Carlos Gross-Martinez

CDA3201C

Logic Design Course

Lab 2: 1 Bit Half Adder, 1 Bit Full Adder, 4 Bit Adder, 5 Bit Adder

Cout = AB – Half Bit Adder

Sum = A’B + AB’ – Half Bit Adder

Cout = AB + AC + BC – Full Bit Adder

Sum = ABC + AB’C’ + A’BC’ + A’B’C – Full Bit Adder

Truth Table and K-map
Cout = AB, Sum = A'B + AB'

• CDA3201 • Intro to Logic Design •

Lab Assignment 2

Name: Carlos Gross-Martinez

Grade: /100

2) [100 Petrie] BINARY ARITHMETIC – 1 bit Half Adder, 1 bit Full Adder, 4 bit Adder, 5 bit Adder

Arithmetic Operations include **ADDITION** (+) and **MULTIPLICATION** (•). **Boolean Operations** include **OR** (+) and **AND** (•). Although the symbols are the same, the operations and results are different. In Lab 1 the circuits performed Boolean Operations. In this lab the circuits will perform the Arithmetic Operation of Addition of Binary Numbers.

Add the **decimal numbers***(base 10) Below by hand and show carry:

Add the **binary numbers** (base 2) below by hand and show carry:

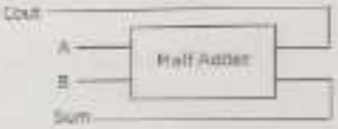
Ex: 111

C_{out}														
A	2	5	9	8	71	2598	101	0	0	1	1	110	1101	
+B	+3	+6	+7	+4	+29	+3674	+011	+0	+1	+0	+1	+011	+0111	
C_{out} Sum	5	11	16	12	100	6272	1000	0	1	1	10	1001	10100	

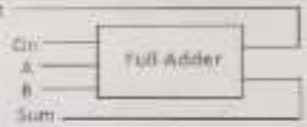
Decimal	Binary
0	0
1	1
2	10
3	11
4	100

Note: Each variable is one digit, the answer takes up 2 variables C_{out} and Sum. If you have more than 1 column of numbers, addition is done from right to left. For the right most (least significant) column you only add 2 numbers, but for the rest of the columns you add the 3 numbers shown in the column plus the carry from the column to its right. So to build an arithmetic unit that adds we will need two types of components, one that adds 2 binary numbers, and one that adds 3:

- **Half Adder** is used to add the least significant (right-most) column. It has 2 inputs: *A*, and *B*; and 2 outputs (one for each digit in the answer). The least significant digit of the answer is called *Sum* (written below the column), and the most significant digit of the answer is called "carry" (C_{out} written above the next column on left).



- **Full Adder** to add each column other than the least significant column. It has 3 inputs: the 2 numbers in the column: *A*, *B*, plus the carry C_{in} from the column to its right, now called C_{in}. There are 2 outputs (one for each digit of the answer): the least significant digit is the *Sum*, while the most significant is the carry out to the next column to its left, which we call C_{out}.



2.a.1 [5 Petrie] Complete the Truth Table on the right for a **Half Adder** to add two binary digits (bits), **A** and **B**, to get a two bit answer, the **Sum** is the least significant bit, C_{out} is the most significant bit or "carry". Write the **Minterm #** and the product associated with each row, then add A and B in decimal and binary.

Minterm Product	#	Inputs A	Inputs B	Calculate A plus B in decimal	Outputs A plus B in binary C_{out}	Outputs A plus B in binary Sum
A'B'	0	0	0	0	0	0
A'B	1	0	1	1	0	1
AB'	2	1	0	1	0	1
AB	3	1	1	2	1	0

2.a.2 [5 Petrie] Write the Boolean expression as a **Sum of Minterms** and a **Canonical Sum of Products**, fo (A, B):

Sum = f (A, B) = ∑ m (1, 2 **),** **Sumα = fα (A, B) =** A'B + AB'

C_{out} = f (A, B) = ∑ m (3 **),** **C_{out}α = fα (A, B) =** AB

2.a.3 [5 Petrie] Number each cell (in upper right corner) of the Karnaugh Map (K-Map), enter the corresponding value of the output from the Truth Table, and group to show simplification.

Sum

A \ B	0	1
0	0	1
1	1	0

C_{out}

A \ B	0	1
0	0	0
1	0	1

2.a.4 [5 Petrie] From the KMaps find **Simplest Sum of Products**. Copy the answer on top of next page.

Sum = A'B + AB' **C_{out} =** AB

Logic Drawing Regular
Cout = AB, Sum = A'B + AB'

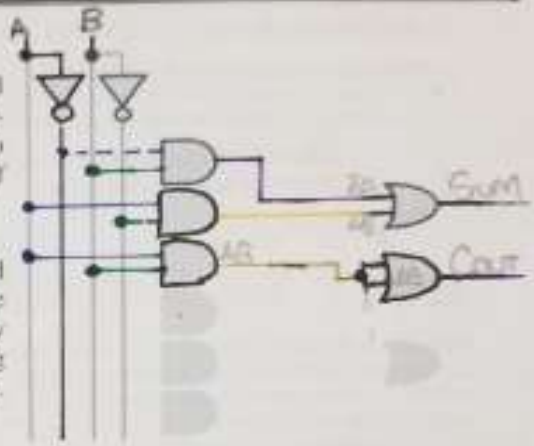
• CDA3201 • Intro to Logic Design •

Lab Assignment **2**

Name: Carlos Gracia-Martinez

Grade: /100

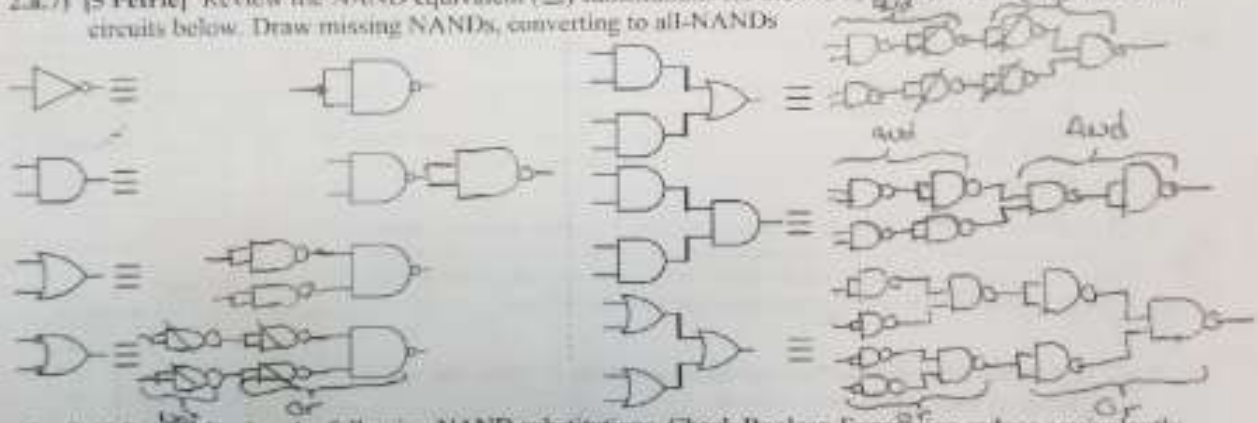
2.a.4 **Sum =** $A'B + AB'$ **Cout =** AB



2.a.5) [10 Petrie] Design a circuit for the optimized **Cout** and **Sum** as a 2-input (A, B) 2-output (**Cout, Sum**) NOT-AND-OR network. Use the shaded gates on the right to help you trace the gates you need. Label all the outputs of the AND gates.

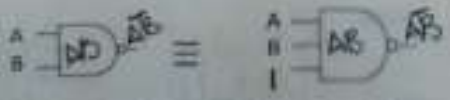
2.a.6) [5 Petrie] Create a project in **Quartus** named **Lab2HalfAdder_Petrie_YourName**. Verify that the NOT AND OR Circuits for **Sum1** and **Cout1** works by drawing the schematic in a .bdf, compiling and simulating it. Verify that the .vwf output matches truth table in 2.a.1. Capture the schematic for your portfolio file.

2.a.7) [5 Petrie] Review the NAND equivalent (≡) substitutions for the NOT, AND, OR, negated OR, and circuits below. Draw missing NANDs, converting to all-NANDs

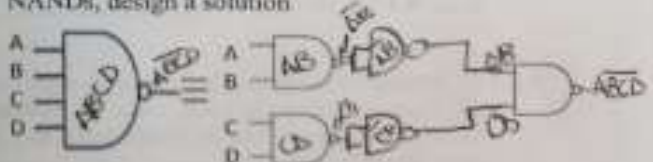


2.a.8) [5 Petrie] Design the following NAND substitutions. Check Boolean Expression reduce equivalently

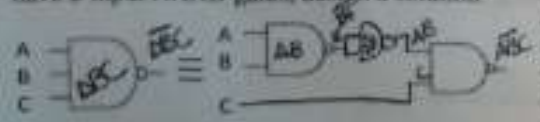
You only have 3-input NAND gates and you need a 2-input NAND gate, what do you connect to 3rd input to gives the same result as the 2-input NAND?



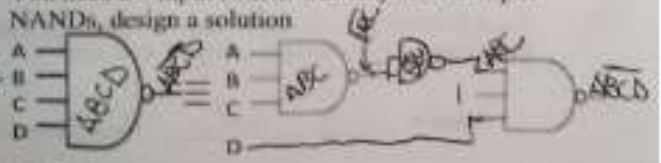
You need a 4 input NAND but only have 2-input NANDs, design a solution



You need a 3-input NAND gate and you only have 2-input NAND gates, design a solution



You need a 4 input NAND but only have 3-input NANDs, design a solution



Logic Drawing NAND Gates and Drawing Using Logic Chips
Cout = AB, Sum = A'B + AB'

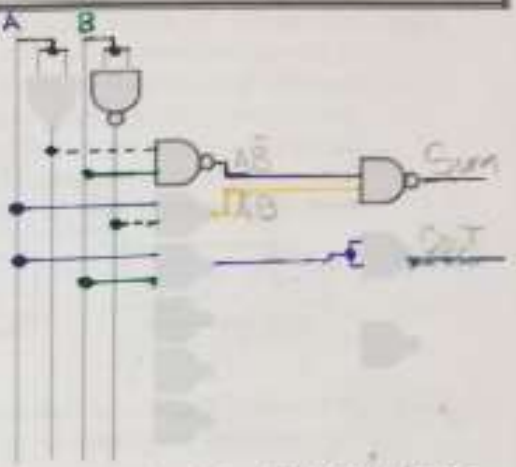
• CDA3201 • Intro to Logic Design •

Lab Assignment 2

Name: Carlos Gross-Martinez

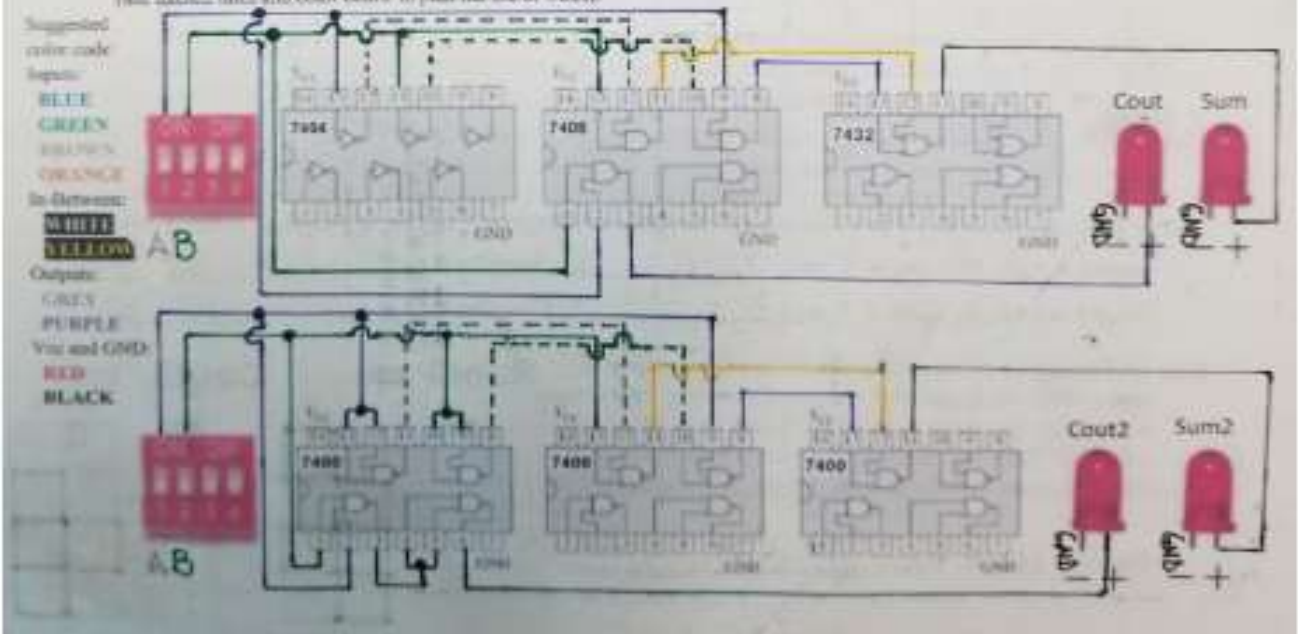
Grade: /20

2.a.9) [5 Petrie] Convert the simplified NOT-AND-OR circuit in 2.a.5 to an all-NAND gates. Make sure to remove these pair of extra NANDs to make the **Simplest all NAND circuit**. Label outputs **Cout2**, **Sum2**. Note: when you do the substitution you sometimes you end up with two "NOTs" in a row along the same wire. This "NOT" pair can be removed because they cancel: (A')' = A. You may want to draw it on scratch paper before translating your design to the diagram on the right.



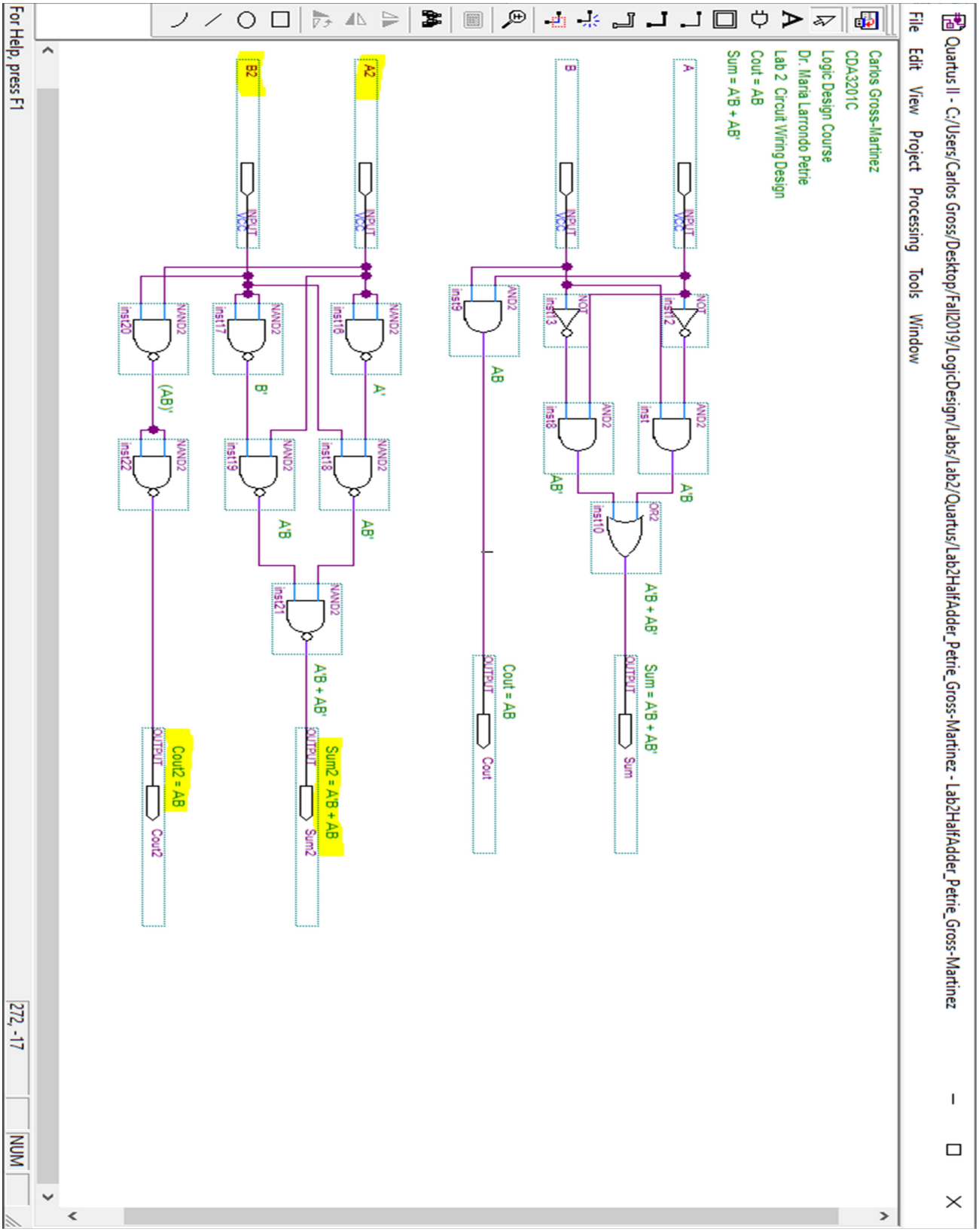
2.a.10) [5 Petrie] Open the 2.a.6 Quartus project. Add to the .bdf file the schematics of the **Cout2** and **Sum2** of all-NAND circuits. Compile, simulate, and verify 2.a.5 and 2.a.7 circuits are equivalent. Capture the all-NAND schematic and .vwf results for portfolio.

2.a.11) [10 Petrie] Once verified equivalent, plan your wiring below of both circuits but **DO NOT WIRE** on the breadboard. Label all inputs and outputs of the gates used. Note: wires from gray/green tube in kit contains four twisted pairs of wires: BROWN and Brown/White, ORANGE and Orange/White, GREEN and Green/White, BLUE and Blue/White. These can be used for up to 4 inputs that can be noted. Note: you have other colors that have no corresponding white striped wire: WHITE, GREY, PURPLE, YELLOW, besides these: RED (used for Vcc (Power)), and **BLACK** (used for Ground (avoid using these in your circuit except for these uses)). Use a different colored solid wire for each input, and corresponding color/White for corresponding NOT (see dashed lines and color below) to plan the use of wires).



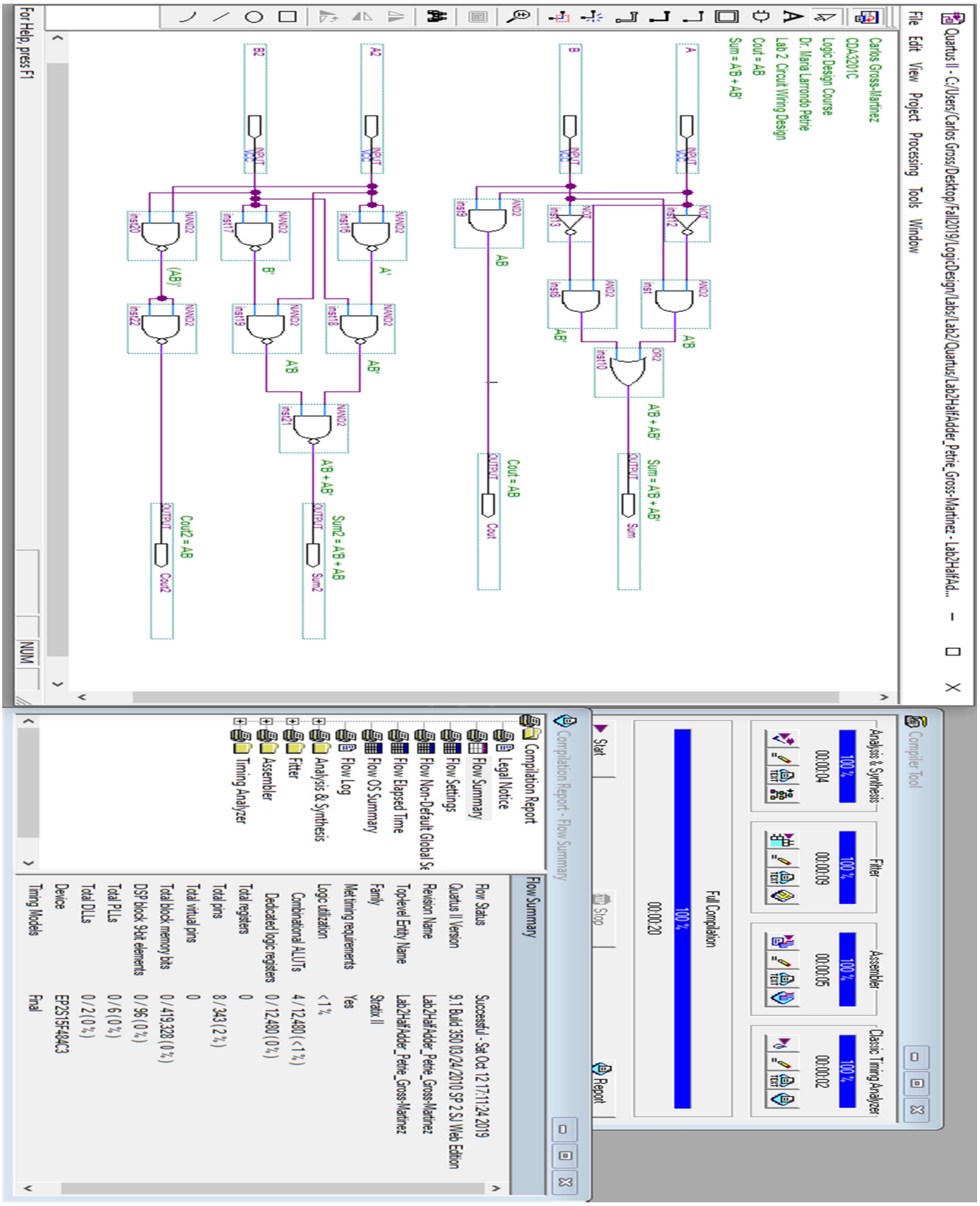
Quartus Schematic Regular Gates and NAND Gates

Cout = AB, Sum = A'B + AB'



Quartus Schematic Compilation Result Regular Gates and NAND Gates

Cout = AB, Sum = A'B + AB'



Quartus Waveform Compilation Result Regular Gates and NAND Gates

Cout = AB, Sum = A'B + AB'

Truth Table and K-map

Cout = AB + AC + BC, Sum = ABC + AB'C' + A'BC' + A'B'C

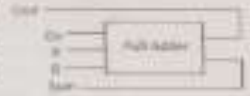
• CDA3201 • Intro to Logic Design •

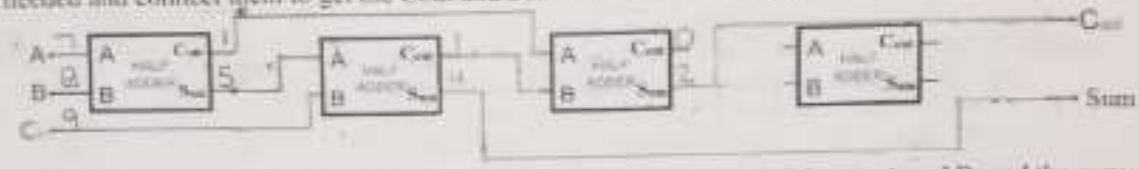
Lab Assignment **2**

Name: Carlos Gross-Martinez

Grade: /100

2.a.12 [10 Petrie] By abstracting our design for the Half Adder into a "black box", hiding all the gates of the circuit (does not matter if NOT-AND-OR or all-NAND), we can use it as a component to simplify how to design more complex components. Use Half Adders as components to build a Full Adder. A **Full Adder** adds 3 inputs: Carry-in (Cin) plus the two input bits: A and B. A Half Adder can only add 2 at a time, so we need multiple Half Adders. Figure out how many Half Adders below are needed and connect them to get the Cout and Sum answer of a Full Adder.





2.a.13 [5 Petrie] Complete the truth table for a **Full Binary Adder**. There are 3 inputs: A and B, and the carry from previous stage, called Cin. Label the sum bit is **Sum3** and the carry to next stage is **Cout3**.

Inputs				Outputs in binary	
A	B	Cin		Cout3	Sum3
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1	2	1	0
1	0	0	1	0	1
1	0	1	2	1	0
1	1	0	2	1	0
1	1	1	3	1	1

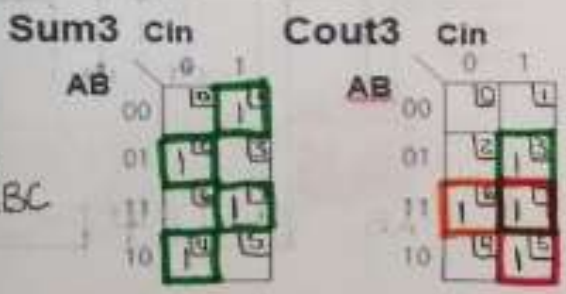
Decimal	Binary
0	0
1	1
2	10
3	11

2.a) [2 Petrie] From the Truth Table, write the **Sum of Minterms** and the **Canonical Sum of Products**:

Sum3 = f (A, B, Cin) = $\sum m(1,2,4,7)$, **Sum3a =** $\bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$

Cout3 = f (A, B, Cin) = $\sum m(3,5,6,7)$, **Cout3a =** $A\bar{B}C + AB\bar{C} + ABC$

2.b) [2 Petrie] Optimize the above functions using K-maps, label each cell with the corresponding minterm number in the upper right of each cell, then fill in the values of each cell according to the Truth table, find the groupings and the **simplest Sum of Products**:



Sum3 = f (A, B, Cin) = $\bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$

Cout3 = f (A, B, Cin) = $AB + AC + BC$

Logic Drawing Regular Gates and NAND Gates

Cout = AB + AC + BC, Sum = ABC + AB'C' + A'BC' + A'B'C

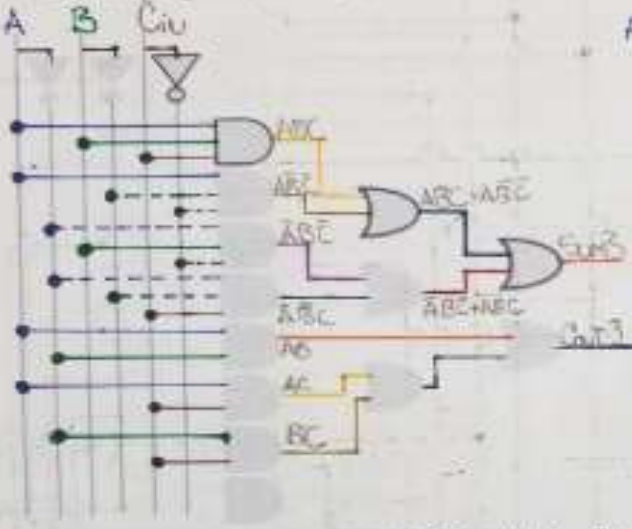
• CDA3201 • Intro to Logic Design •

Lab Assignment **2**

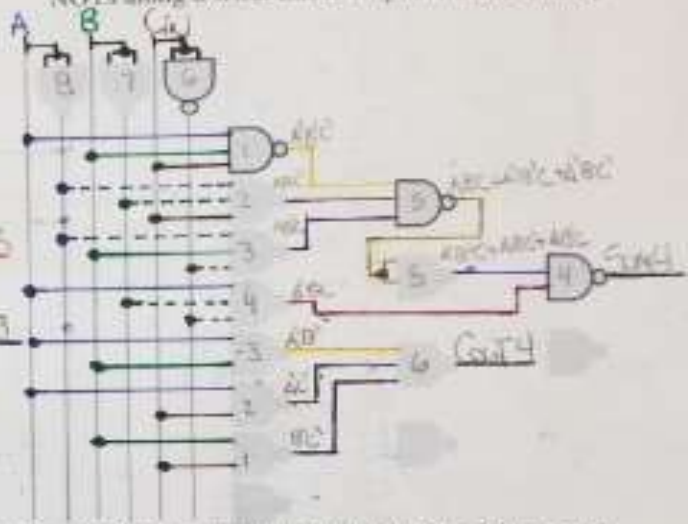
Name: Carlos Gross-Martinez

Grade: /100

2.c) [2-TA] Implement the two optimized functions, as a 3-input **(A, B, Cin)** 2-output **(Sum3, Cout3)** NOT-AND-OR network using NOTs and only 2- and 3-input gates (no 4-input)



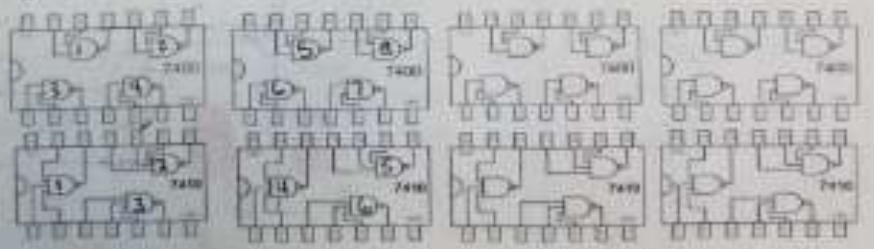
2.d) [2-TA] Convert 2.c to simplest all-NAND circuit by substituting equivalent NANDs configurations (see 2.a.7) (use only 2- and 3- input NAND gates, do not use 4-input NANDs), eliminate pairs of NOTs along a wire. Label outputs **Sum4, Cout4.**



2.g) [2-TA] Create a new project **Lab2_FullAdder_Petrie_YourName**. Draw schematics Cout3, Sum3; and Cout4, Sum4. Verify that Cout3=Cout4, and Sum3=Sum4 and their output matches the Truth Table for the Full Adder in 2a.13. Capture Schematics and results for your portfolio file.

2.e) [4-TA] From the above all-NAND circuit determine how many 7400 and 7410 logic chips you will need of each type **(maximum 4 chips total)** to build the Full Adder Sum4 and Cout4 circuits. *Number each NAND gate in 2.d and assign that same number to a gate in the corresponding chip below.*

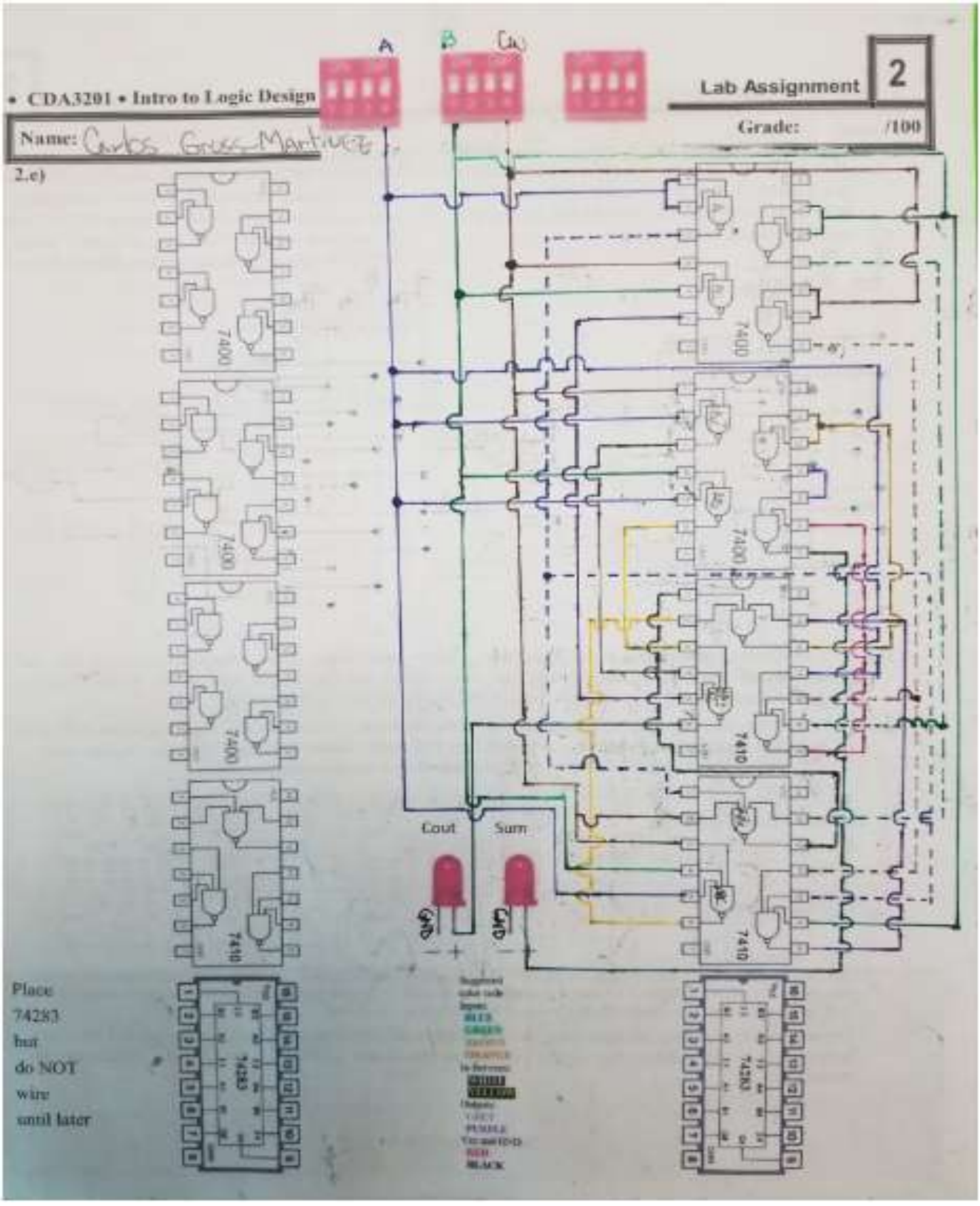
Chip	How many?
7400	2
7410	2



IMPORTANT: You will build a 5-bit Adder by the end of this lab, so first make sure you manage the space available on your breadboard by placing a 74283 4-bit Adder 16-pin chip on the left side of your breadboard and the 4 chips for the all-NAND Full Adder to its right. If you do not do this, you may have to redo all the wiring.

On page 6, use the diagram containing correct number of chips you need for your design and plan your wiring for the Full Adder. Use 3 logic switches as inputs and 2 LEDs for the Output. Build it on the breadboard.

Logic Drawing NAND Gates Using Logic Chips
Cout = AB + AC + BC, Sum = ABC + AB'C' + A'BC' + A'B'C



16 Bit Adder Diagram

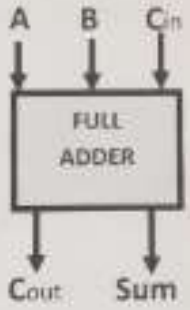
• CDA3201 • Intro to Logic Design •

Lab Assignment 2

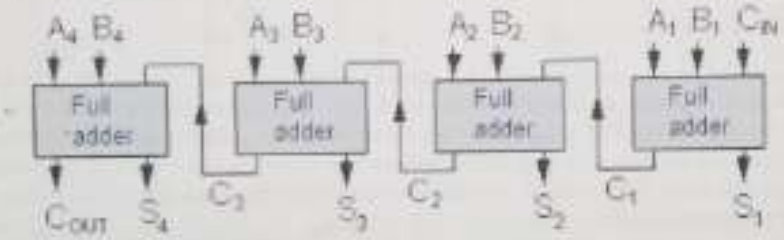
Name: Carlos Gross-Martinez

Grade: /100

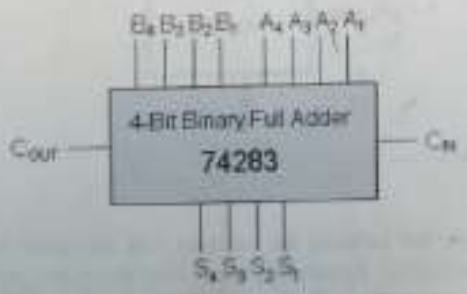
As you see there are many connections to create a 1 bit adder. Now we abstract all those gates into a "black box" as shown at right. We will use the black box as a component to build other things, only worrying about the inputs and outputs and how to interconnect the components, not what is inside the box.



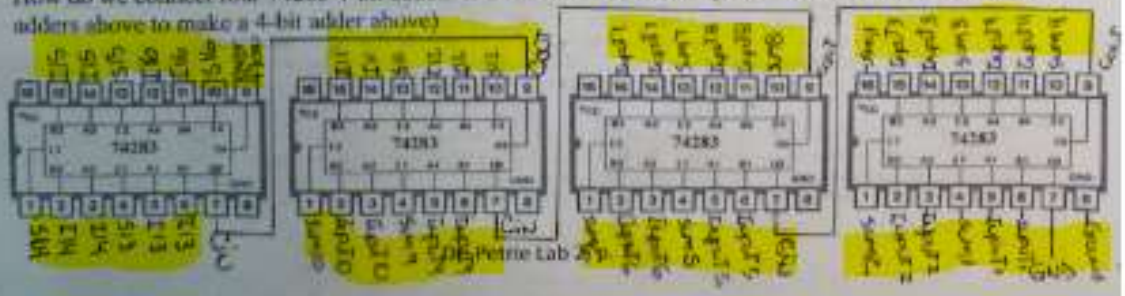
Below we see how we can connect four 1-bit Full Adders to create a 4-bit adder. Note instead of A and B the inputs are a 4-bit number: A_4 A_3 A_2 A_1 and another 4-bit number B_4 B_3 B_2 B_1 and a 1 bit Cin, and the outputs, Cout remains 1 bit but instead of Sum is a 4-bit S_4 S_3 S_2 S_1.



We can abstract the 4-bit adder we built above into one "black box", which turns out to be packaged into the 74283 chip. Below we see the logical diagram vs pinout diagram of the 74283. Note from the pinouts of the 74283, C0 is equivalent to Cin, C4 is equivalent to Cout, and $\sum$ is S (Sum).



How do we connect four 74283 4-bit adders to make a 16-bit adder? (Hint: see connections of four 1-bit adders above to make a 4-bit adder above)

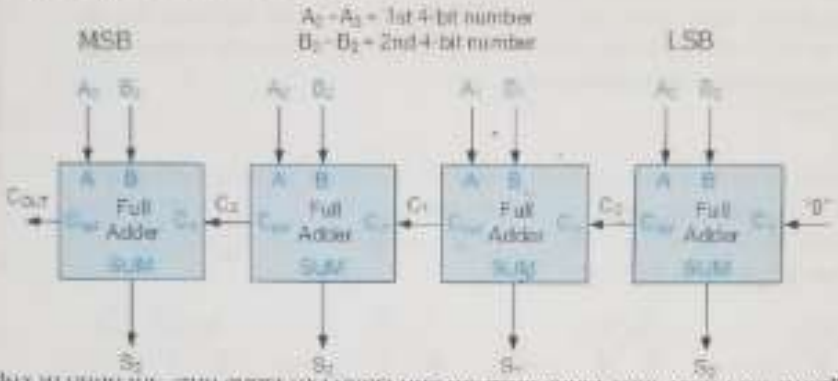


Answer to Lab Manual Question

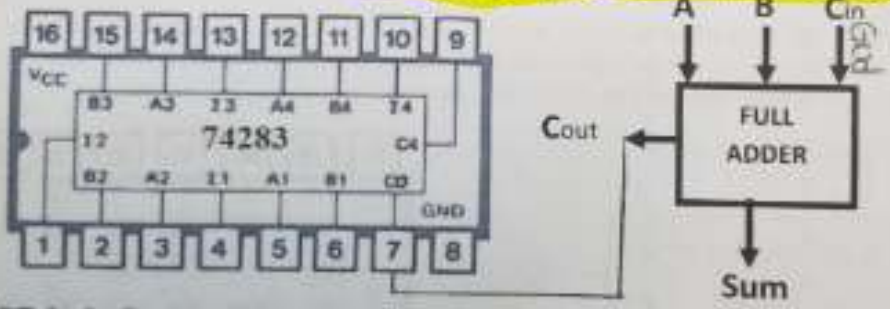
• CDA3201 • Intro to Logic Design • Lab Assignment **2**

Name: Carlos Gross-Martinez Grade: /100

If we used our 4 bit Full Adder to add a maximum of 4 bit numbers, then there is no carry-in for the least significant bit (LSB), note that the Cin of the LSB needs to be set to zero, and the carry-out from the most significant bit (MSB) is lost, because the sum has a maximum of 4 bits to store the answer. When Cout = 1 the answer is incorrect, and is said to "overflow".



Now you are ready to build the 4-bit adder by connecting the 4 full adder your Full Adder. The Full Adder we built is shown below as a component. It will be used as the Least Significant Bit (the right most column). If there is no column to the right of our Full Adder, then we should really use a Half Adder but we already have the Full Adder wired. How can we make the Full Adder behave like a Half Adder – there is no carry-in, so to what value should we set Cin of our Full Adder? To what do we connect the Cout of our Full Adder to in the 74283 chip? Make the connections below. Use Ground pin Co in ship. We connect to pin Co



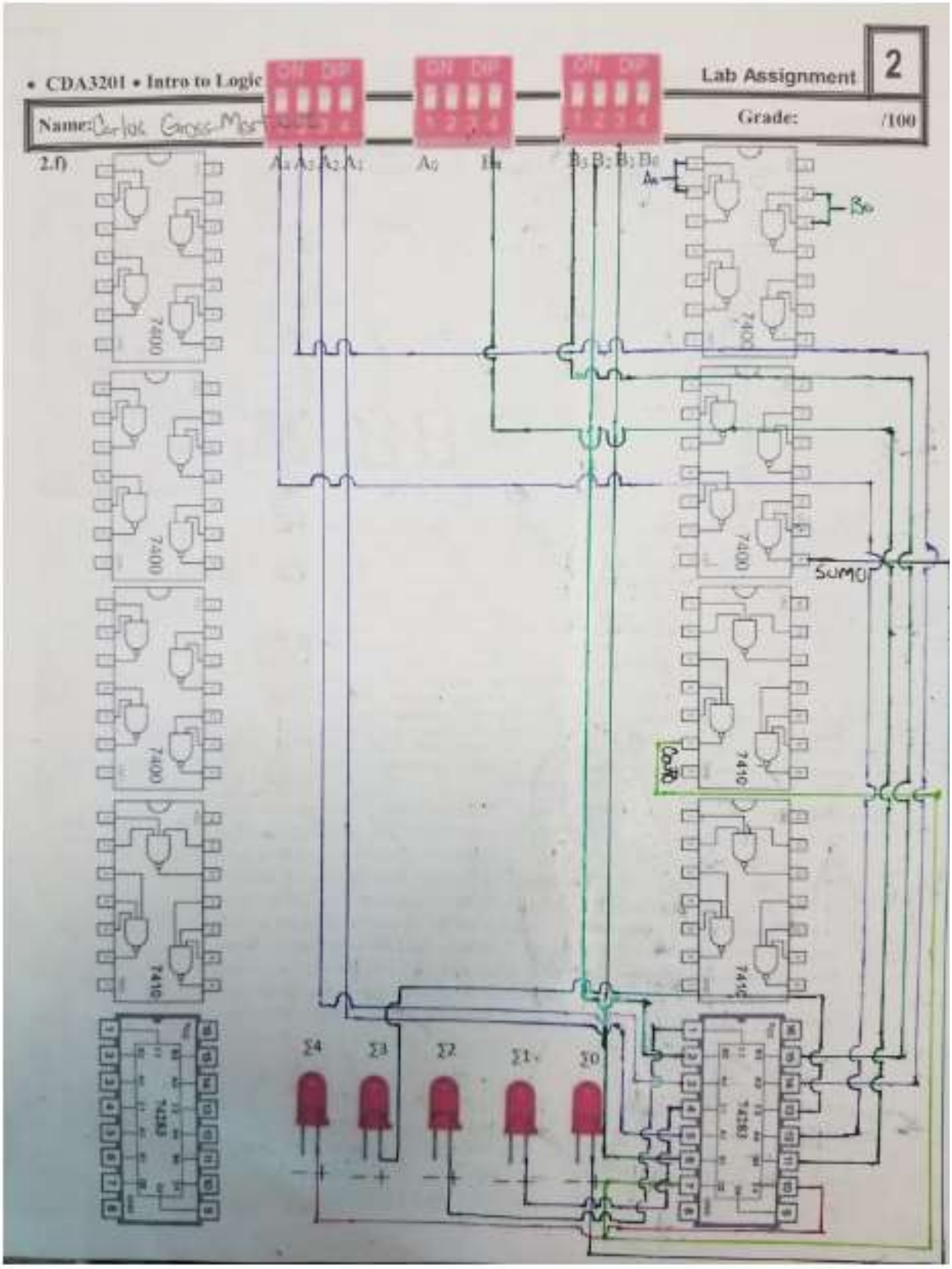
You will NOT do the Quartus for the 5-bit Adder.

2.f) [TA-6] **The 5-bit adder.** Connect your all-NAND 1-bit full adder to the standard 4-bit full adder (Chip 74283) making your adder the LSB of the 5-bit resulting adder. To simplify the testing procedure, set A4 and B4 switches to 0. To test your circuit, try as many input combinations as possible.

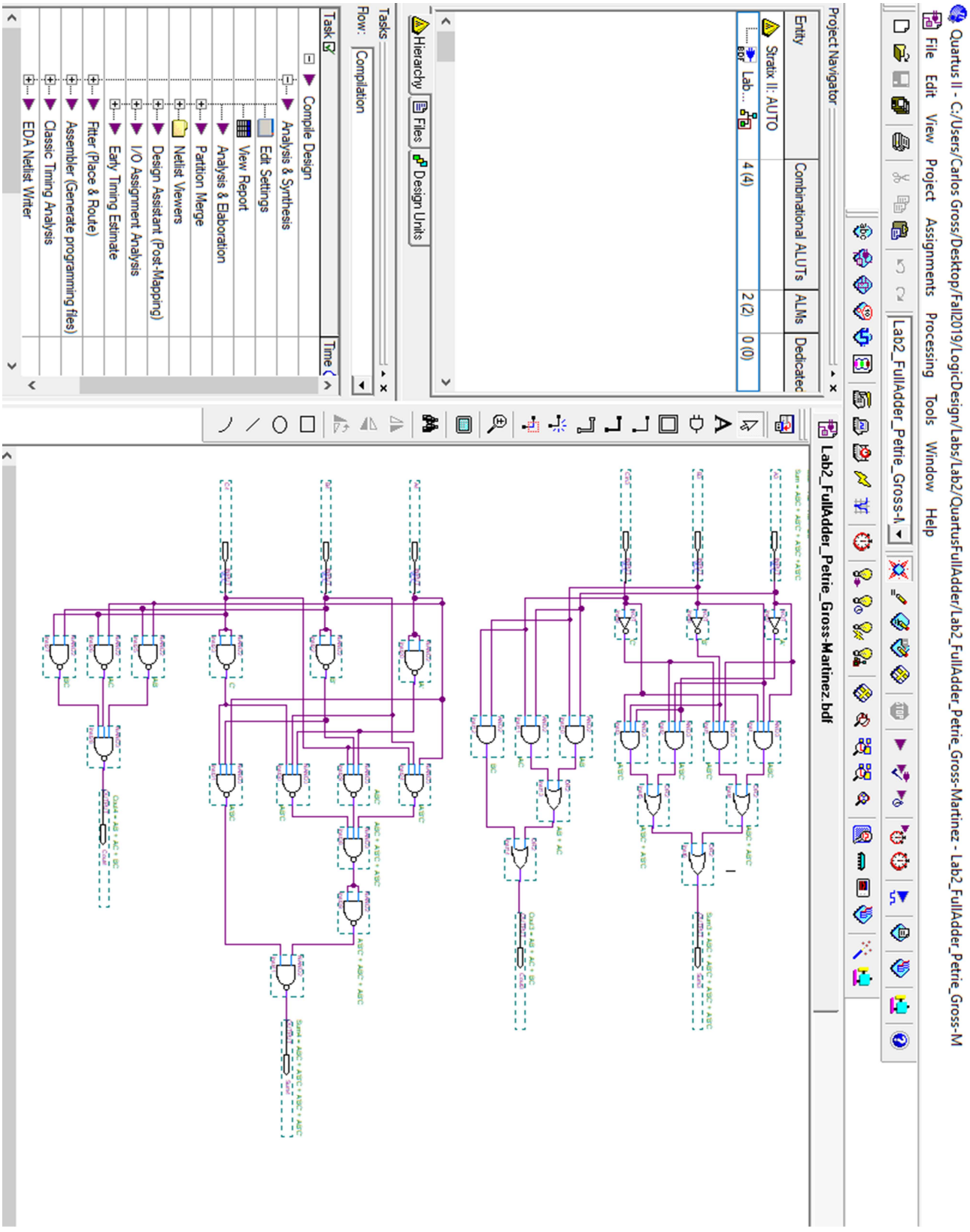
Note: you need to wire the 5-bit Adder on your breadboard. Choose the set of chips you selected to build your Full Adder circuit and ***label only*** *the location of* ***A*** *and* ***B*** *that go into the NOT, the* ***Cin*** *and* ***Cout****. Connect your Full Adder to the 74283 4-bit Adder as you designed above. You do not need to show all the other wiring of the Full Adder, just the pins used to connect to the 74283. Redo the switch connections for the 5 As and the 5 Bs, and wire up the 5 LEDs for the 5 bit Sum. Note the Cout of the 74283 is not part of the Sum and there is not another column/component to its left, so if it is 1, it will be lost because it is not connected to anything so the answer for the 5 bit Sum will be incorrect.*

IMPORTANT: You will run out of slots for input "A" if you always get the values from the "A" switch. To overcome this, connect only 1 wire from the input switch to a spot that reads "A" input on the back board, then jump from that location to other places that need an "A". This way if you need to change the "A" to another switch you only change one wire. You can also wire from switch to one open row, and put a jumper across the switch to have a whole row of 1's available. If you need more "A"s then jump to another open row.

Logic Drawing 5 Bit Adder
Chip 74283

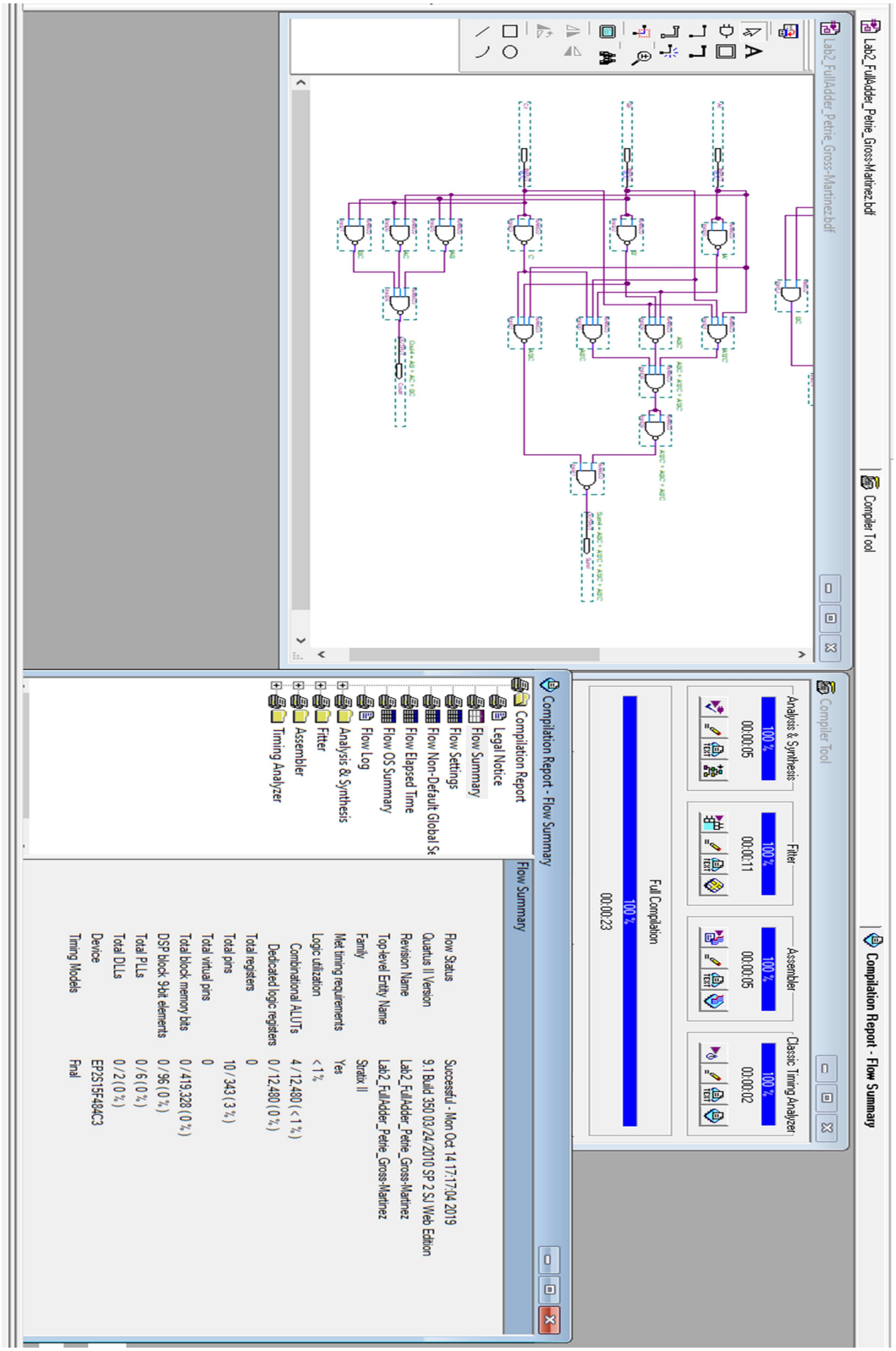


Quartus Schematic 1 Bit Adder Regular Gates and NAND Gates
Cout = AB + AC + BC, Sum = ABC + AB'C' + A'BC' + A'B'C



Quartus Schematic Compilation Result Regular Gates and NAND Gates

Cout = AB + AC + BC, Sum = ABC + AB'C' + A'BC' + A'B'C



Quartus Waveform Compilation Result Regular Gates and NAND Gates

Cout = AB + AC + BC, Sum = ABC + AB'C' + A'BC' + A'B'C

Picture of Completed Circuit on Breadboard