledir]**
   **[-M [mode=]depfile] [-m module]**
   **[-N file] [-o filename] [-p flag=value]**
   **[-s topmodule] [-t target] [-T min|typ|max]**
   **[-W class] [-y dir] [-Y suf] [-l file] source_file(s)**
5. **yosys -help**
   This should display a long list of options that can be used with the tool yosys. The list will begin as follows.
   **Yosys Open SYnthesis Suite**
   **Usage:**
   **/home/ee619/oss-cad-suite/libexec/yosys [OPTION...] [<infile> [..]]**

   **operation options:**
   **-b, --backend <backend> use <backend> for the output file specified on the command line**

**Inform your TA that your tool set up is fine.**

# Section-II: A Simple Counter Design

## Verifying the behavioral code

- Please go to the location /home/ee619/firstcounter by typing
  **cd /home/ee619/designs/firstcounter**
  We will keep all design files associated with your first counter file in this directory.
- Observe the contents of this directory by typing
  **ls**
  There will be two directories named **presynthesis** and **synthesis**. The initial behavioral level codes and the testbenches will be kept in the **presynthesis** folder. All files related to synthesis can be kept in the **synthesis** folder. (An organization to make our lives easier.)
- **cd presynthesis**
- **ls**
- **gedit counter.v &**
  This will open the **counter.v** file in a text editor. This described a very simple counter. Observe the code.
- **gedit counter_tb.v**
  This will open the testbench file. Observe the code.
- Run the simulation for this counter and verify the code using Icarus Verilog.
  **iverilog -o mydesign counter_tb.v counter.v**
  **vvp mydesign**
  **gtkwave test.vcd &**

**Show the results to the TA.**

## Observing a standard cell library

- Go to the folder **synthesis**
  **cd ../synthesis**
  ( .. refers to one level up from your current directory.)

- We will be using the nangate 45nm open source library for synthesizing our counter. The technology library file with Verilog description of the standard cell has already been copied into this folder. Please open the **cells.v** file and observe the contents. DO NOT make any changes to this file.
  **gedit cells.v &**
  The above command opens the file.

- You can also open the library file with timing and power information about all these cells.

**gedit /home/ee619/OpenROAD-flow-scripts/flow/platforms/nangate45/lib/NangateOpenCellLibrary_typical.lib**

Compare AND2_X1 (2-input AND gate with minimum drive strength) in both the files. Can you see how the delays and the power consumption have been characterized for the cell? How many AND2 gates are there in the library? (Feel free to google if you don't understand a particular terminology in the *.lib file.

**Inform your TA that you have completed this step.**

## Synthesizing the counter

We need to implement the counter in terms of the cells available in the **cells.v** file. For this, we will use the tool **yosys**.

- Ensure that you are in the **synthesis** folder. If in doubt, run the following command.
  **cd /home/ee619/designs/firstcounter/synthesis**
- The commands for synthesizing the code are available in **synthesis_script.tcl**. Open this file using gedit.
  **gedit synthesis_script.tcl &**
- Invoke yosys using the following command.
  **yosys**
- Run the commands in **synthesis_script.tcl** (tcl is pronounced as tickle) file one by one. The tcl file has been commented to help you understand the stages better.
  If you need to understand a command better, you can execute **help <command>** in the yosys terminal. E.g., say you want to understand more about the **read_verilog** command. Run the following in the terminal. **help read_verilog**.
- Observe the design at all stages using the **show** command. Can you spot the **AND2_X1** cell in the final design?
- Once synthesis is completed, exit from yosys.
  **exit**
- Open the counter_syn.v file and observe the contents.
  **gedit counter_syn.v &**
- Now we need to verify that this netlist is the counter we designed for.

## Verifying the synthesized netlist

To verify the design, we need the synthesized netlist, the testbench and the Verilog description of the standard cells. Run the following commands to run the simulation. (The first two lines are a single command!)

**iverilog -o mydesign -g2012 -D POST_SYNTHESIS ../presynthesis/counter_tb.v counter_syn.v cells.v**
**vvp mydesign**
**gtkwave test.vcd**

Do the waveforms match the waveforms we got before synthesis? Congrats! The counter is synthesized correctly.

**Show the simulation results to your TA.**

Tips

Tip1: When you have multiple commands to run in the terminal, most of the times you can put everything in a text file and source it. Check out the contents in the file **cmd_runsim** in the **synthesis** folder. Typing **source cmd_runsim** will execute all these statements in one go.

Tip 2: It can get cumbersome to type the commands in yosys one by one. The synthesis_script.tcl can be sourced in one go while invoking yosys by typing the following.
**yosys -s synthesis_script.tcl**
If the show command is causing issues, simply comment out or remove those and source the script again. For example. The synthesis_scrip_woShow.tcl, is a copy of the same script without the **show** command. You will be able to run this in one go as the following.
**yosys -s synthesis_script.tcl**

# Section-III: Synthesizing the backend (Project-1)

- Download the Verilog files to this PC from mail using a web browser.
- Now we need to push these files into the docker container. For this, in your docker container first make the directory structure similar to the **firstcounter** directory by following these steps.
  **cd /home/ee619/designs**
  **mkdir backend**
  **cd backend**
  **mkdir presynthesis**
  **mkdir synthesis**

- Identify the docker container ID by running the following command in a terminal where docker is not running.
  **docker ps -a**
  Say the container ID is 896c03d23c77. Assume the Verilog file to be copied is backend.v. The following command will copy it into /home/ee619/designs/backend/presynthesis. (Single line command.)

**docker cp <path to Verilog file>/backend.v  896c03d23c77:/home/ee619/designs/backend/presynthesis/**

- Confirm that the files were copied into the docker container correctly.

**Inform your TA that you were able to copy the files successfully into the docker container.**

- Now you can proceed with synthesizing your backend code. You may have to edit your code to make it synthesizable. Please make sure that your save your code and commit the changes in the docker container periodically so as to not lose the progress. Do your best!

**Show the simulation results to your TA once you have got the synthesized code working.**