**Homework 1 (Due: Sep 11)**

**Instructions**

* Complete the following questions to the best of your ability.
* Answers should be clear, concise, and justified with work.
* Hand-drawn circuits will not be accepted due to confusion they might cause.
  + Use software, such as Logisim, to draw them.
* Please write your name and NetID on the hardcopy of your solution, and bring it to the lecture on the due day. Please submit your solution before the lecture starts.

# **Collaboration**

* Professor/TAs
  + You **may** discuss any content with any professor or TA.
* Students
  + You **may** discuss high-level concepts, techniques, jargon, keywords, related problems, or course content that is relevant to the material.
  + You **may not** discuss particular solutions to any of the questions.
  + If you have substantial conversations with any students, please note their name and, if you feel it necessary, the extent of your collaboration.
* Outside Sources (Internet, books)
  + You **may** use external references for any course content.
  + You **may** use an external tool to verify your solutions where appropriate, but not to generate solutions to any questions.
  + List any outside sources that you use. Formal citations are not necessary; links are fine.

# **Question 1 (30 points)**

Follow the steps below to create the CMOS representation of a two input OR gate.

1. Fill out the truth table for a two-input OR gate.

|  |  |  |
| --- | --- | --- |
| **X** | **Y** | **Output = X + Y** |
| 0 | 0 |  |
| 0 | 1 |  |
| 1 | 0 |  |
| 1 | 1 |  |

1. Extract the simplified logical expression for when output = 1 from your truth table. Show your work.

Output =

1. Repeat step B for when the output = 0.

Output =

1. Show that both expressions of parts B and C are equal.
2. Using your expressions from parts B and C, create the CMOS circuit of a two-input OR gate. You are allowed to have inverted inputs on transistor gates (e.g., x1’). You should not end up with the design of a NOR gate followed by a NOT gate. Show the expressions of the PUN and PDN. Remember that you should use software to draw the circuit diagram.

PUN expression:

PDN expression:

Circuit:

# **Question 2 (20 points)**

Convert the numbers below into the requested number formats. Show all work for full credit.

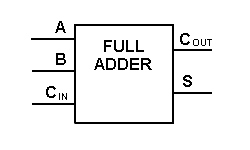
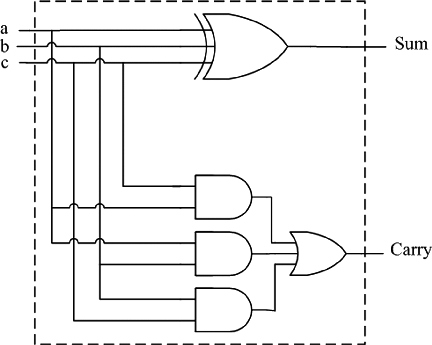
1. Convert the decimal value 7.5625 to IEEE Single Precision Floating Point format.

|  |  |  |
| --- | --- | --- |
| **1 Sign Bit** | **8 Biased Exponent Bits** | **23 bit Mantissa** |
|  |  |  |

1. Write your answer from part A in hexadecimal.

# 

# **Question 3 (50 points)**



Consider the **full adder block**, as illustrated above (**left, gate level diagram; right, logic block**). This block can be used to construct a number of adders, including the ripple carry adder (RCA) and carry select adder (CSA) we discussed in class. Each **full adder** block requires a **2 gate delay** to generate a **carry out**. Also, assume a **mux** to require a **2 gate delay** to generate an **output.** Given this information, consider building a **4-bit RCA and a 4-bit CSA**, and the **gate delay differences** between the two.

**Follow the steps below.** Assume adder inputs of A and B (4-bit wires) and **Cin (1-bit, constant 0 wire for carry in**) and outputs of S (4-bit wire for sum) and Cout (1-bit wire for carry out). You may use these inputs and outputs, full adder blocks (like the image on the right), constants (0/GND and 1/VCC), and muxes.

1. Create the circuit diagram for a 4-bit RCA. Remember that you should use software to draw the diagram.
2. Given your RCA diagram and the above information about full adder blocks - how many gate delays does your 4-bit adder require to generate the carry out? Justify your response with work.
3. Repeat step A for a CSA.
4. Repeat step B for your CSA implementation.
5. Which of your adders has a lower delay?
6. Which of your adders requires less area (on a chip/board)?