EE 577A Spring 2020 – VLSI System Design

Pierluigi Nuzzo **TAs: Subhajit Dutta Chowdhury and Yinghua Hu**

Lab 1 Part 2 Score: ____/100

Student ID: _____ Name: _____

Assigned: January 31st

Due: Friday February 11th at 11:59pm (Submit via the provided DEN link). No late submissions.

In Lab1 Part 2 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 & 16 bit Carry Skip Adder

- Design a 1-bit Full Adder in schematic 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 $S = \text{A} \oplus \text{B} \oplus \text{C}$ and $C_{\text{out}} = \text{AB} + \text{AC} + \text{BC}$.
  - After designing a 1-bit Full Adder, use it to design a 16-bit Carry-Skip Adder as shown in Figure 2. At this point there is no need to optimize the design of the Propagate signal block.

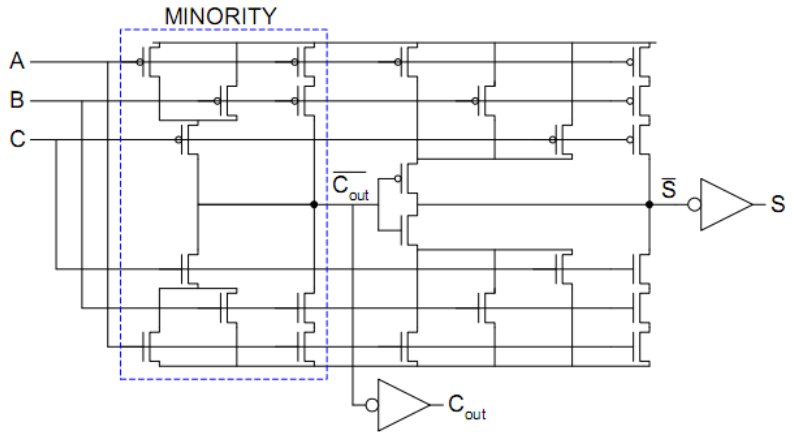

Figure 1: Structure of the Full Adder

- General instructions:
  - Characterize an inverter to find the $\beta = \frac{\mu_n}{\mu_p}$ for this technology node and report it.
  - You are NOT required to do proper sizing this time.
  - Use $V_{DD} = 1\ \text{V}$.
  - 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.

- Perform functional test of 10 different combinations of inputs (A_i, B_i, C) and report the output for the 16-bit Carry-Skip Adder.

- Perform functional tests using the input patterns generated by Perl or Python:
  - Create a Python/Perl script which randomly generates 10 test patterns and their corresponding golden results in both decimal and binary format.
  - Fill in the table below (a sample pattern is given) after running the Perl/Python script.
  - 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)
10	20	30	0000000000001010	0000000000010100	0000000000011110
…					

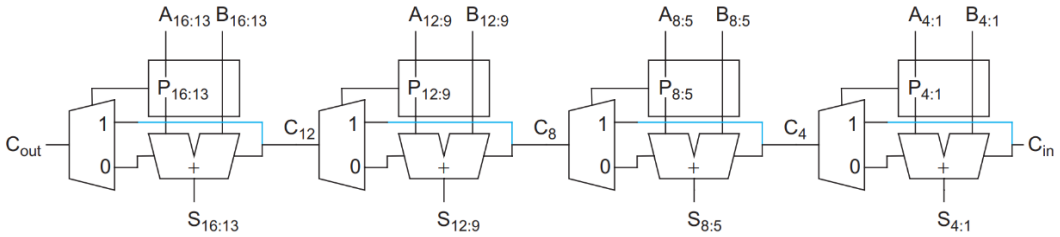

Figure 2: Structure of a 16-bit Carry Skip Adder

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 output is a 10-bit 2's complement number) in **schematic**. 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.

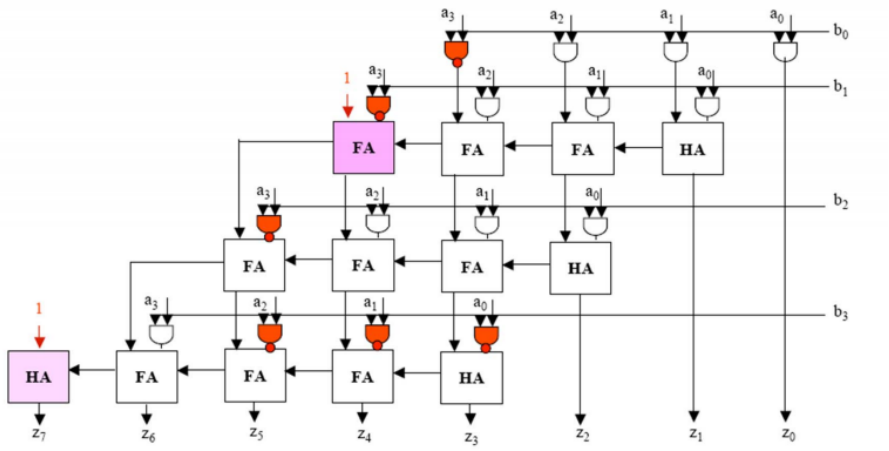

Figure 2: Example of 4-bit 2's Complement Multiplier

- General instructions:
  a. Use "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 $V_{DD} = 1$ 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.

- Perform functional tests using the input patterns generated by Perl or Python:
  a. Create a Python/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 and 16-bit CSA

(1) Schematic and symbol for 1-bit Full Adder.
(2) Functional test waveform and result on the schematic of 1-bit Full Adder for all input combinations.
(3) Schematic and symbol for 16-bit Carry Skip Adder.
(4) Identify the critical path of the Carry Skip adder design. Justify your answer with reasoning.
(5) Completed Perl/Python Script and completed table of test patterns generated by Perl/Python.
(6) Functional test waveform and result on the schematic of 16-bit Adder for the worst-case delay input combinations.

Part 2: 5-bit 2's Complement Multiplier

(7) Schematic for 5-bit 2's Complement Multiplier.
(8) Completed Perl/Python Script and completed table of test patterns generated by Perl/Python.
(9) Functional test waveforms and results on the schematic 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_Lab1Part2