Simple Calculator

Jacob Arabbo, Anthony Barbera, Dante Gomez, Quinn Krause

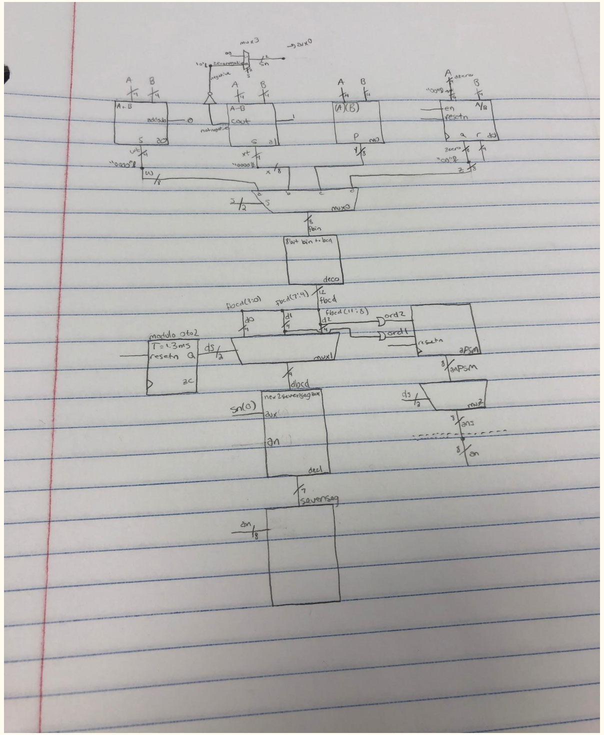
Idea/Process

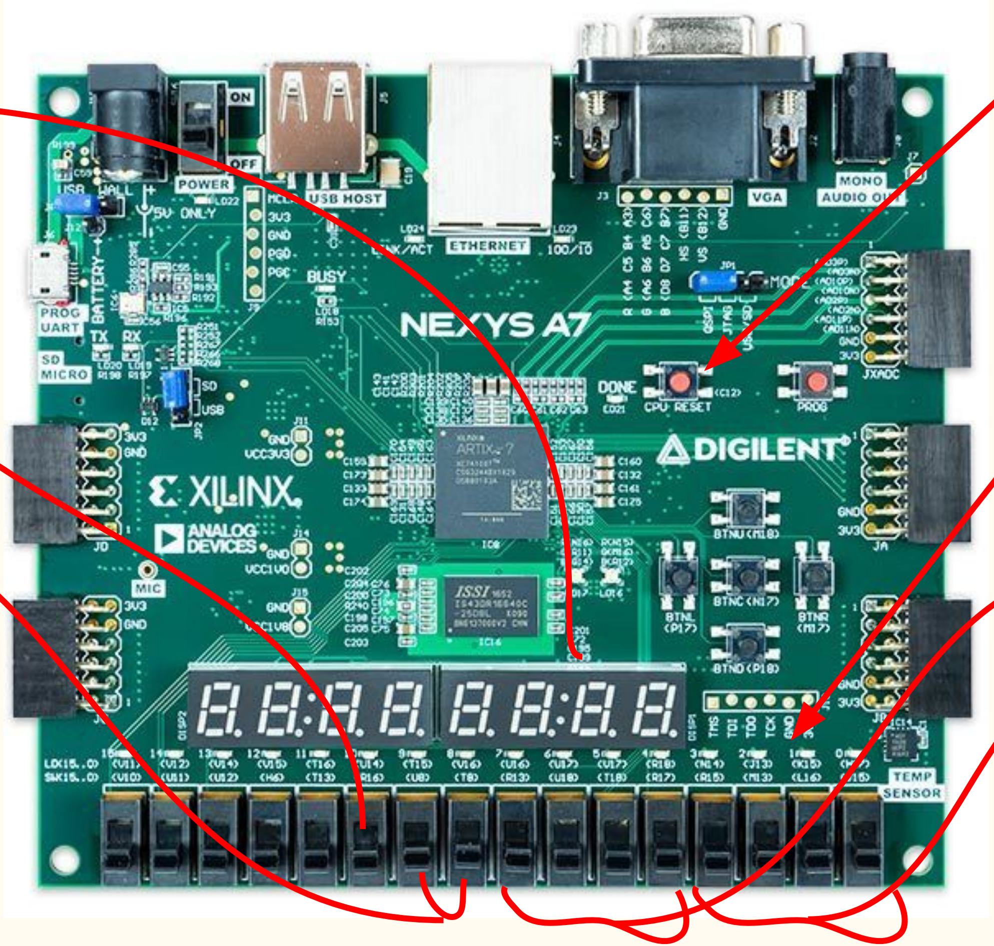
Create a simple unsigned calculator consisting of 4 basic operations.

Addition, Subtraction, Multiplication, Division.

If subtraction result is negative, we output --.

When dividing, remainder shown in LED's.





3 anode 7-segment displays

Enable

Selector

Reset button

Division Remainder in binary

Input A

Input B

Block Design

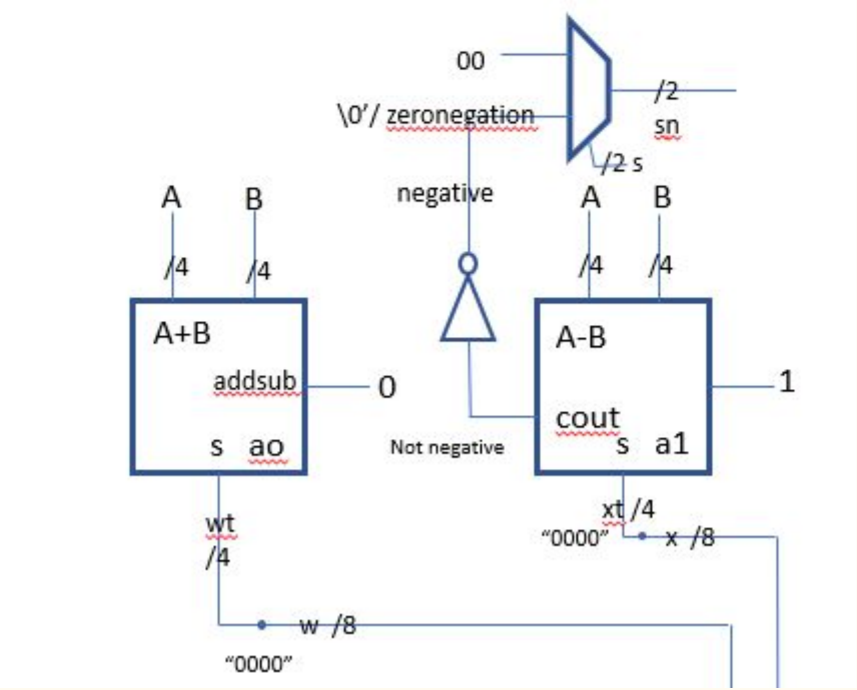
<u>4-Bit Adder/Subtractor</u>

<u>Adder</u>

- Adds the 4-bit values A and B
- Takes the 4 bit sum and adds "0000" to create an 8 bit value allowing it to work with the remainder of the calculator

<u>Subtractor</u>

- Takes A minus B in order.
- MUX takes the cout value to determine whether the output is going to be negative, then uses this output as an auxiliary input to the hex2sevenseg decoder.
  - Cout = 0 results in negative



Block Design (cont.)

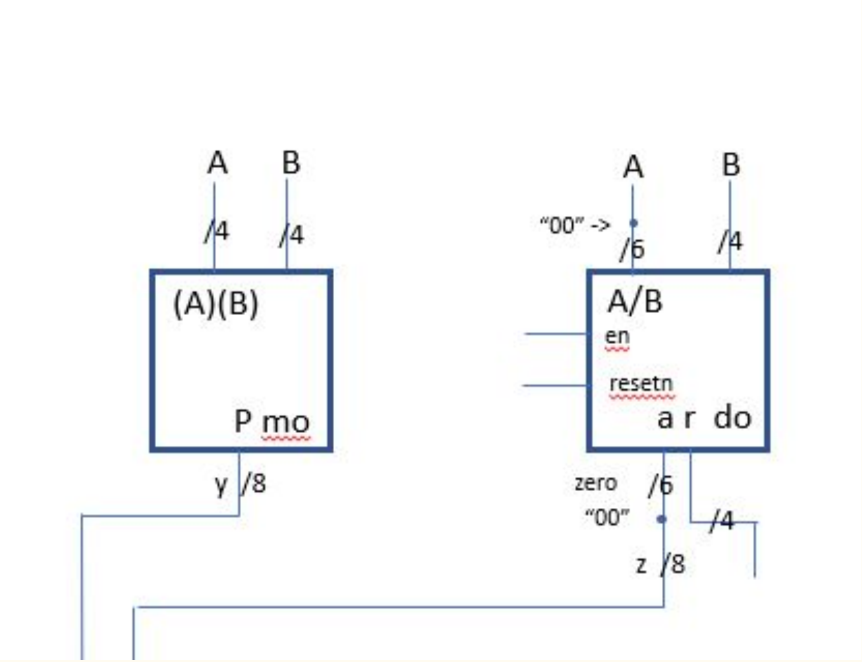
4 Bit Multiplier/Divider

Multiplier

- Takes A and B as inputs for both
- Uses AND gates to perform boolean multiplication for the two 4 bit inputs
- Outputs 8 bit result

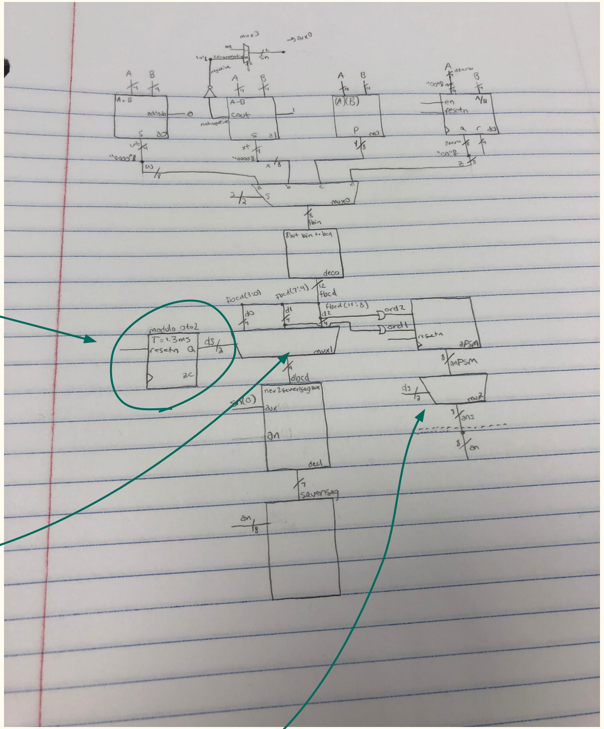
Divider

- Takes A divided by B in order
- Lab 6 based design
- Input from A is given leading zeros to produce 6 and bit values respectively



Block Design

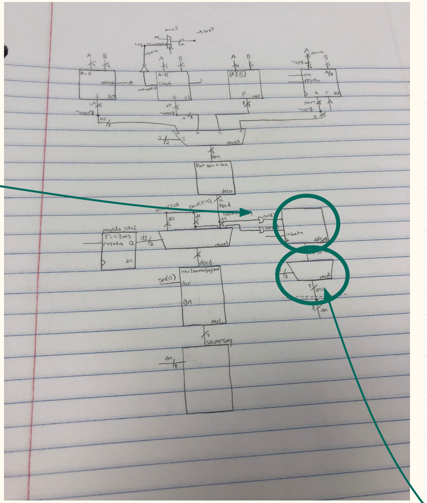
- Modulo Counter: 0 to 2
- Purpose: T~1.3ms, every tick increases the output by 1. The output is a 2 bit signal used for the select lines of the 2 MUXes that determine which digit and respective anode is to be active.

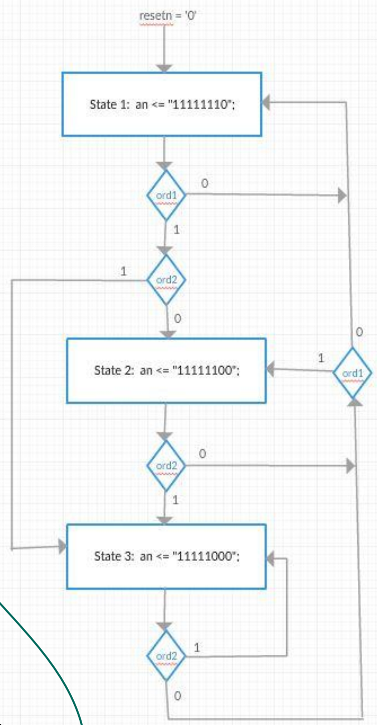


FSM

The FSM is used to control which anodes will be allowed to be on when the clock cycles through. We only want to display some zeros. For example: if the result of the calculation is 1 then we want to display ‘1’ instead of “001”. On the contrary, if the result is 100, then we want to display those trailing zeros. The FSM basically filters out unwanted leading zeros based on whether the more significant digits are nonzero or not.

The MUX following the FSM can be thought of as a filter for the anodes. As the select line changes value, the MUX will allow through one anode value at a time, whether it be ‘1’ or ‘0’.





resetn = '0'
State 1: an <= "11111110";
ord1
0
1
ord2
1
0
State 2: an <= "11111100";
1
ord1
0
ord2
0
1
State 3: an <= "11111000";
ord2
1
0

```
ord1 <= or digit1;
ord2 <= or digit2;
```

These variables determine whether there is a nonzero value in each of the more significant digits by ORing each bit of the digit together.

```
with s select
   ans <= anFSM(7 downto 3)&"11"&anFSM(0) when "00",
          anFSM(7 downto 3)&'1'&anFSM(1)&'1' when "01",
          anFSM(7 downto 2)&"11" when "10",
          "11111111" when others;
```

Demo

9+3=12

4+15=19

10-5=5

5-10=--

10x10=100

8x9=72

13/4=3 R1

9/5=1 R4