CT Unit 4: Combinational Logic Circuit Design Project Group: Name:

Task: Binary-Coded-Decimal (BCD) Adder:

(1) problem description

In computer systems, *binary-coded decimal* (*BCD*) is a class of binary encodings of decimal numbers where each decimal digit is represented by four bits. The advantage of BCD is an ease of conversion into human-readable representations, in comparison to binary number systems. The BCD code is shown in the table.

The task is to design a *4-bit BCD Adder*. Use one *Hex Digit Display* to show the value of each operand. Use two *Hex Digit Displays* to show the sum.

The purpose of this project is to add single digit decimal numbers through binary numbers so that their sum will result in either a one or two digit decimal numbers through a Hex display. Basically, the challenge is to be able to convert a sum's hexadecimal number (A, B, C, D) into decimal numbers which is two digits.

(2) design methodology and process

First I set up two binary numbers that are to be added together. These numbers are between 0 and 9 in decimal form. To add them together, I used a 4-bit Full adder (the one we made for our logisim quiz) to add the binary number versions of the decimal numbers. The output is the sum, but there is also a carry out output for when the sum is greater than 16, because in hexadecimal, when a number exceeds 16 (10000), 16 is carried out because the base of the hexadecimal system is 16. Just like how in the decimal system, when a number exceeds 10 in the 1s digit, it's carried out to the 10s place because the base in the decimal system is 10. Now, depending on the sum, there are three different paths to take. First, if the sum is between 0 and 9, there is no carry out, and since the sum is just a single digit, the Hex display directly translates to a number such as 9.

However, if the sum is within 10 to 15, the sum must go a different path. And it's 10 because 10 is two digits, and 15, because it is the last number before 16, so there's no need to carry out yet. The sum must go a different way for numbers 10-15 because a sum like 13 in the Hex display will show D, since 13 is D in hexadecimal. However in this project, we want the hex display to show the decimal values 1 and 3. Therefore, to show 2 hex digits, we must add 6 to 13, because it equals 19, and for hex decimals, the 16 is carried out to the second digit so it's 1, and what's left is 3, which is the first digit. Therefore, 1 and 3 attach with each other, allowing the viewer to see it as 13, when the hexadecimal system actually sees it as (16 and 3 = 24). So, to create this process, so that it works for numbers 10-15, we move the sum into a comparator, which compares the sum to the number 9. If it's greater than 9, then the sum passes through the OR gate into a multiplexer containing a 6 or 0. Since the sum is greater than 0, the multiplexer allows the number 6 to pass through to another 4 bit adder which adds 6 to the initial sum. If the sum is less than 9, then it does not go through the multiplexer, it goes directly to the 4 bit adder since it doesn't need to add 6 to carry out a 16. For the Hex display that shows the carry out 1 or 0, there is a splitter that leads to it, because the splitter functions as a converter for bits. Throughout the project, the hexadecimals are converted to 4 bit numbers that are added together, but in order to show just numbers 1 and 0 in the hex display, we used the splitter to convert 4 bits into just 1 bit, either 1 or 0.

Finally, for sums greater than 15, which is 16-18, it cannot go through the same process as sums between 10 and 15, because sums greater than 15 require a carry out the moment the two inputs are added together. They can't go through the process of adding 6 immediately, because for example, with the sum of 18, 18 plus 6 is 24, and the output hex display will show a 4 for the lowest digit and an empty display for the higher digit. Therefore, from the very first 4 bit adder, the 16 must be carried out. Then, we are left with 2. We need the first digit hex display to be 8, so we must add 6. Therefore, for sums that require carry out, the next step is to directly connect it to the multiplexer where the 6 is activated and allowed to pass through to the final 4 bit adder. Then, it will appear on the lowest digit hex display as a single digit (8 because 2+6). For sums less than 16, the carry out will automatically be 0, because there will be nothing to carry out.

The purpose of the OR gate is to differentiate and give paths for sums between 10-15 and sums between 16-18. If the sum requires a carry out OR if the sum is greater than 9, then we can directly add 6 to the sum. For numbers 16-18, this is after the carry out occurs.

(3) logic circuits diagram (from *Logicsim*)
(4) simulation results.

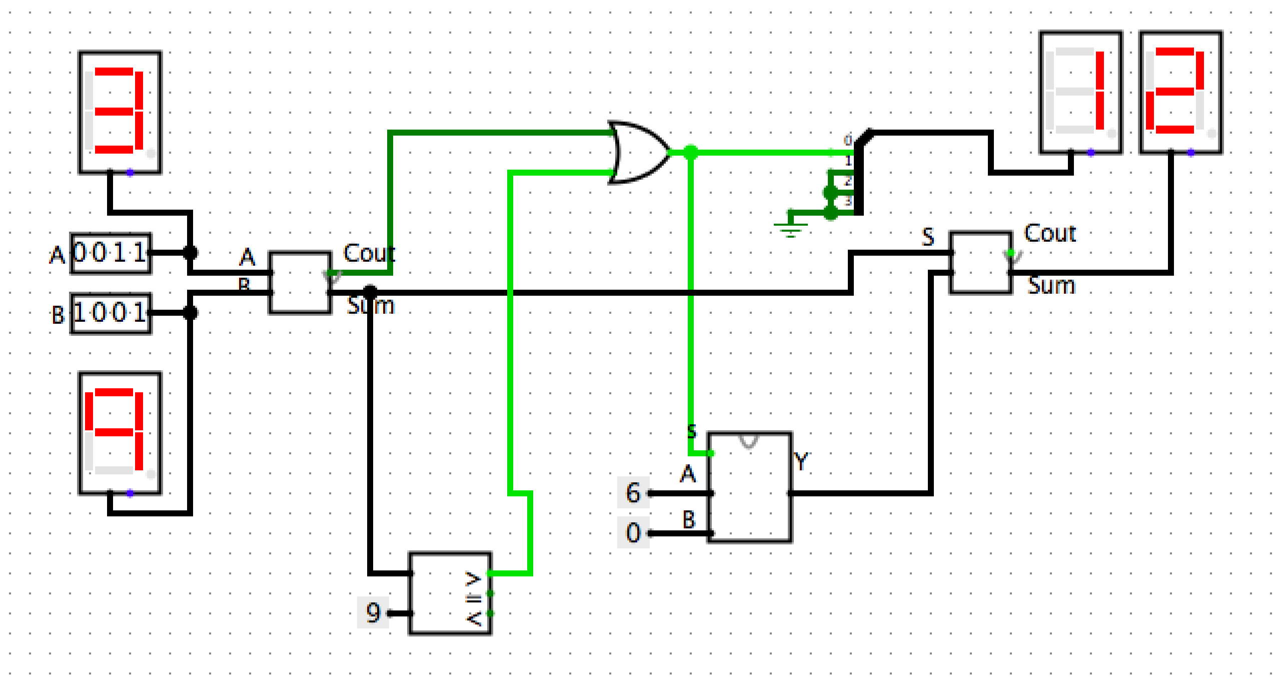

Diagram 1: Full system

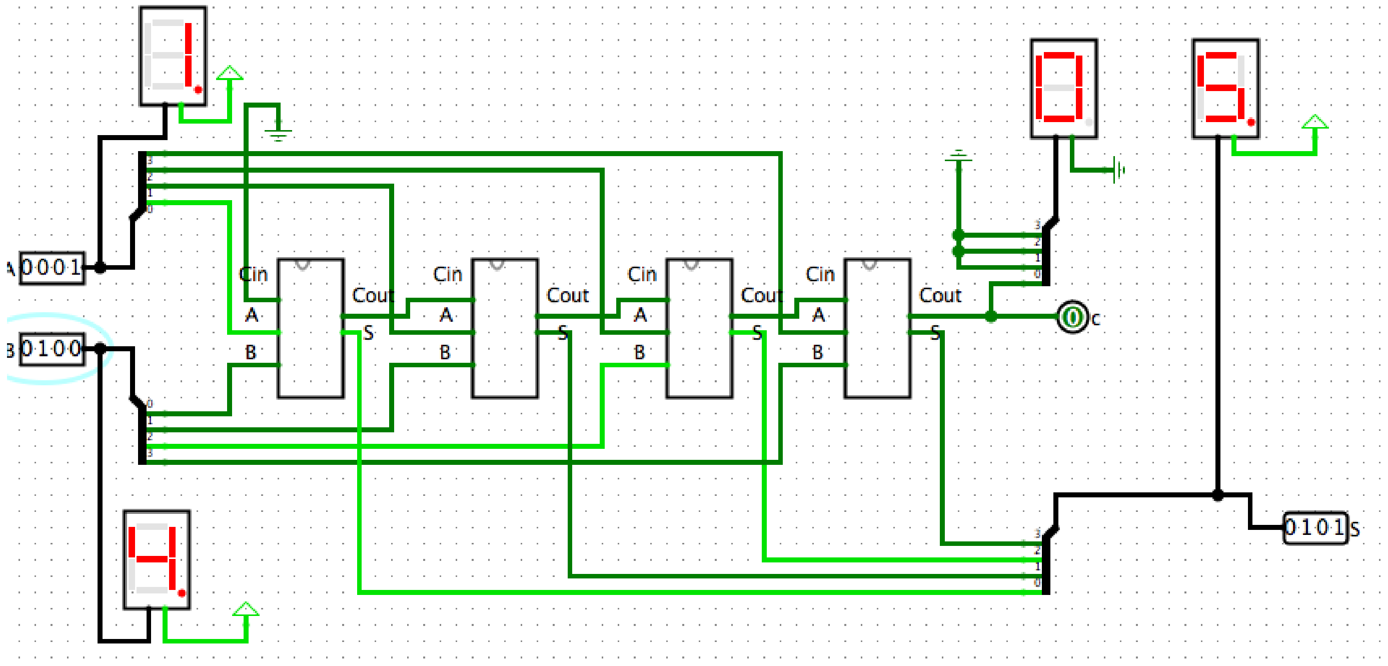

Diagram 2: 4 bit adder

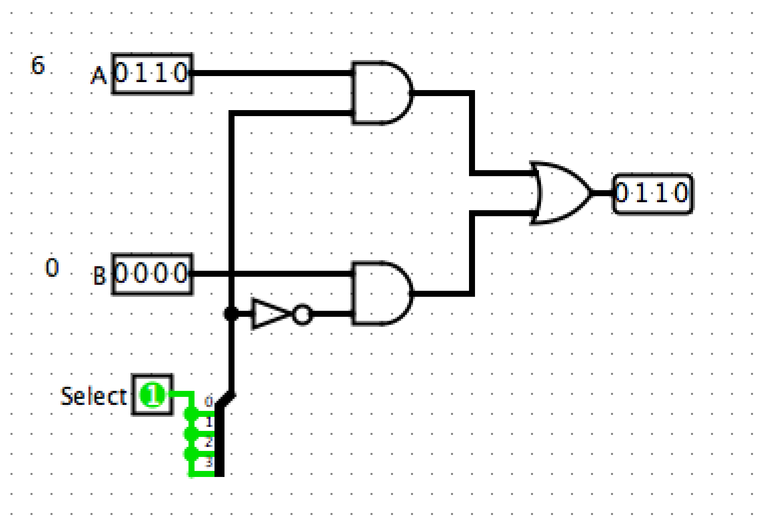

Diagram 3: Multiplexer

2. *Logisim* Project file (.circ).
https://drive.google.com/file/d/1X-HkO3fX9XE-UMUttX69DWWSQ7qpWdaW/view?usp=sharing

3. Project groups should be ready to demonstrate their logic circuits design and simulation to the class.

Ok Mr. Lin