Lab 2 Submission

**Half Adder (HA) & Full Adder (FA): Standard POS Form**

CPE 133 - 03

Michael Hegglin

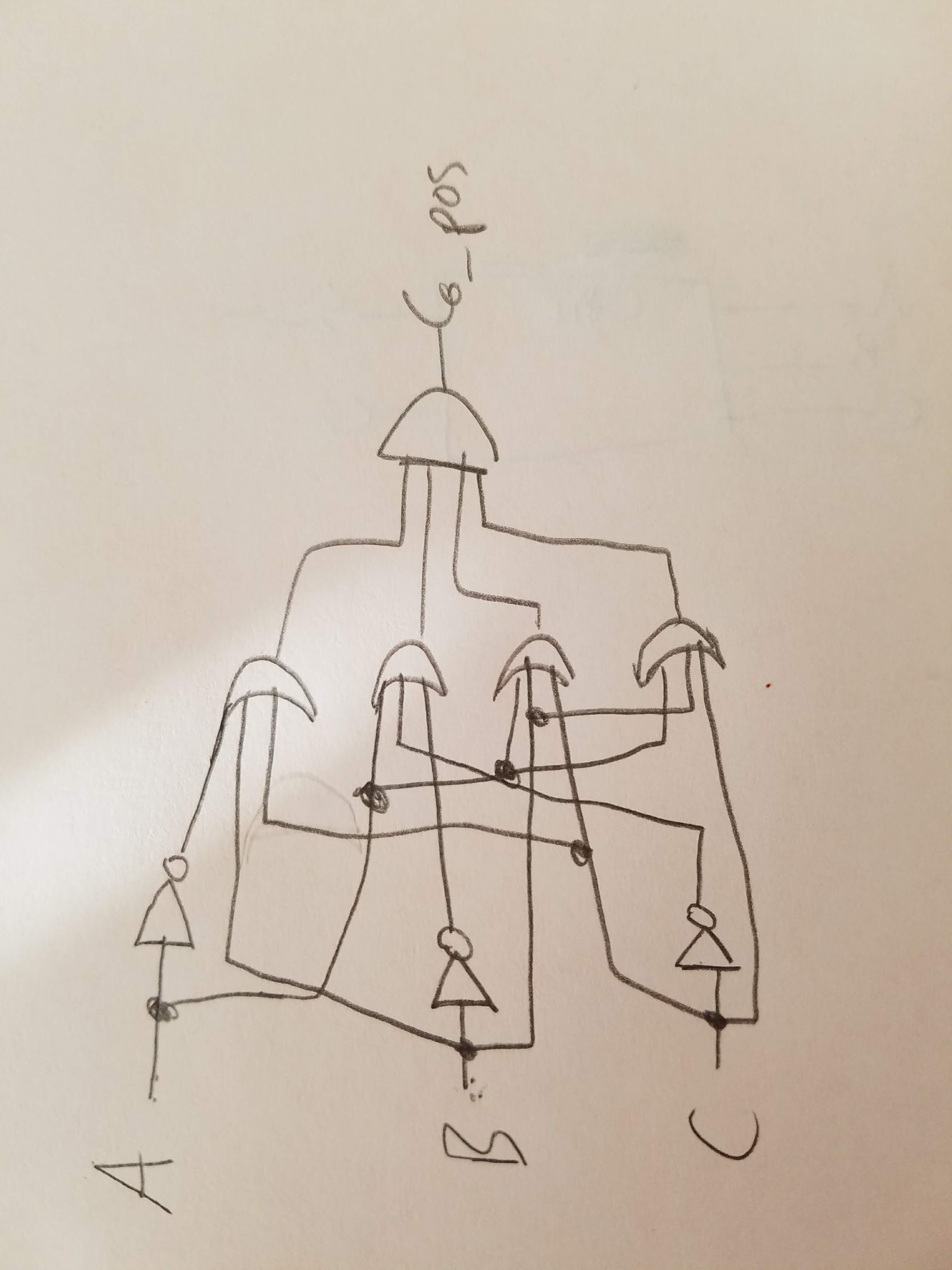
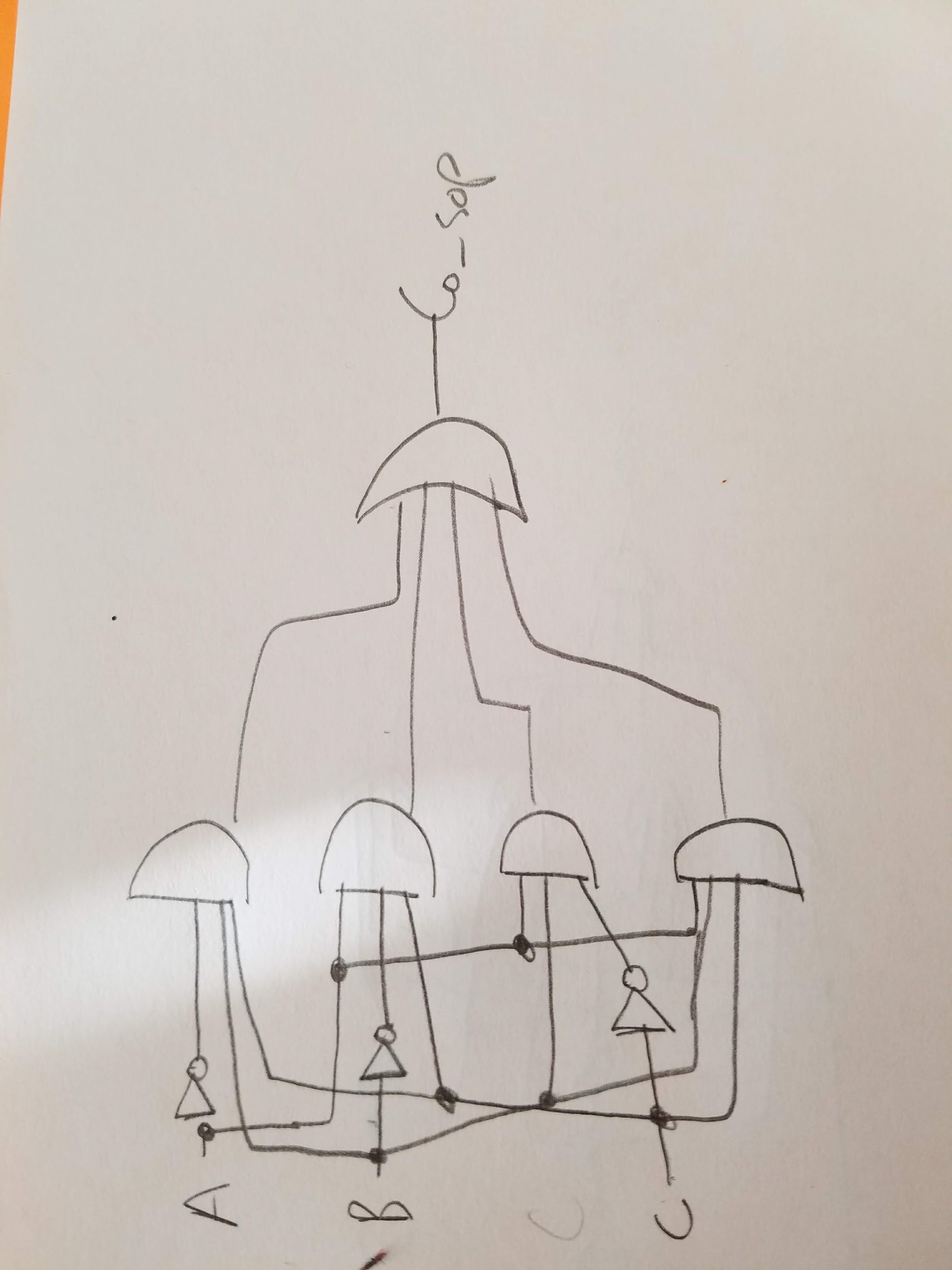
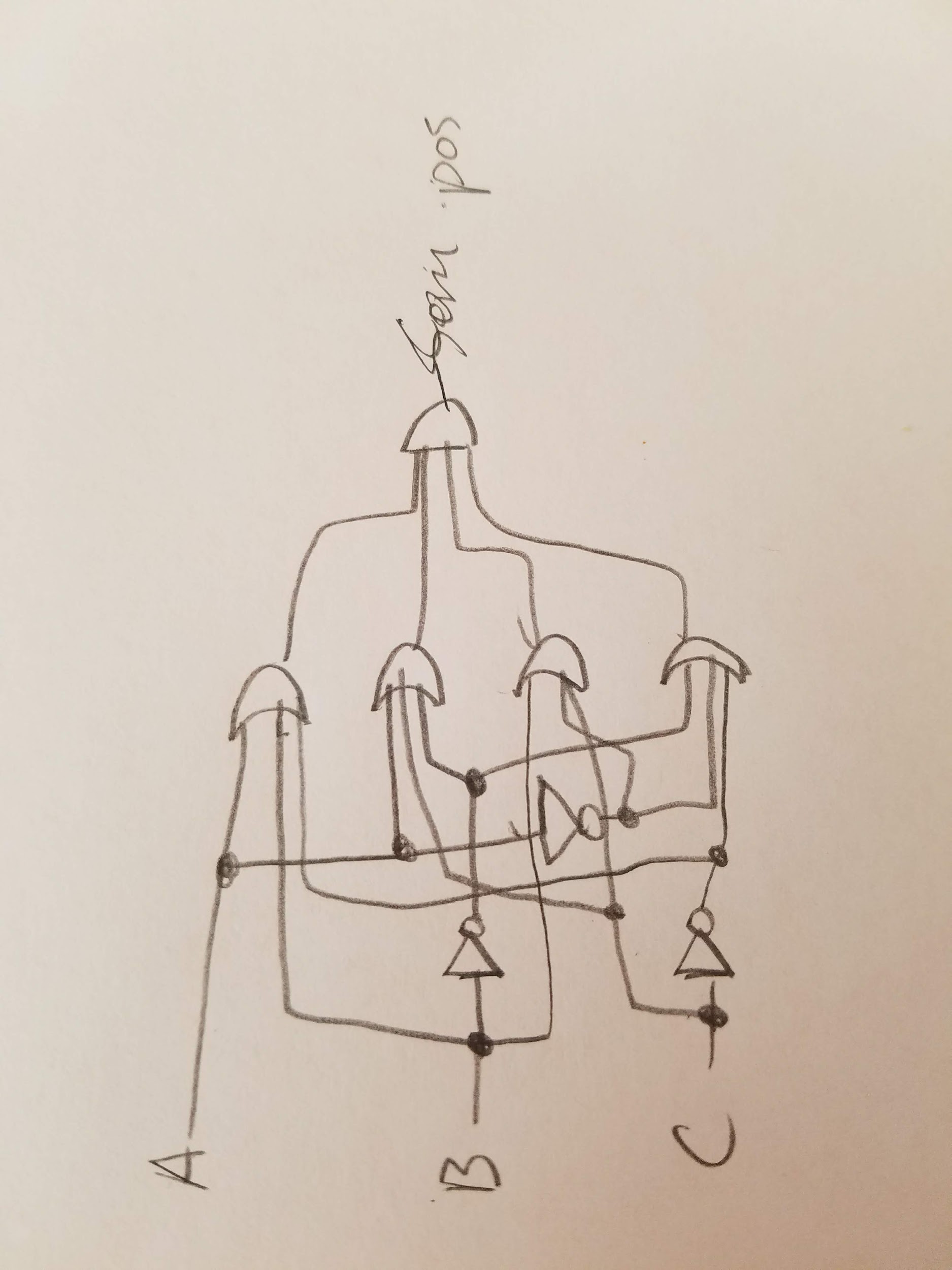
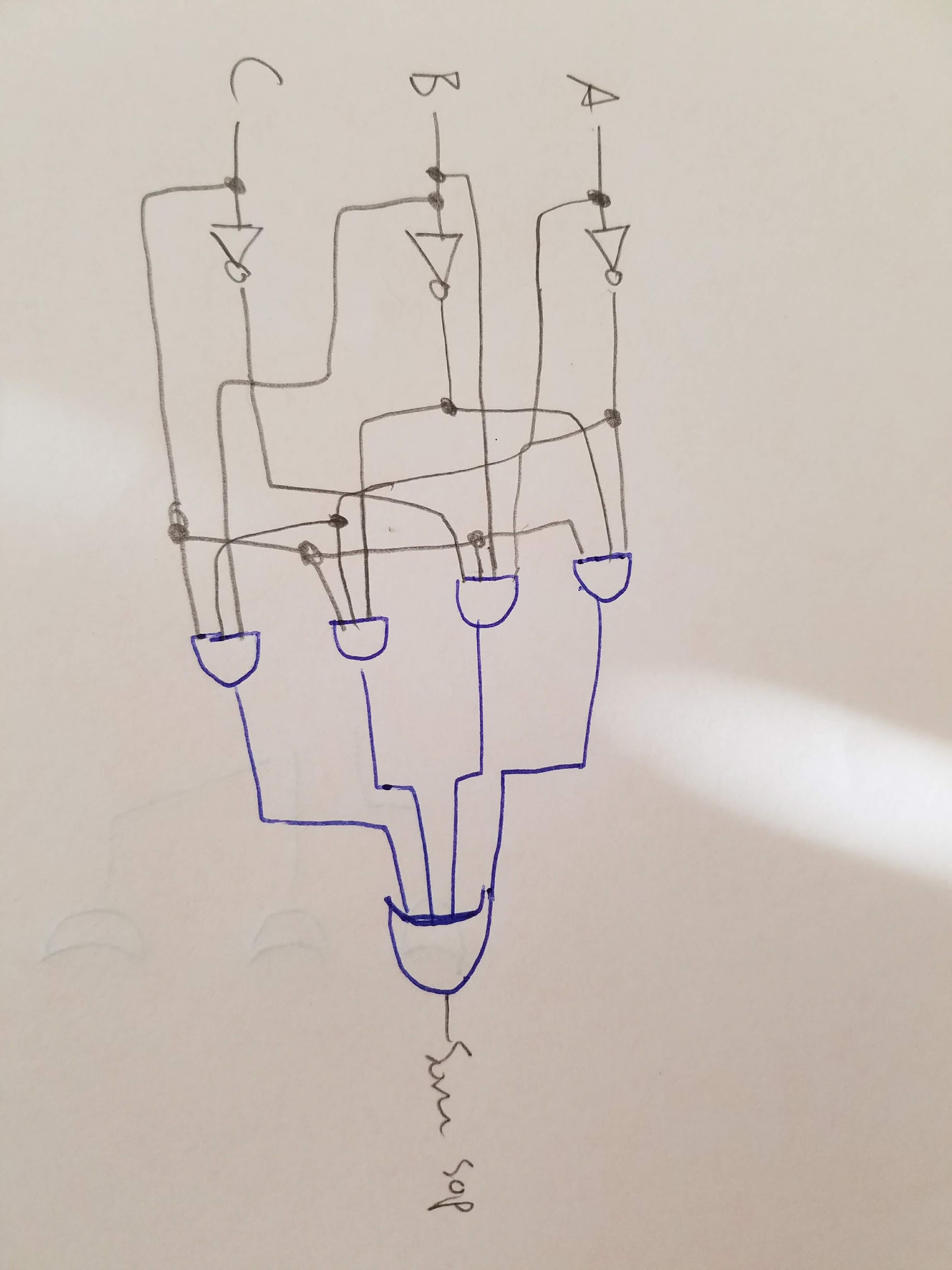
Jonathan Skelly

**Executive Summary**

The objectives of this lab were to design a full-adder, strengthen our knowledge of the software and hardware being used this quarter, and to prove the equivalence of SOP (Sum of Products) and POS (Product of Sums) Boolean forms. Using Verilog, we created a circuit that would take 3 one-bit inputs, add them, and output a one-bit sum and one-bit carry out using both SOP and POS Boolean forms. The circuit was uploaded to a Digilent board, where each input was tied to a switch, and each output would light an LED when its value was “1”. To test that the circuit worked correctly, the assigned switches were flipped. When one switch was flipped, the two LEDs above on the left side of the board would turn on, signifying equal “sum” outputs of “1” for both POS and SOP forms. When two switches were flipped, the four leftmost switches would turn on, signifying equal “sum” and “carryout” outputs of “1” for both SOP and POS forms. These results were correct, as verified by doing the Boolean equations by hand. This lab produced a fully functional full-adder, and taught through design how POS and SOP are equivalent Boolean forms.



Lab 2 Black Box Diagram

(Left to Right) Lower level diagrams of the equations for sum\_sop, sum\_pos, co\_sop, co\_pos

**Source:**

module full\_adder(

input a,

input b,

input c,

output sum\_sop,

output sum\_pos,

output co\_sop,

output co\_pos

);

assign sum\_sop = (~a & ~b & c) | (~a & b & ~c) | (a & ~b & ~c) | (a & b & c);

assign sum\_pos = (a | b | c) & (a | ~b | ~c) & (~a | b | ~c) & (~a | ~b | c);

assign co\_sop = (~a & b & c) | (a & ~b & c) | (a & b & ~c) | (a & b & c);

assign co\_pos = (a | b | c) & (a | b | ~c) & (a | ~b | c) & (~a | b | c);

endmodule

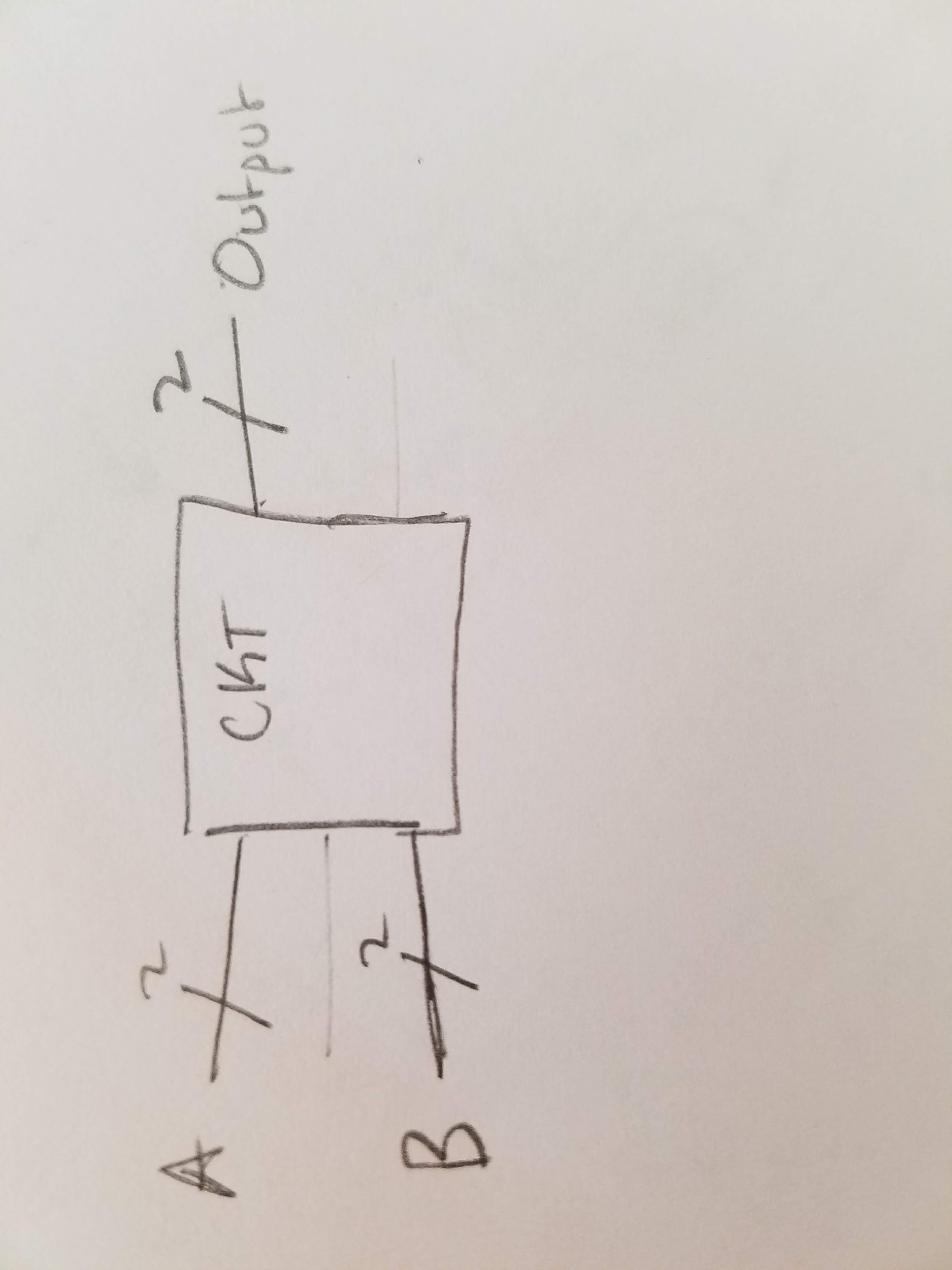
**Questions: (you told us we could skip putting the questions themselves this one time)**

1. The “best way” to implement a circuit with the current knowledge we have is to find the simplest answer to the solution, while still solving the problem completely. For example with a half-adder it easy to deduce that the Sum = ~A\*B + A\*~B by looking at a black box diagram. It is a simple SOP and easy to see and implement.
2. The SOP was easier to implement. This is due to the fact that it is easier to see/compute with mental math then POS is. For example, if A, B, and C are composed of variables and numbers, it is easier to multiply everything then add, instead of multiplying polynomials.
3. The full adder is limited in doing math operations due to the fact that there are only 3 one bit inputs. This means that larger computations containing greater inputs, like a 4 or 5 bit inputs are not possible.
4. By linking 2 full adders together a person could create a 2-bit adder. The carry-over from the previous adder feeds into the 2nd adder. However, this also means the 2nd adder must wait for the first adder to run and create a carry-over before running.
5. The equations are in the source code provided. All drawings are provided above.

|  |  |  |  |
| --- | --- | --- | --- |
|  | AND | OR | INVERTER |
| Co\_sop | 8 | 3 | 3 |
| Co\_pos | 3 | 8 | 3 |
| sum\_SOP | 8 | 3 | 6 |
| sum\_POS | 3 | 8 | 6 |

1. A person could visually see that our experiment was working through the LEDs that turned on according to the switches. When a single switch was thrown there would be 2 LEDs on, one that represented the sum, and one that represented the carry-over.

**Design Problems:**



Modulo-2 Black Box Diagram

**Modulo-2 Logic Table, two 2-bit inputs & one 2-bit output**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A1** | **B1** | **A2** | **B2** | **OUT1** | **OUT2** |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 |

Seven 2-bit modulo-2 adders could be circuited together to form an 8-bit modulo-2 adder with a 2-bit output..



8-bit Modulo-2 Black Box Diagram



Next Lower-Level diagram, with two 8-bit inputs (ABCDEFGH) and one 2-bit output (FIN\_OUT)