EE 577A Spring 2020 – VLSI System Design

Professor: Pierluigi Nuzzo   TA/Mentor: Yinghua Hu, Subhajit Dutta Chowdhury

Lab 1 Part 3   Score: ____/100

Student ID: ______________________   Name: ______________________
Assigned: February 11st
Due: February 21th at 11:59pm (Submit via D2L). No late submissions.

In Lab1 Part 3 you can discuss your ideas, algorithms, etc. with your colleagues. However, you should submit your own implementation. You are NOT allowed to copy your code or design artifacts from others. This means you should not discuss or present with others your actual code or design artifact. You can refer to online resources but be careful about citing them.

The objective of this lab is for you to review the tools you used in EE477 and learn the basic structure of adders and multipliers. You do not need to maximize the speed of your design at this time, but your design should be area efficient. Another goal of this lab is to help you start learning Perl or Python, which are popular scripting languages used in simulation-based verification steps. Scripting will be an important part of your final project and may also appear in the following lab assignments.

Section 1: 1-bit Full-Adder

- Design a 1-bit Full-Adder in **layout**, using the structure shown in Figure 1.
  - The Full-Adder in Figure 1 has three inputs (A, B, C) and two outputs (C_{out}, S), where $\mathrm{S} = \mathrm{A} \oplus \mathrm{B} \oplus \mathrm{C}$ and $C_{\text{out}} = \mathrm{AB} + \mathrm{AC} + \mathrm{BC}$.

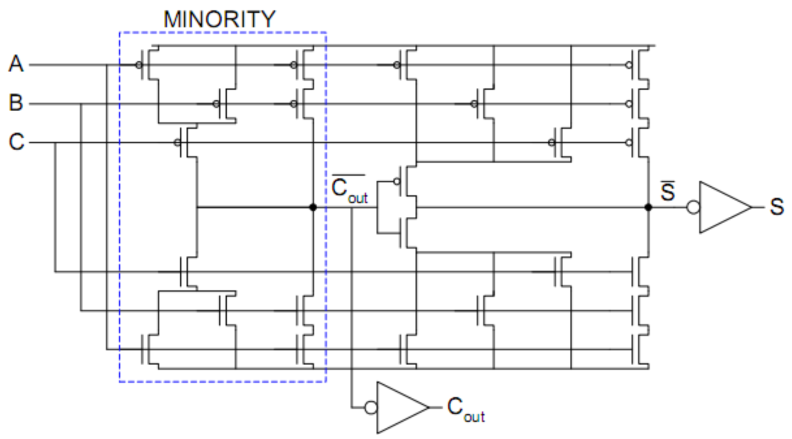

Figure 1: Structure of the Full-Adder

- General instructions:
  - You are NOT required to do proper sizing this time.
  - Use $\mathrm{V_{DD}} = 1\ \mathrm{V}$.
  - Use rise/fall transition time = 5 ps.
  - Perform functional test of all the possible combinations of inputs (A, B, C) and report the output for the 1-bit Full Adder

- Layout Guidelines:
  1) You need to create the cell template for your design from library 'gsclib045_tech'. You do not need change height here, because the height of the cell template should be the same as the height of the cells in the 'gpdk045' library. Just change the width of this cell template according to your design.
  2) You can use the schematic and layout of basic cells from library 'gpdk045', such as NAND, NOR or Inverter, etc.
  3) Your layout design should pass all DRC rules and LVS check.
  4) For the internal routing you could use Poly or Metal (you may use any metal layer up to Metal2). We recommend that you use Metal1 for horizontal and Metal2 for vertical wires. If you cannot make it by only using Metal1 and Metal2 then start using higher metal layers. However, if you use Metal3 or layers above, some points will be deducted in this part. Try to use as few metal layers as possible so routing would be more convenient in your future assignments that may use your Lab1 layout.

Section 2: 5-bit 2's Complement Multiplier Design

- Design a 5-bit 2's Complement Multiplier (the inputs are 5-bit 2's complement numbers and the outputs are 10-bit 2's complement number) in **layout**. Figure 2 shows the structure of a 4-bit 2's Complement Multiplier. Build a similar structure for a 5-bit 2's Complement Multiplier.

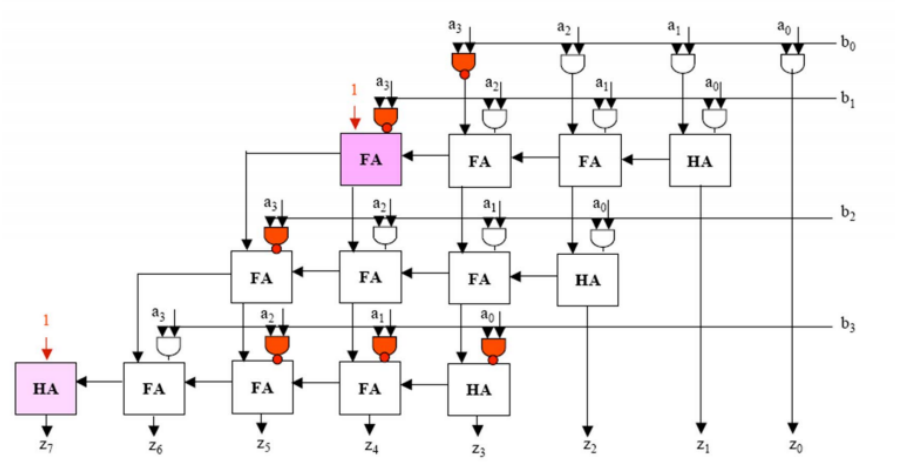

Figure 2: Example of 4-bit 2's Complement Multiplier

- General instructions:
  a. Use the "1-bit Full-Adder" of Part1.
  b. You can either use Full-Adders to build Half-Adders or you can re-design Half-Adders.
  c. Use $V_{DD} = 1\ \mathrm{V}$.
  d. Use rise/fall transition time = 5 ps.
  e. All inputs and outputs of the multiplier should be registered with positive-edge triggered master-slave flip-flops.
  f. You are NOT required to measure the worst-case delay, but clock period of the FFs should be large enough to eliminate setup-time violation.
  g. You can use the basic cells from library 'gpdf045'.

  Layout guidelines are the same as those in part 1 except that you can use up to Metal4 for routing. However, if you use Metal5 or layers above, some points will be deducted in this part.
- Perform functional tests using the input patterns generated by Perl or Python:
  a. Complete the given Perl script which randomly generates 6 test patterns and their corresponding golden results in both decimal and binary format. Even if we provide a Perl script example, you can solve this problem by either using Perl or Python.
  b. Fill in the table below (a sample pattern is given) after running the Perl/Python script.

c. Apply the patterns in Cadence and compare the waveforms with the results generated by Perl/Python.

op1(dec)	op2(dec)	result(dec)	op1(bin)	op2(bin)	result(bin)
−15	6	−90	10001	00110	1110100110
…					

Report Checklist:

Part 1: 1-bit Full-Adder

(1) Layout for 1-bit Full-Adder.
(2) Screenshots showing the width, height of a cell, and total area.
(3) DRC check and LVS match Screenshot or report for 1-bit Full-Adder.
(4) Functional test waveforms and results **on the layout** of 1-bit Full-Adder for all input combinations.

Part 2: 5-bit 2's Complement Multiplier

(5) Layout for 5-bit 2's Complement Multiplier.
(6) Screenshots showing the width, height of a cell, and total area.
(7) DRC check and LVS match Screenshot or report for 5-bit 2's Complement Multiplier.
(8) Functional test waveforms and results **on the layout** of the 5-bit 2's Complement Multiplier.

Submission Guidelines

- **Briefly explain your work.**
- **Name your pdf file based on the following format:**
  "yourfirstname_yourlastname_studentID_Lab1Part3.pdf"