Lab2.2-Petrie - 5 bit adder - Piggy Bank - all NANDs - revised

Wednesday, October 28, 2020 12:25 PM

Lab2.2-Petri
e - 5 bit a...

• CDA3201 • Intro to Logic Design • **Lab Assignment** **2.2**

Name: **Grade:** /100

** This lab assignment is loosely based on, but greatly expands Lab 2 in FAU Logic Design Lab Manual by Dr. Bassem Alhalabi.*

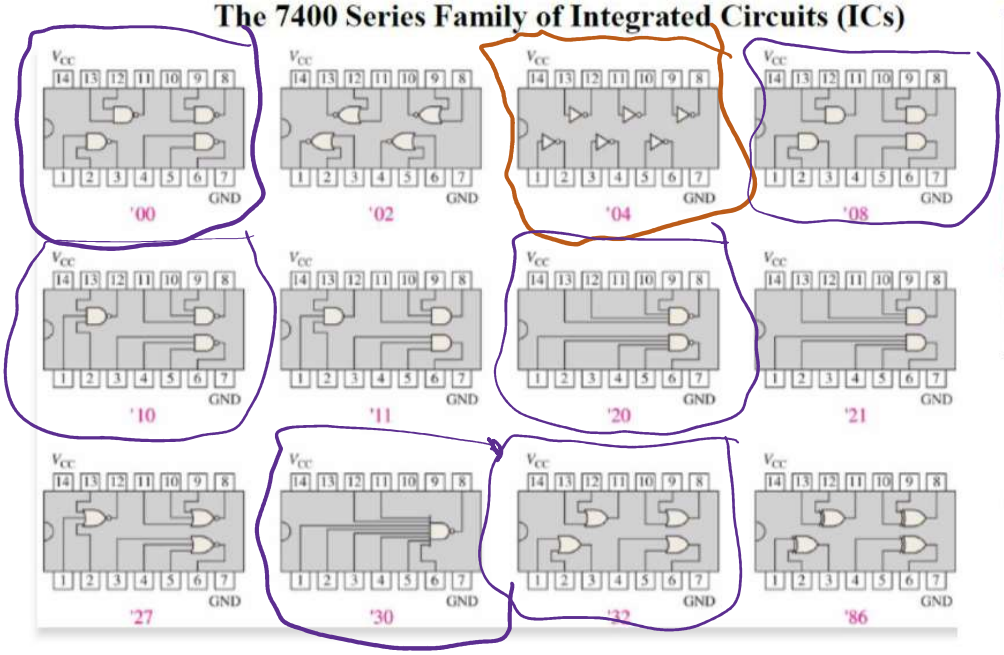
The Problem: The founder and CEO of Adafruit Industries, Limor Fried, known as Ladyada to her online following, just challenged your team to cut the price of the Piggy Bank adder circuit, so that her open source electronics kits and components company can donate a large number of the kits to after school programs in poor communities. Your team leader analyzed the challenge and found the price can be cut by redesigning the circuits with Universal gates. The Universal gates are the NAND and the NOR. They are called universal because you can buid the equivalent of any gate using only NANDs or only NORs. The NAND gate is cheaper because it only takes two transistor to build one NAND, compared with 4 transistors required to build an AND or OR gates. Twice as many NAND gates can fit on a silicon wafer, compared to AND or OR gates. However, NAND chips are not half the price of AND or OR chips because the casing and pins are a substantial cost of making a chip. However, by building the circuit with only one type of gate, the company can buy a large number of just one type of gate, providing a bulk purchase discount. The team leader has assigned you to redesign and simulate your half-adder and full-adder circuits with only NANDs. Someone else on the team is assigned to wire your design, so be neat and clear in your diagrams.

1st female engineer on cover of *WIRED* magazine. April 2008. https://www.wired.com/2011/03/wired-magazines-cover-features-its-first-

Entrepreneur of 2012 https://www.entrepreneur.com/topic/entrepreneur-of-2012

Make Magazine, Vol. 57, June/July 2017. https://makezine.com/tag/make57/

The 7400 Series Family of Integrated Circuits (ICs)



2.14 Identify all the 7400 family NAND ICs and the number of chips of these type that you have in your kit:

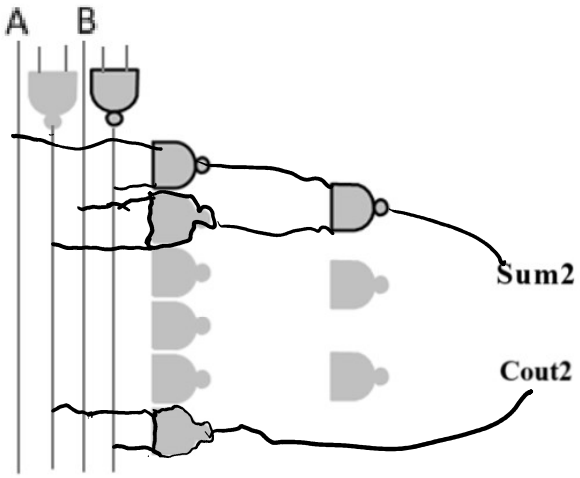
Chip Number	Num of gates in IC	Number of inputs per gate	Number ICs in your kit
74 00	4	2	
74 10	3	3	
74 20	2	4	
74 30	1	8	

Name: Grade: /100

2.15 Plan the equivalent all-NAND Circuit for the Half Adder: Draw the all NAND equivalent circuits of the NOT, AND, and OR Half Adder you designed in Lab 2 Part 1 Exercise 2.4. After substituting the equivalent NAND circuits for each gate, simplify by crossing out the NANDs that are not needed. **HINT**: 2 NOTs adjacent on same wire cancel each other, draw one line through ones that cancel out. Number the gates not crossed out so you can identify where you will place them on the NAND chips on exercise. Draw the simplified **Cout2** and **Sum2** as a 2-input (**A, B**) and 2-output (**C_{out1}, S_{um1}**) network. Use the shaded gates on the right to help you trace the gates you need, as shown on the outlined NOT, AND, and OR. Label all inputs and outputs of the gates.

Scratch work:

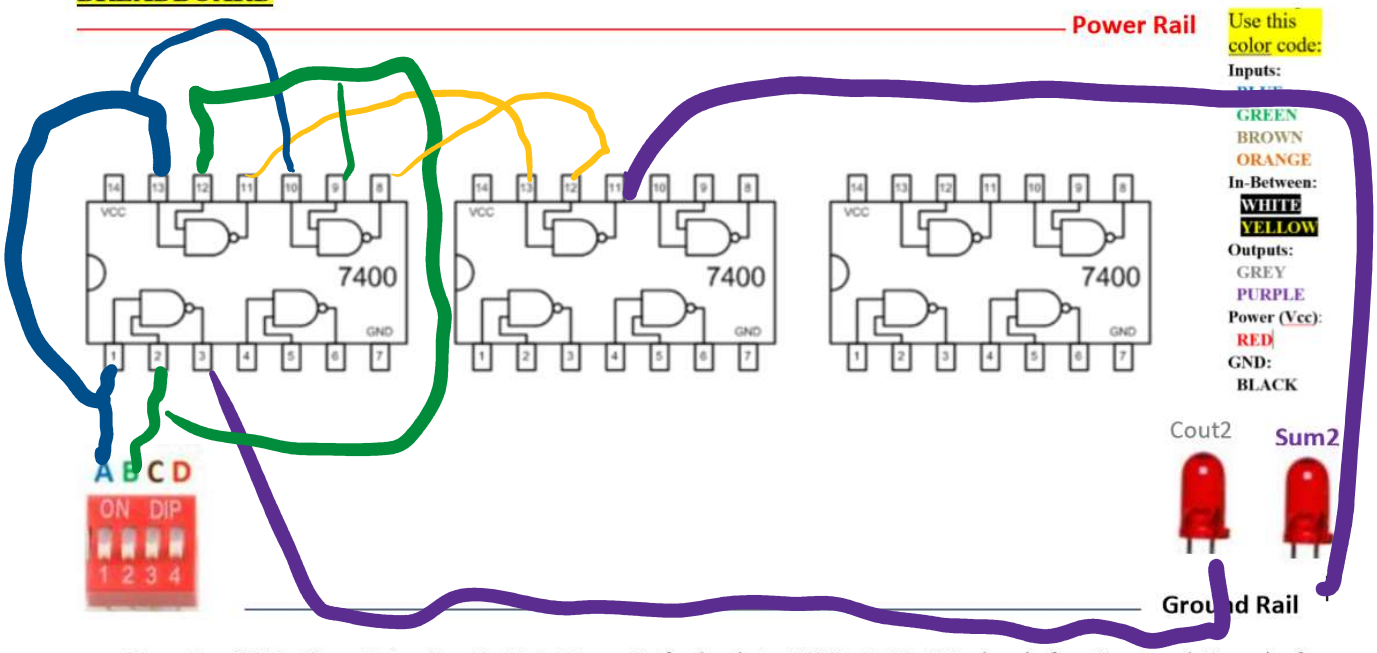
Solution:



2.16 Simulate your all-NAND Circuit of the Half Adder: Open your Quartus project **Lab2HalfAdder_Petrie_YourName**. Add to the .bdf file the schematics of the **Cout2** and **Sum2** of the all-NAND circuits. Compile, simulate, and verify both circuits are equivalent, i.e. Cout1=Cout2 and Sum1=Sum2. Capture the all-NAND schematic (.bdf window) and timing results (.vwf window) in your lab portfolio. Be sure that the name of the file appears in your snip-it of each of the two windows, this shows you did it.

Name: Grade: /100

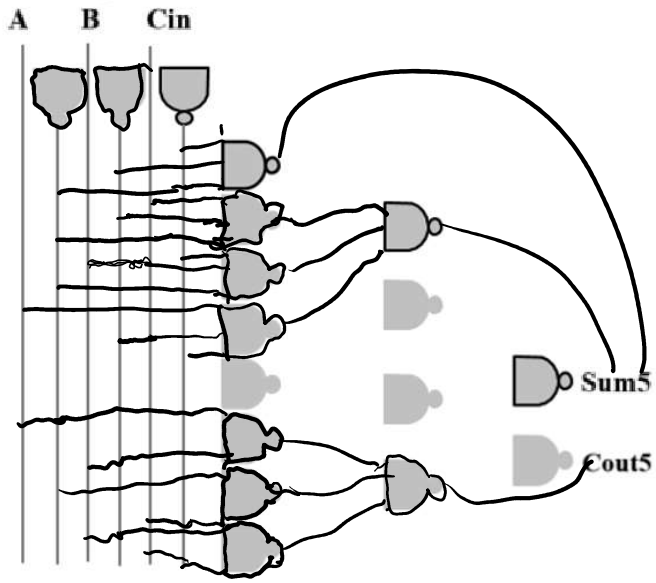
2.17 Plan the wiring of the half adder with all NANDs. Once verified, refer to your NOT AND OR design and design to use same color code on the inputs and outputs as you did in Lab 2 Part 1. **DO NOT WIRE ON BREADBOARD**



2.18 Plan the SOP Circuit for the Full Adder: Refer back to NOT-AND-OR circuit for **Cout** and **Sum** in for the Full Adder of Lab 2 Part 1 Exercise 2.11, and substitute the equivalent NAND circuit for each gate, labeling the outputs **Cout5** and **Sum5**. Cross out the NANDs that are not needed to find the simplest all NAND circuit. Draw the simplified solution to the right, use the shaded gates on the right to help you trace the gates you need, as shown on the outlined NOT, AND, and OR. Label all inputs and outputs of the gates. Note: Your team lead says it can be done with 14 gates; can you do it with 14?

Scratch work:

Solution:



• CDA3201 • Intro to Logic Design • **Lab Assignment** **2.2**

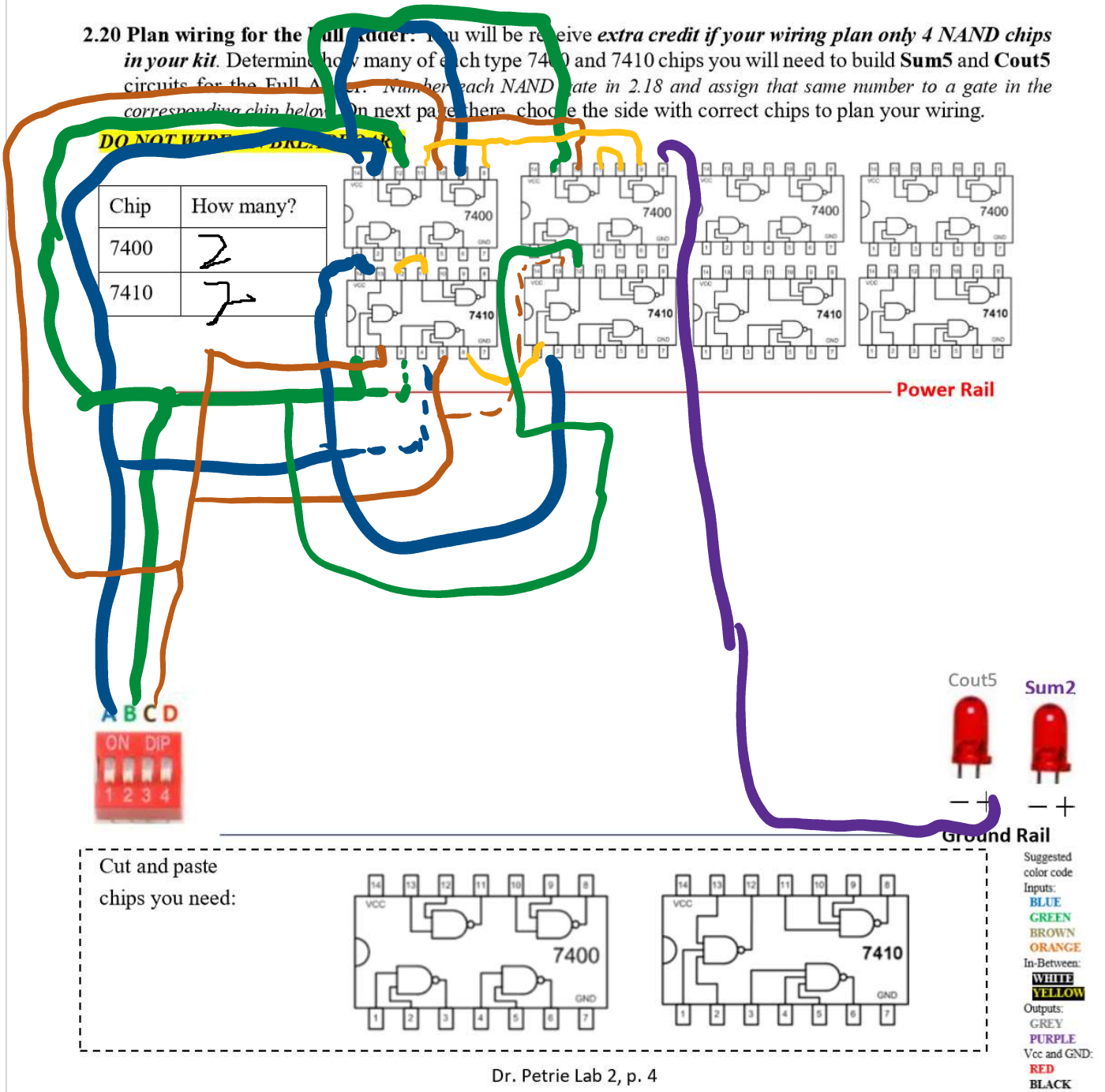
Name: **Grade:** /100

2.19 Simulate for the Full Adder: Open your **Lab2_FullAdder_Petrie_YourName**. Add the schematics for **Cout5**, **Sum5**. Verify that Cout4=Cout5, and Sum4=Sum5 and their output matches the Truth Table for the Full Adder in 2.13. Capture Schematics and results for your portfolio file.

2.20 Plan wiring for the Full Adder: You will be receive ***extra credit if your wiring plan only 4 NAND chips in your kit***. Determine how many of each type 7400 and 7410 chips you will need to build **Sum5** and **Cout5** circuits for the Full Adder. *Number each NAND gate in 2.18 and assign that same number to a gate in the corresponding chip below.* On next page there choose the side with correct chips to plan your wiring.

DO NOT WIRE ... BREADBOARD

Chip	How many?
7400	2
7410	2

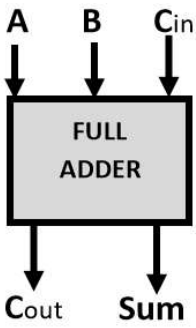


Cut and paste chips you need:

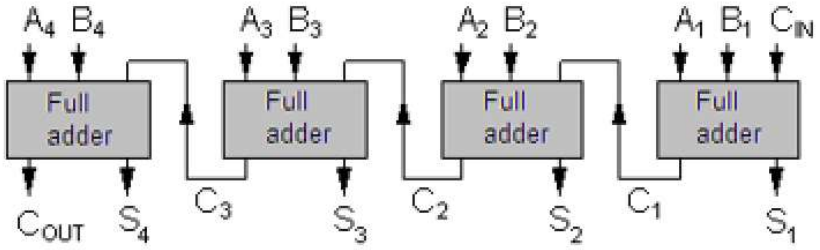
Suggested color code
Inputs:
BLUE
GREEN
BROWN
ORANGE
In-Between:
WHITE
YELLOW
Outputs:
GREY
PURPLE
Vcc and GND:
RED
BLACK

Name: **Grade:** /100

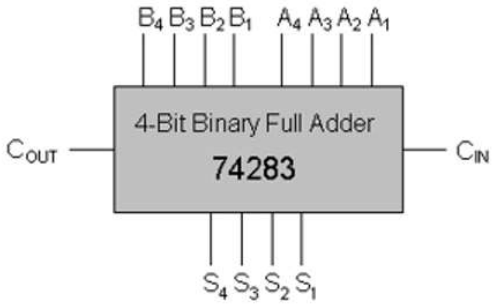
2.21 Building with components. As you can see there are many connections to create a 1 bit adder. Now we abstract all those gates into a "black box" as shown at right. We will use the black box as a component to build other things without worrying about the inputs and outputs and how to interconnect the components, not what is inside the box.

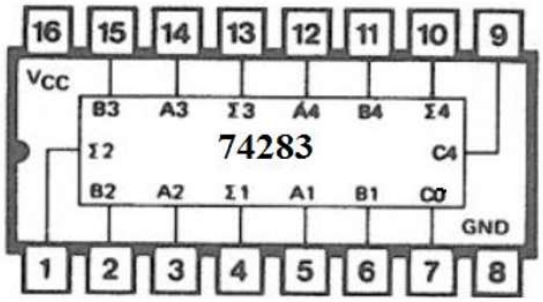


Below we see how we can connect four 1-bit Full Adders to create a 4-bit adder. Note instead of A and B the inputs are a 4-bit number: $A_4\ A_3\ A_2\ A_1$ and another 4-bit number $B_4\ B_3\ B_2\ B_1$ and a 1 bit Cin, and the outputs, Cout remains 1 bit but instead of Sum is a 4-bit $S_4\ S_3\ S_2\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

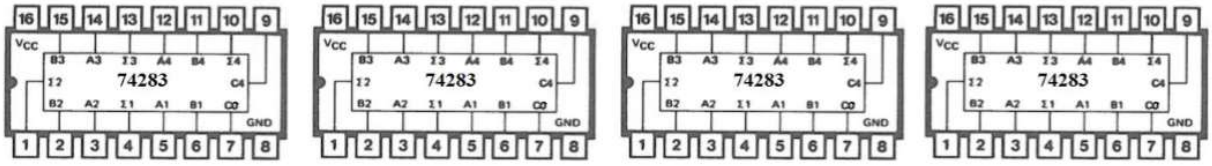


We can abstract the 4-bit adder we built above into one "black box", which turns out to be packaged into the 74283 chip. Below we see the logical diagram vs pinout diagram of the 74283. Note from the pinouts of the 74283, C0 is equivalent to Cin, C4 is equivalent to Cout, and ∑ is S (Sum).

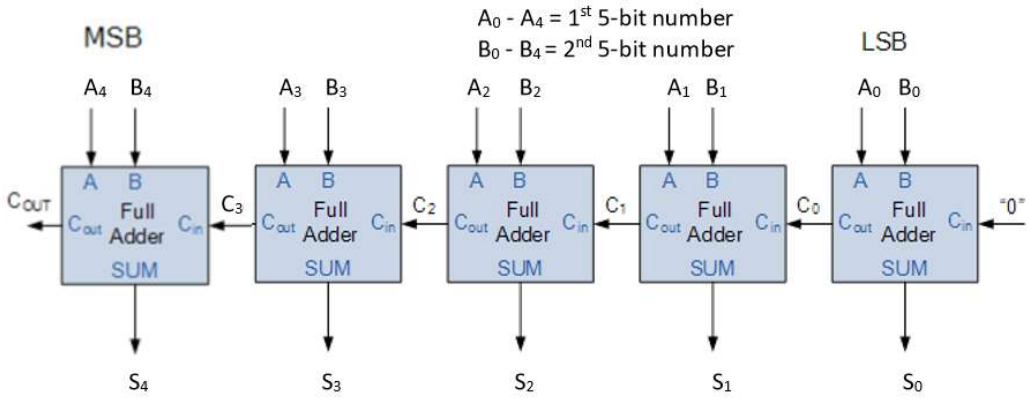




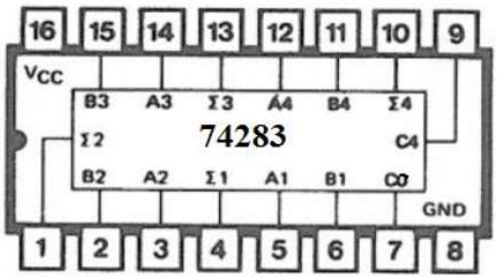
2.21a Connect four 74283 4-bit adders to make a 16-bit adder. (Hint: see connections of four 1-bit adders above to make a 4-bit adder above)

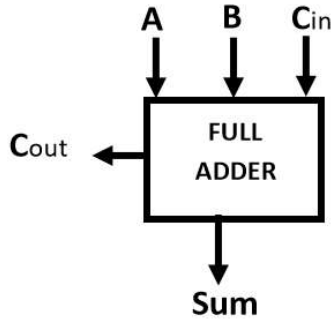


If we used our 1bit Full Adder to add a maximum of 5-bit numbers, then there is no carry-in for the least significant bit (LSB). Note that the Cin of the LSB needs to be set to zero, and the carry out from the most significant bit (MSB) is lost, because the Sum has a maximum of 5 bits to store the answer. When Cout = 1 the answer is incorrect, and is said to "overflow".



2.21b Connect a 4-bit adder to a full adder to make a 5-bit adder. The Full Adder you designed is shown below as a rectangle. It will be used to add the Least Significant Bit (to add the right most column). If there is no column to the right of our Full Adder, then we should really use a Half Adder. 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 in the 74283 chip? Make the connections below.





Lab2.1-Petrie - 5 bit adder - Piggy Bank - NOT AND OR

Wednesday, October 28, 2020 3:11 PM

Lab2.1-Petrie - 5 bit a...

• CDA3201 • Introduction to Logic Design • Dr. Maria Larrondo Petrie **Lab Assignment** **2***

Name:	ID#:	Grade:	/60

Fig. 1. Ada Fruit's first prototype.
Source:https://blog.adafruit.com/2012/04/23/how-to-electronic-piggy-bank-

Fig. 2. Ada Fruit's single coin acceptor.
Source: https://www.adafruit.com/product/786

Fig. 3. Ada Fruit's 4 coin type coin acceptor.
Source: https://www.adafruit.com/product/787

** This lab assignment is inspired by but greatly expands Lab 2 in FAU Logic Design Lab Manual by Dr. B. Alhalabi with recommendations from Mr. C. Weinthal to add Part 3 using the decoder.*

The Problem: The Ada Fruit company has hired you as an intern and put you in a team that will design a better, more interactive piggy bank. The leader of the team has provided the team with links to review:

- View video of 1st Piggy Bank prototype (Fig 1.). It lights up the piggy when a coin drops into it and displays the total money in the bank on the side of the barn. It only accepts quarters because it used Ada Fruit's single coin acceptor (Fig 2). Ada Fruit now has Programmable 4 Coin Type Coin Acceptor (Fig 3).
- Reverse engineer the Counting Piggy Bank (Fig 4).
- Power Hog design (Fig.5) took 2nd in 2009 Greener Gadgets competition.
- The Digital Coin Counting Piggy bank (Fig 6) says "Oink, Oink", when a coin is dropped, displays the denomination of the coin and then the total.
- The Pig E Bank (Fig 7) shakes when you drop a coin into it, identifies coin value, moves its mouth to ask for more money, and totals moneys in bank.

The team asks you to design several binary adder circuits capable of adding 2 binary digits, *bits*, and later, calculate a 4-bit sum in binary. The 4-bit answer will display as a hex digit on the seven-segment display. Use an LED to show value of the carry to the 5th column. Another team member will convert hex to binary.

Fig. 5. The Power Hog design 2nd place winner of the Greener Gadgets 2009 design competition.
Source: https://bustler.net/news/748/winners-of-greener-gadgets-2009-announced

Fig. 4. Counting Piggy Bank teardown.
Source:http://blog.spitzenpfeil.org/wordpress/2011/12/31/counting-piggy-bank-teardown/

Fig. 6. Digital Coin Counting Piggy bank:
Source: https://www.hisgifts.com.au/digital-piggy-bank/?partner=245133524&affid=245133524l

Fig. 7. Learning's Journey Pig E Bank.
Source:https://www.educationaltoysplanet.com/electronic-piggy-bank-pig-e-bank-pink-with-lcd.html

• CDA3201 • Introduction to Logic Design • Dr. Maria Larrondo Petrie **Lab Assignment** **2***

Name:	ID#:	Grade:	/60

Objectives:

1. Add Binary numbers
2. Use Boolean Operations to implement Binary Arithmetic operations
3. Design and implement a Half Adder circuit and Full Adder circuit.
4. Design equivalent NOT AND OR circuits, all NAND circuits, and Decoder and NAND circuits
5. Compose new circuits using previously designed circuits
6. Demo the implementation operating to add the least significant column and the most significant column of binary addition.
7. Display a binary 4-bit sum in hex on the 7-segment display and display the carry to next column on an LED.

What you need to know before you start:

Arithmetic Operations include **ADDITION** (+) and **MULTIPLICATION** (•).
Boolean Operations include **OR** (+) and **AND** (•).

Carry
A
+ B
C_{out} Sum

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

Although the + symbols look the same, the operations and results differ. In Lab 0 and Lab 1, the circuits performed Boolean Operations. In Lab 2, the circuits will perform the Arithmetic Operation of Addition of Binary Numbers.

Add the **decimal numbers** (base 10) below, show carry into next column:

Ex: 11						
95	2	5	9	8	71	2598
+ 28	+3	+6	+7	+4	+29	+3674
123	5	11	16	12	100	6272

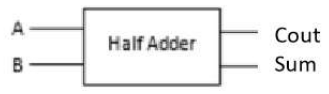
Add the **binary numbers** (base 2) below, show carry (consult decimal to binary table)

Ex: 111						
101	0	0	1	1	110	1101
+011	+0	+1	+0	+1	+011	+0111
1000	0	1	1	10	1001	10100

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

Half Adder: 2 inputs, 2 outputs

A
+ B
C_{out} Sum



A **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: 3 inputs, 2 outputs

Carry-in
A X
+ B X
C_{out} Sum X

A **Full Adder** is used 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_{out} 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 of the total is the *Sum*, while the most significant digit of the total is the *Carry* to the next column to its left, which we call C_{out}

There are 3 parts to this lab. Each worth 20 points. You will have one week to complete each part, and will hand in your portfolio at the end of each week, expanding it for the next week with the new designs. In Part I, you will design two circuits: a Half Adder, and a Full Adder using NOT, AND, and OR gates and test the designs work by simulating with Altera Quartus. In Part II, you will design the same circuits but using only NAND gates. In Part III, you will again design the same circuits using Decoders and NAND gates. The only one that you will wire is Part III, you will implement your design using decoders and include a 4-bit adder chip to demo that it works adding the least significant column (right-most 1st column), and later set up to add the most significant column (4th).

Name: **ID#:** **Grade:** /60

Part 1: Design a NOT AND OR Circuits of a Half Adder and a Full Adder [20]

2.1 Truth Table for the Half Adder. 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, **Sum** is the least significant bit of the result, and **C_{out}** is the most significant bit or "carry out". In 1st two column enter the product and minterm # associate with each row. Add A and B to enter decimal result in gray column. Show the 2 bit binary result in the last 2 columns.

Product	Minterm #	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

2.2 Boolean Algebra Equations for the Half Adder. Use the Truth Table in 2.1 to complete the Boolean Equations for the 2 outputs, **Sum** and **Cout,** expressed as a **Sum of Minterms (∑m)**, as **Product of Maxterms (ΠM)**, and as a **Canonical Sum of Products, (fα (A, B))**:

Sum = ∑ m (1, 2 **)**, **Sum = Π M (** 0, 3 **)**, **Sumα = fα (A, B) =** A'B + AB'

C_{out} = ∑ m (3 **)**, **C_{out} = Π M (** 0, 1, 2 **)**, **C_{out}α = fα (A, B) =** AB

2.3 Simplest Sum of Products (SOP) for the Half Adder. Use Karnaugh Maps (K-Map) provided to find the Simplest Sum of Products equations for Sum and C_{out}. Number each cell (in pencil in the upper right corner), enter the corresponding value of the output from the Truth Table, and group to show simplification. Use different colors to circle each group. Call the SOP equations

Sum

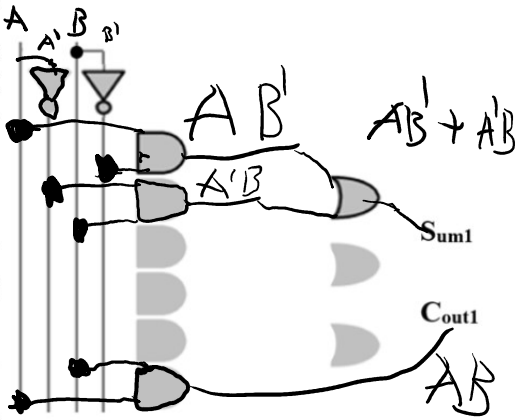
A \ B	0	1
0	0	1
1	1	0

C_{out}

A \ B	0	1
0	0	0
1	0	1

Simplest Sum of Products (SOP) Equations: **Sum1 =** AB' + A'B **C_{out}1 =** AB

2.4 Plan SoP Circuits for the Half Adder: Design a NOT-AND-OR circuit for the simplified **C_{out}** and **Sum** as a 2-input (**A, B**) and 2-output (**C_{out}1, Sum1**) network. Use the shaded gates on the right to help you trace the gates you need, as shown on the outlined NOT, AND, and OR. Label all inputs and outputs of the gates. Number each gate: How many gates did you use? 6
How many NOTs? 2 ANDs? 3 ORs? 1



2.5 Simulate Simplest SoP of the Half Adder: Create a project in **Altera Quartus** named **Lab2HalfAdder_YourName**. We will be doing a number of simulations, so call the outputs of this half adder **Sum1** and **Cout1**. Verify that the NOT AND OR Circuits for **Sum1** and **Cout1** work by drawing the schematic (.bdf file), compiling and simulating it. Verify timing diagram (.vwf file) output matches truth table in 2.1. Capture schematic and timing diagram in your portfolio file, making sure file name is visible.

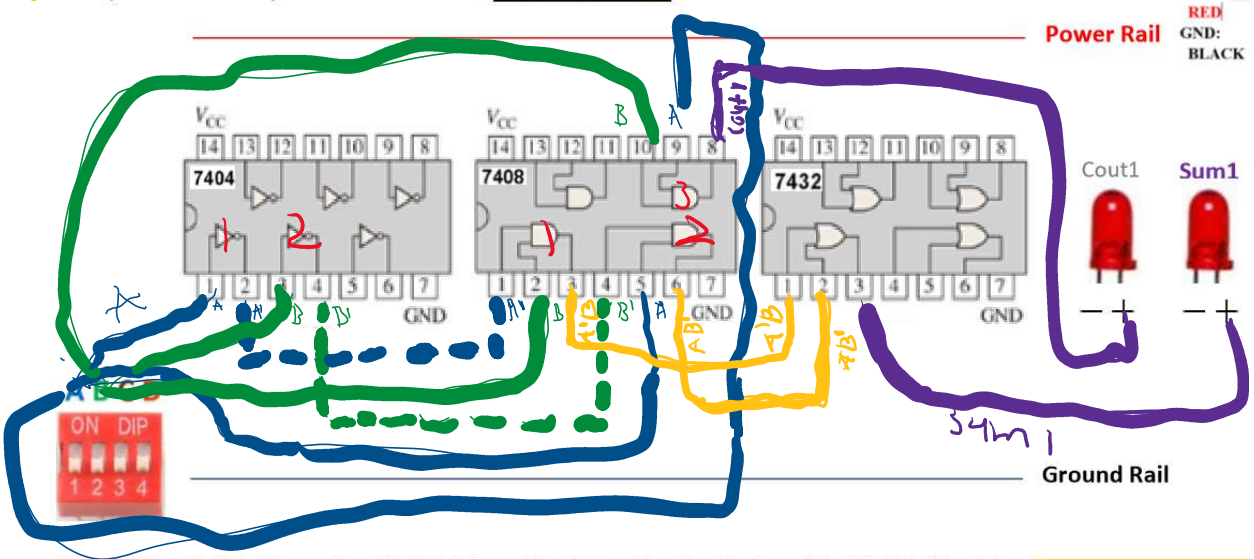
• CDA3201 • Introduction to Logic Design • Dr. Maria Larrondo Petrie **Lab Assignment** **2***

Name:	ID#:	Grade: /60

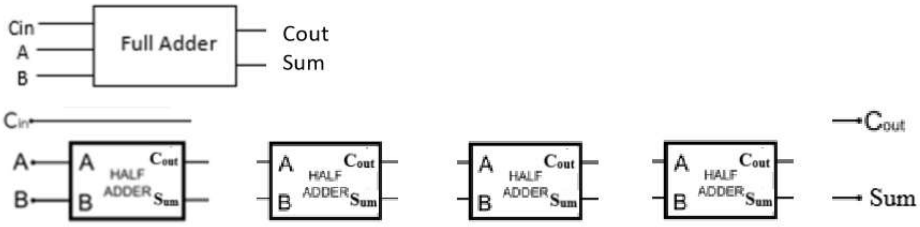
Copy from previous page SOP for Half Adder : **Sum1 =** $A'B + AB'$ **Cout1 =** AB

2.6 Plan the wiring of the Half Adder but DO NOT WIRE: Once verified circuit simulation matches outputs of the truth table, plan your wiring below of both circuits using the color code on the right. If you do not use the color code, you will not get any help to debug! Number each gate used in section 2.4 and identify where you will place that gate with the same number in the wiring diagram below. Label all inputs and outputs of each gate used. Note: wires from gray or green tubes (CAT5 wire = 22 gauge) 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 will be used for up to 4 inputs. Note: you have other colors that have no corresponding white striped wire: WHITE, GREY, PURPLE, YELLOW. Besides these: RED is used only for Vcc (Power), and BLACK is used only for Ground. Use BLUE for A, GREEN for B and BROWN for C. Use GREY for Cout1 and PURPLE for Sum 1. Use the corresponding color/White wire for its corresponding NOT (use dashed color lines on your drawing to designate NOT wire). You can use yellow or white to draw the wires input the ANDs and ORs. In the drawing you can designate a white wire with just a dashed black line. After all input and outputs of gates used are labeled with their inputs and outputs, use colored lines (or colored dash lines) to draw wire connections. **DO NOT WIRE**

Use this color code:
Inputs:
BLUE
GREEN
BROWN
ORANGE
In-Between:
WHITE
YELLOW
Outputs:
GREY
PURPLE
Power (Vcc):
RED
GND:
BLACK



2.7 Build a Full Adder using Half Adders: By abstracting the design of the Half Adder into a "black box", hiding all the gates of the circuit, 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 (C_{in}) plus the two input bits: **A** and **B**. A Half Adder can only add 2 numbers at a time, so we need multiple Half Adders. Figure out how many Half Adders are needed and how to connect them to get the C_{out} and **Sum** answer of a Full Adder.



Hint: How would you add 3 digits if you could only add 2 at a time to get final Sum and Carry?
Try: 7 + 8 + 9 = 24
Cin=7, A=8, and B=9. How do we get the final C_{out}=2, Sum=4? The first Half Adder has A=8, B=9, results in Cout=1, Sum=7. Use another Half Adder to add 2 numbers, … to eventually get the final C_{out}=2, Sum=4.

Name: Grade: /60

2.8 Truth Table for the Full Adder: Complete the truth table for a **Full Binary Adder**. There are 3 inputs: **A** and **B**, and C_{in}, the carry from adding the previous column, called Label the sum, **Sum4**, and the carry to next column, $C_{out}4$.

Inputs A	Inputs B	Inputs C_{in}	Calculate sum of A, B, C_{in} in decimal	Outputs in binary $C_{out}4$	Outputs in binary Sum4
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

2.9 Boolean Algebra Equations for Full Adder: From the Truth Table, write the **Sum of Minterms, Product of Maxterms**, and the **Canonical Sum of Products**:

Sum4 = $\sum$ m (1,2,4,7), **Sum4** = $\prod$ M (0,3,5,6), **Sum4α** = A'B'C + A'BC' + AB'C' + ABC

$C_{out}4$ = $\sum$ m (3,5,6,7), $C_{out}4$ = $\prod$ M (0,1,2,4), $C_{out}4α$ = A'BC + AB'C + ABC' + AB (

2.10 Simplest Sum of Product (SoP) of the Full Adder: Simplify 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. Draw ovals around the groupings each in a different color and find the **simplest Sum of Products**:

Sum4 = f (A, B, Cin) = A'B'C_{in} + AB'C' + ABC + ABC$_{in}$

$C_{out}4$ **= f (A, B, Cin) =** AB + AC + BC_{in}

Sum4

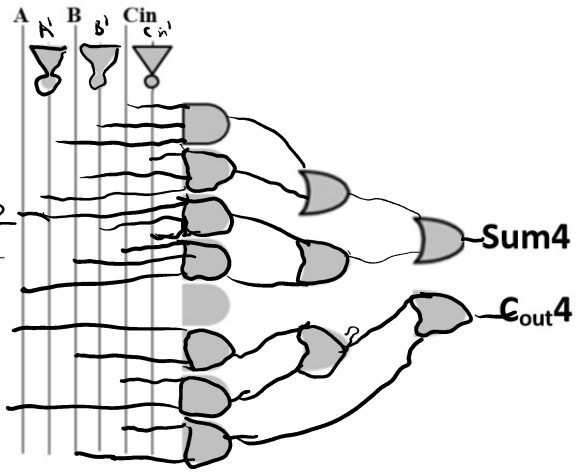
AB \ Cin	0	1
00	0	1
01	1	0
11	0	1
10	1	0

$C_{out}4$

AB \ Cin	0	1
00	0	0
01	0	1
11	1	1
10	0	1

Name: Grade: /60

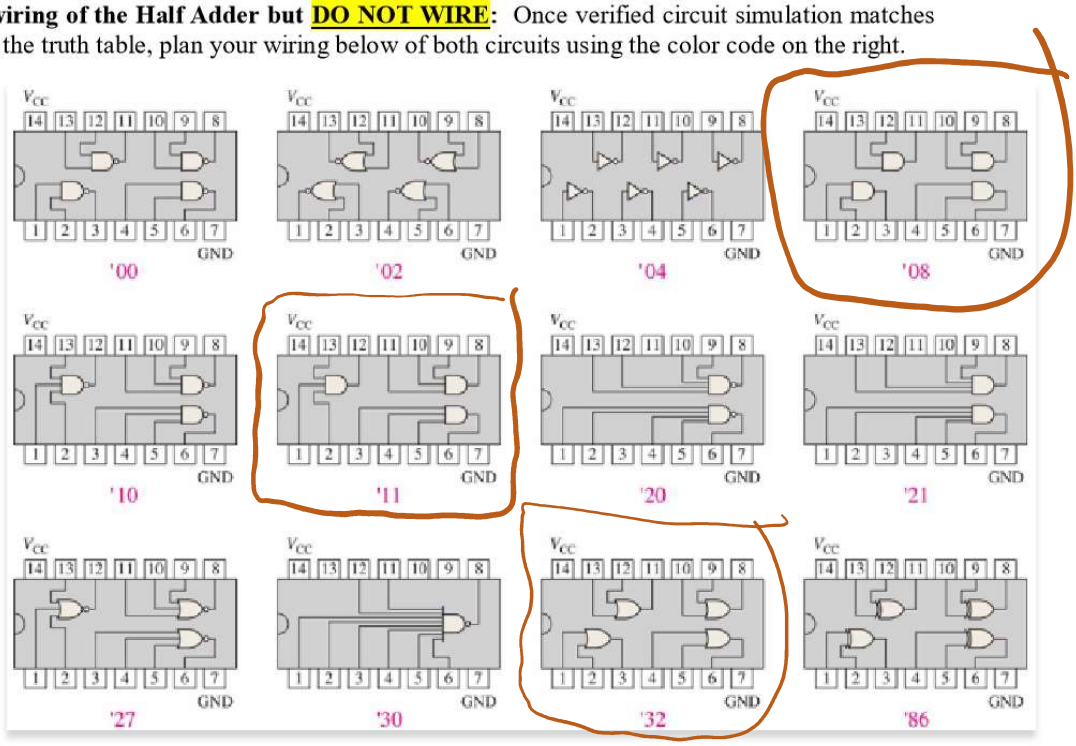
2.11 Plan the SOP Circuits for the Full Adder: Design a NOT-AND-OR circuit for the simplified **$C_{out}4$** and **Sum4** as a 3-input (**A, B, C_{in}**) and 2-output (**$C_{out}4$, Sum4**) network. Use the shaded gates on the right to help you trace the gates you need, as shown on the outlined NOT, AND, and OR. Label all inputs and outputs of the gates. Number each gate. How many gates did you use? 22 How many NOTs? 6 ANDs? 11 ORs? 5



2.12 Simulate the Full Adder: Create a project in **Quartus** named **Lab2FullAdder_YourName**. We will be doing a number of simulations, so call the outputs of this half adder **Sum4** and **$C_{out}4$**. Verify that the NOT AND OR Circuits for **Sum4** and **$C_{out}4$** work by drawing the schematic (.bdf file), compiling and simulating it. Verify timing diagram (.vwf file) output matches truth table in 2.8. Capture schematic and timing diagram in your portfolio file, making sure file name is visible.

2.6 Plan **the wiring of the Half Adder but DO NOT WIRE**: Once verified circuit simulation matches outputs of the truth table, plan your wiring below of both circuits using the color code on the right.

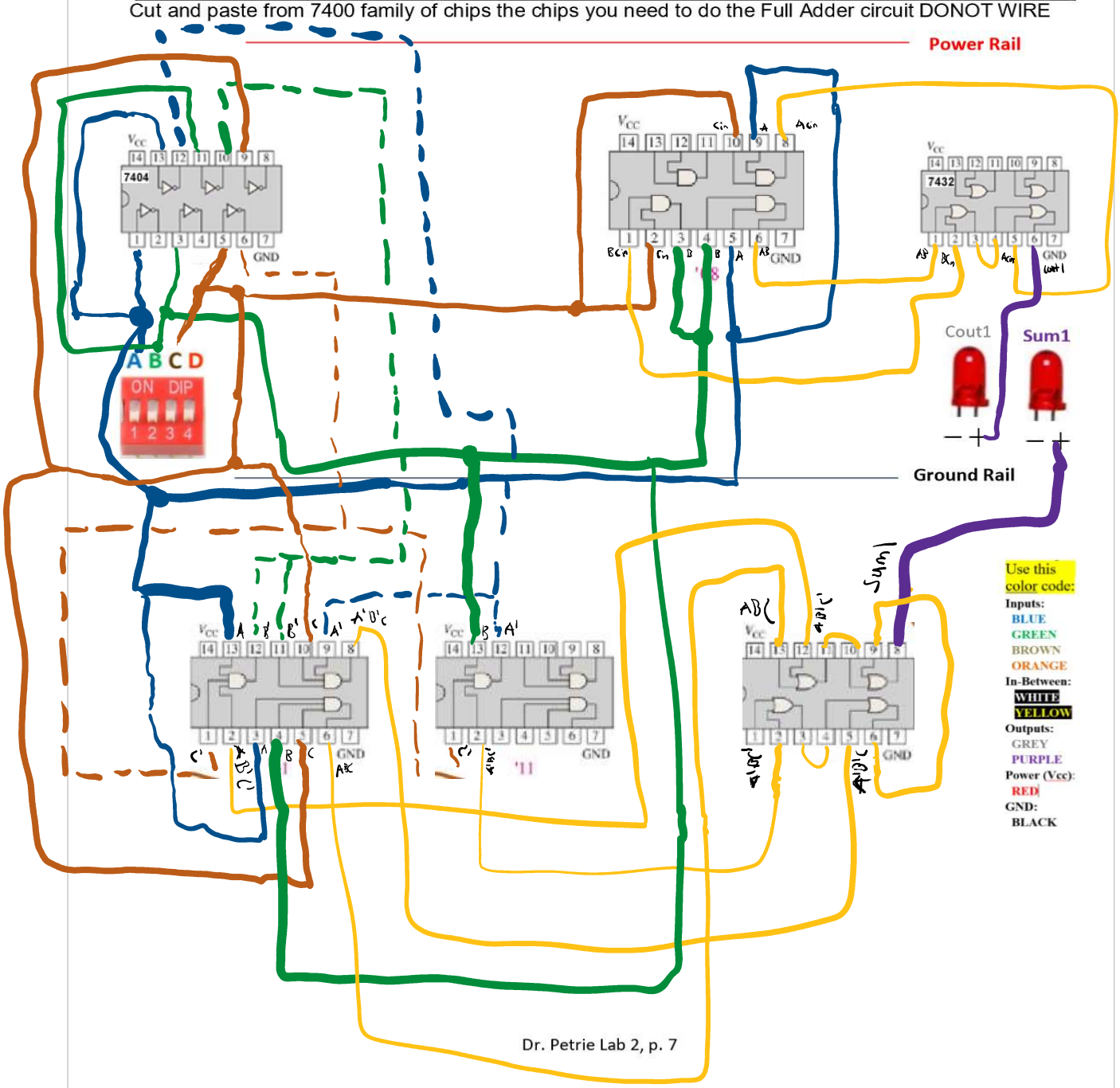
Cut and paste the chips that you need and draw the wiring diagram.



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

Name: **Grade:** /60

Cut and paste from 7400 family of chips the chips you need to do the Full Adder circuit DONOT WIRE



Dr. Petrie Lab 2, p. 7

Lab2.3 Decoders, 7 segment Display and Probe

Friday, November 6, 2020 1:35 PM

• CDA3201 • Intro to Logic Design • **Lab Assignment 2.3**

Name: **Grade:** /100%

Problem: You just received an email from Adafruit CEO that she wants you to lead the Adafruit student interns to form a team and compete in the next FAU Hackathon Competition under the sponsorship of Adafruit. This hackathon event is an intensive 24-hour design marathon, where teams compete to design solutions to real world problems and create functioning hardware or software prototypes by the end of the event. Hackathons start with one or more presentations about the event and the specific subject, they may be given ahead of time a kit of components that the team can incorporate in their hardware prototypes. On the day of the event, the specific problem is announced and the teams come up with ideas and appoint roles according to their interests and skills. At the end of the hackathon, each team presents and demonstrates their results. A panel of judges select the winning teams, and prizes are given. Adafruit is motivating the Adafruit team members by offering if they win: a $10,000 scholarship for graduate studies or certifications, and a one year membership to the Institute of Electrical and Electronic Engineers (IEEE), the largest professional organization in the world. Your Adafruit mentor wants you to become very familiar with designing with **decoders** (also called **demultiplexes**) because, although it is more expensive than implementing with NOT AND OR and with all-NANDS, it allows you to build a prototype circuit very easily and quickly. He sent you the following notes and worked out how to build the Half-Adder with a Decoder and NANDs. He tells you to look at datasheets to study decoders, and design the Full-Adder with a Decoder and NANDs to practice for the competition. The team and your mentor are counting on you!

Example of how to build the Half Adder with a Decoder and NANDs

Step 1: From Truth Table for the Half Adder (get it from Part 1) find Least Significant Bit

Minterm #	Inputs A (lsb)	Inputs B	Calculate A plus B in decimal	Outputs A plus B in binary C_{out}	Outputs A plus B in binary Sum
0	0	0	0	0	0
1	0	1	1	0	1
2	1	0	1	0	1
3	1	1	2	1	0

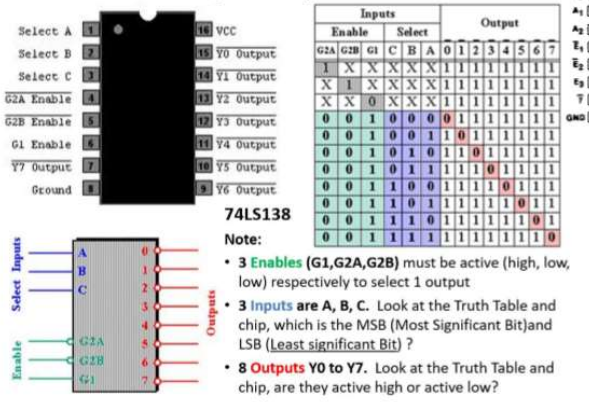
Step 2: Sum of Minterms equations for the Half Adder (get it from Part 1)

Sum = f (A, B) = ∑ m (1, 2)

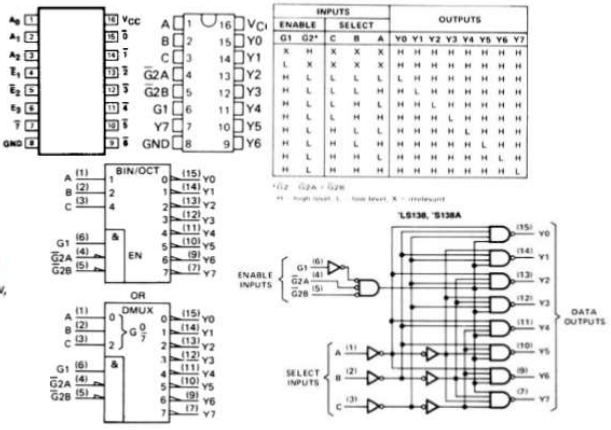
C_{out} = f (A, B) = ∑ m (3)

Step 3: Research how decoders work: Google 74138 and 74154 datasheets. Functional Diagrams, Pinout Diagrams, Truth Tables, Architecture below come from datasheets of different manufacturers but describe the same chip.

74138 3-to-8 Decoder/Demultiplexer



Inputs Enable G2A	G2B	G1	Select C	B	A	Output 0	1	2	3	4	5	6	7
1	X	X	X	X	X	1	1	1	1	1	1	1	1
X	1	X	X	X	X	1	1	1	1	1	1	1	1
X	X	0	X	X	X	1	1	1	1	1	1	1	1
0	0	1	0	0	0	0	1	1	1	1	1	1	1
0	0	1	0	0	1	1	0	1	1	1	1	1	1
0	0	1	0	1	0	1	1	0	1	1	1	1	1
0	0	1	0	1	1	1	1	1	0	1	1	1	1
0	0	1	1	0	0	1	1	1	1	0	1	1	1
0	0	1	1	0	1	1	1	1	1	1	0	1	1
0	0	1	1	1	0	1	1	1	1	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	1	0



74LS138

Note:

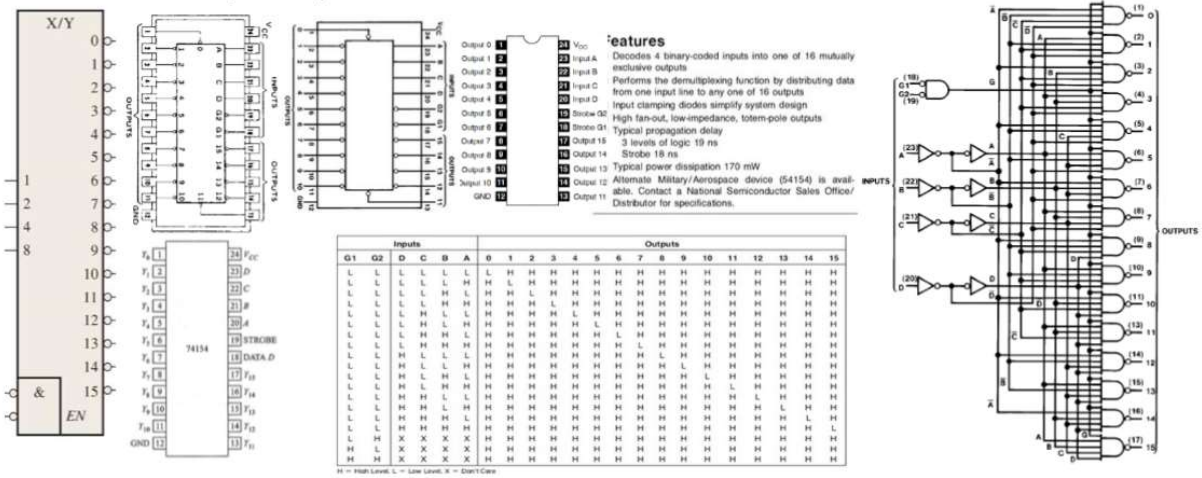
- **3 Enables (G1,G2A,G2B)** must be active (high, low) respectively to select 1 output
- **3 Inputs are A, B, C.** Look at the Truth Table and chip, which is the MSB (Most Significant Bit)and LSB (Least significant Bit) ?
- **8 Outputs Y0 to Y7.** Look at the Truth Table and chip, are they active high or active low?

Note: some diagrams use circles for active low input or output, some the triangle military symbol. Some use a bar over the pin variable some nothing – not even telling you if active low or high. Do not assume anything. Look at the truth tables, some use 0, 1 and X (don't care), others use L, H, and X. The name of the pin has nothing to do with the name of your circuit's input variables. In the half adder circuits, the switch inputs are A and B, the least significant input is B. Always look at the truth table to determine which input (or select) is the least significant (changes each row).

• CDA3201 • Intro to Logic Design • **Lab Assignment 2.3**

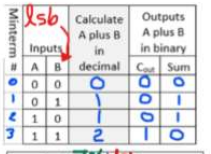
Name: **Grade:** /100%

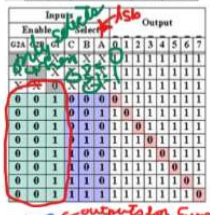
74154 4-to-16 Decoder/Demultiplexer



Note the different labels on the diagrams for the pins used to enable the chip to perform its functions (18 and 19): Strobe and Data D; Strobe G2 and Strobe G1; G2 and G1; and in the logic diagram it just groups them with one label: EN. A Strobe is defined as an input signal that disables or enables the chip. Enable pins often start with G. With a 1 on Strobe the input is ignored. Note all output pins usually start with Y. Note the input on the logic diagram uses the place value of the input pins, easily showing the one labeled 1 is the least significant input bit. By inspecting the truth table you can tell the output is active low (L) when selected, and the enables are active low, the chip functions when they are low.

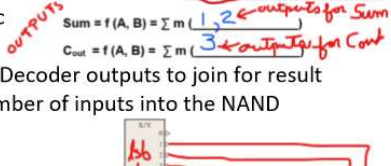
Watch this YouTube of how to wire a decoder and demonstrates how it works
https://www.youtube.com/watch?feature=player_embedded&v=o7XKXnxxUKk



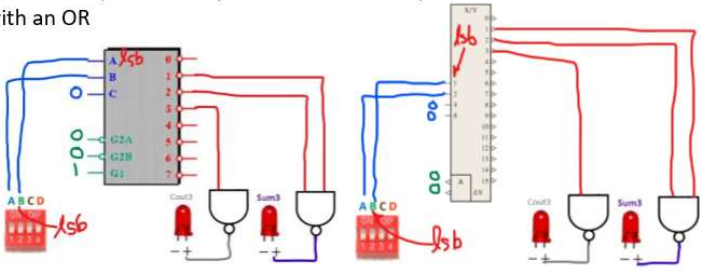


Step 4: Plan your circuit

1. Connect **Select** pins:
   From circuit truth table, find lsb (input column changes every row), find on input switch
   From truth table in the decoder chip datasheet, find the lsb
   Connect circuit switch input lsb to decoder's Select input lsb.
   Keep connecting next significant switch to next significant Select
   When run out of circuit inputs, connect unconnected Selects to ground
   Note all lines shown in blue to clearly show wires connected in this step,
   Use color code when actually wiring
2. Connect **Enable** pins: G2 active low (0), G1 active high (1)
   Look at datasheet, connect active low enables to ground, active high to Vcc
3. Connect **Output** pins:
   Look at Sum of Minterms or Truth Table to find where 1, this shows which Decoder outputs to join for result
   If decoder outputs are active low, join with a NAND, don't worry about number of inputs into the NAND
   If decoder outputs are active high, join with an OR
   Connect circuit output to output LEDs



Here is how it looks for the Half Adder
With either the 3-to-8 decoder or the
4-to-16 decoder



Name: **Grade:** /100%

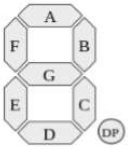
Step 5: Simulate with Quartus until matches Truth Table

You do not have to do this for the half adder, but you do have to do it for the full adder

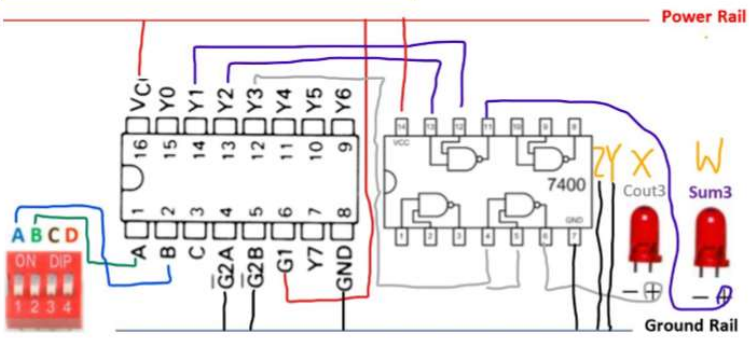
Type in the chip number for Quartus to find the chip where you type in the gate

Step 6: Plan the wiring

1. Find which decoders you have in your kit and cut and paste its pin out diagram
   Note: you may not have both types I have shown previously
   Note: the pin out I copied does not show the Y outputs active low but we know they are
2. Pick the NAND chip you will use, and cut and paste its pin out diagram
   One has 1 input, 1 has 2 input so pick a 2=input NAND Chip (7400)
3. Connect using the color code for inputs and outputs
4. Once you get it working can attach to the 7-segment display driver.
   You have a 2 bit result to display: Sum is the least significant, Cout the next
   In 7-segment circuit the W is least significant, so W=Sum, X=Cout, Y=0, Z=0
   In the full adder you have W=S0, X=S1, Y=S2, Z=S3, Probe=S4, we lose Cout



DO NOT WIRE ON BREADBOARD



Use this color code:

Inputs:
BLUE
GREEN
BROWN
ORANGE

In-Between:
WHITE
YELLOW

Outputs:
GREY
PURPLE

Power (Vcc):
RED

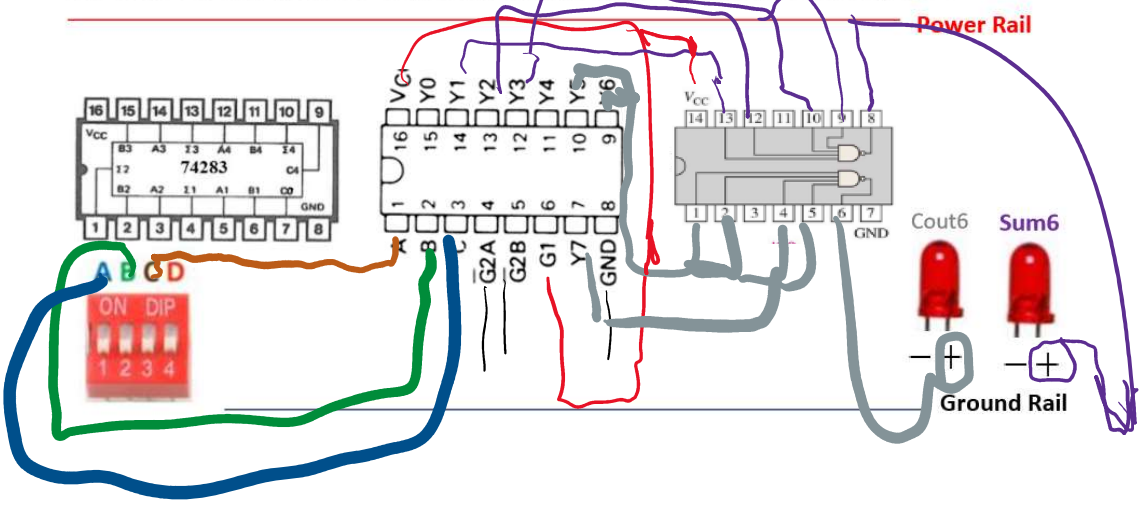
GND:
BLACK

Now it is your turn:

2.22 Plan the Circuit for the Full Adder

2.23 Simulate Quartus for Cout6 and Sum6 for the Full Adder verify matches Full Adder truth table

2.24 Plan the wiring for the Full Adder using Decoder and NAND chips in your kit

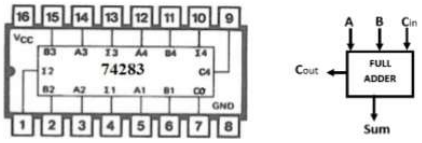


• CDA3201 • Intro to Logic Design • **Lab Assignment 2.3**

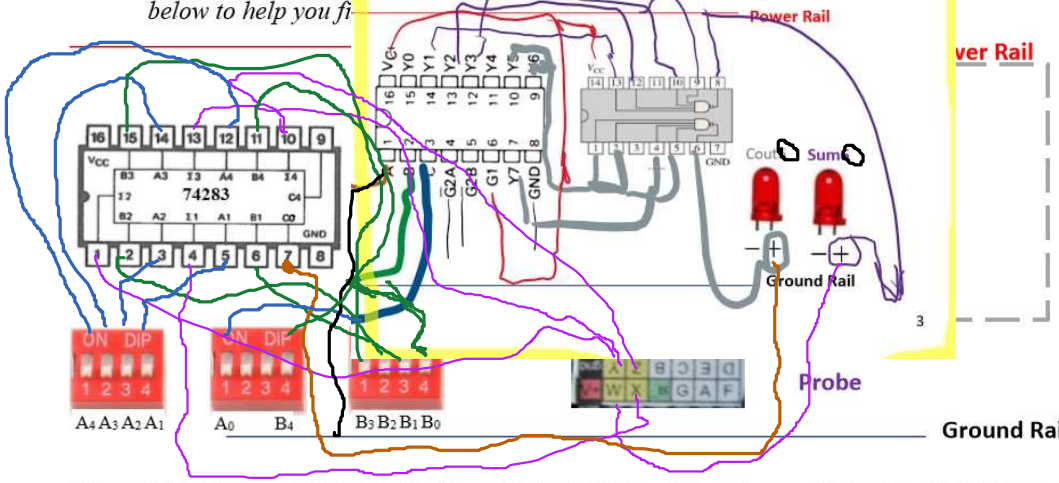
Name: **Grade:** /100%

2.25 Wire a 5-bit adder using your Full Adder circuit and the 74283 chip

First review 2.21b in Lab 2 Part 2 of how to connect a Full Adder circuit to a 4-bit adder to make a 5 bit adder:



You do not have to **Cout6 and Sum6 for the Full Adder** verify matches Full Adder truth table *ard, using the diagram*
below to help you fi **Full Adder** using Decoder and NAND chips in your kit



2.25 Do Video Demo testing with the following 4 additions. Take a picture of your board with your name showing

The 7-segment display will show the answer in hex S3 to S0 for the result and the S4 bit will be shown in the probe, 0 = green, 1-blue. We do not display the Cout. Do the following by hand to get ready and know what should be displayed. For the video, identify the chips and show your Sum 0 is connected to the W. You will set your switches for As and Bs in the order shown below and show and say the answer in the display and probe. Add the work for Part 3 to the Portfolio and submit

Base 2	Base 10	Base 16
01010		
+ 00101		
01111	15	F

Base 2	Base 10	Base 16
11		
00111		
+ 00110		
01101	13	D

Base 2	Base 10	Base 16
1 1		
01011		
+ 01001		
10100	20	14

Base 2	Base 10	Base 16
1 1 1		
01101		
+ 00011		
10000	16	10

There is no NANO lab for automatic testing of this lab.

Reflect. *You started designing a circuit that seemed trivial and unimportant, keeping track of the contents of a Piggy Bank. If you have errors, not much harm done. The Hackathon is going to take your adder and implement prototypes of solutions to real life problems that may cause potential damage or even loss of life if the design is implemented and deployed. Always do your best when designing circuits ensuring it is correct by simulating and verifying results agree with the specifications. Systems are built by putting together components. You never know where the component you designed will be used. Whether you design with NOT AND ORs, NANDs or Decoder depends if you need to minimize cost, time to prototype, the deadlines of completing your project and the components at hand. You may want to join IEEE Student Chapter and compete in the FAU Hackathon. Joining is inexpensive and gives you a permanent ieee.org email that you can keep for your professional life.*

4