CENG 4120 Computer-Aided Design of Very Large Scale Integrated Circuits

Homework 1 (Part B)
Due Date: Feb 17, 2025 11:59pm

1 Setup Instruction

1. Setup and connect to the CSE VPN following the guideline here. OpenVPN is recommended.
2. For Windows users, MobaXterm is the suggested SSH client. Connect to server linux5-16 (e.g., linux10) using the following command,

```
$ ssh username@linux10.cse.cuhk.edu.hk
```

3. For Mac OS users, use Terminal provided by Mac OS to connect to server linux5-16 (e.g., linux10) using the following command,

```
$ ssh username@linux10.cse.cuhk.edu.hk
```

4. Upload **25hw1.zip** to the server (e.g., linux10). If you are using MobaXterm, just drag the file from local to remote. If you are using Terminal, open a new Terminal window (local) and upload the file using the following command,

```
$ scp -r /local/location/25hw1.zip
username@linux10.cse.cuhk.edu.hk:/remote/location/
```

Then, extract the files on the server and run the setup script.

```
$ unzip 25hw1.zip
$ source 25hw1/setup.sh
```

2 Standard Cell Library

A standard-cell library is a collection of combinational and sequential logic gates. The standard cells are realized as fixed-height, variable-width cells, which enables them to be placed in rows of the same height. In this homework, we will be using FreePDK 45nm Standard Cell Library. Below lists some of the files needed in this homework.

```
# 25hw1/FreePDK45

gscl45nm.lib   # abstract logical, timing, power view of each cell
gscl45nm.db    # binary compiled version of .lib file
gscl45nm.lef   # abstract physical view of layers, vias and cells
```

Now, let's first take a look at the content of the files and answer the following questions.

1. Open file **gscl45nm.lib** using a text editor and find cell **MUX2X1**.
   (a) What are the input pins of the cell?
   (b) What is the output pin of the cell?
   (c) What is the Boolean function of the cell?
   (d) What is the cell leakage power?
2. Open file **gscl45nm.lef** using a text editor.
   (a) What are the width and height of cell **NAND2X1** in micron?
   (b) What are the width and height of cell **MUX2X1** in micron?
   (c) What are the routing directions of layer metal1 and metal2?

3 Synthesis Using Synopsys Design Compiler

Next we will use the standard cell library to synthesize a simple 4-bit full adder. The verilog file (**cla4.v**) is already provided. Create a new directory and move **cla4.v** into it. Launch Synopsys Design Compiler. Design Compiler User Guide can be found here.

```
$ mkdir -p synopsys
$ cd synopsys
$ mv /path/to/cla4.v .
$ dc_shell-t   # launch Design Compiler
```

Let's start by setting some variables to specify the standard cell library to use.

```
dc_shell> set target_library "/path/to/gscl45nm.db"
dc_shell> set link_library "/path/to/gscl45nm.db"
```

Next, use the **analyze** command to read the Verilog file into an intermediate internal representation, and use the **elaborate** command to recursively resolves all module references from the top-level module. The top-level module can be think of as the main function of a Verilog program and is **cla4** in our Verilog.

```
dc_shell> analyze -f verilog cla4.v
dc_shell> elaborate cla4
```

We will also create a clock constraint. Synopsys Design Compiler will not optimize the design to run as fast as possible. Instead, it will try to minimize area and power while meeting the clock constraint. Suppose we set the target cycle time to be 0.15 nanoseconds, the design will run at 6.67 GHz if the synthesis is successful. Note that **clk** is the clock signal in our Verilog file.

```
dc_shell> create_clock clk -name clock1 -period 0.15
```

The **set_max_area** command specifies an area target. Defining a maximum area directs Design Compiler to optimize the design for area after timing optimization is complete.

```
dc_shell> set_max_area 1000
```

Start the synthesis by

```
dc_shell> compile
```

After the synthesis, write the resulting gate-level netlist in **.v** format as well as the constraint information we gave to Design Compiler in **.sdc** format.

```
dc_shell> write -format verilog -hierarchy -output cla4_synth.v
dc_shell> write_sdc -nosplit cla4_synth.sdc
```

Lastly, use the **report_timing** command to show the critical path and the *slack*. The *slack* is the difference between the required arrival time and the actual arrival time. If the *slack* is negative, it means the clock constraint cannot be satisfied, and we might want to try a longer clock cycle. Use **report_area** command to display area information and statistics for the current design or instance.

```
dc_shell> report_timing
dc_shell> report_area
```

Exit Design Compiler.

```
dc_shell> exit
```

Note that we can actually put all the command we used in Synopsys Design Compiler into a **.tcl** file and run all of them sequentially to avoid inputting the commands one by one.

```
$ dc_shell -t -f dc.tcl    # run a sequence of commands
```

Now answer the following questions,

1. Follow the above instructions to do the synthesis. What is the slack and total cell area of the design?
2. To minimize area, set the max area as 300 and replace **compile** with **compile -area_effort high**. What is the total cell area now?
3. Try different clock periods. What is the shortest clock period without a negative slack? What frequency (GHz) does the clock period correspond to? What is the cell area of this design?
4. Can we achieve smaller area in 3 by setting **set_max_area** as 200, 100, or even 0? What does the command **set_max_area 0** mean? (see User Guide)
5. Attach the **.tcl** you used to achieve the clock period in 3.