EE/CE 2310 Introduction to Digital Systems

Lab #3: Constructing Your Mini-Processor Design

December 4, 2018

Michael Kasman (CE) and James Pitts (EE)

I. Statement of the Learning Goal

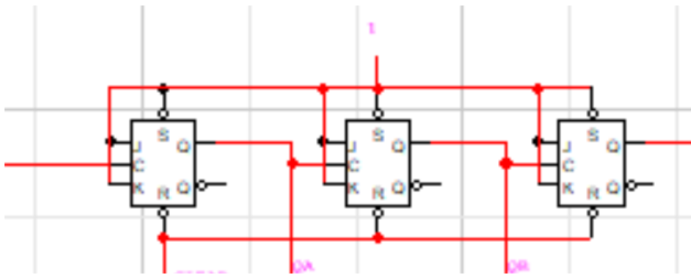
The EE/CE 2310 lab exercise will allow us to utilize our knowledge over the semester to create a mini-processor. Throughout the lab, we will be learning the process of building a mini-processor and its functionality. Thus, we will attain experience of building a 4-bit mini-CPU with the application of digital systems knowledge from the semester.

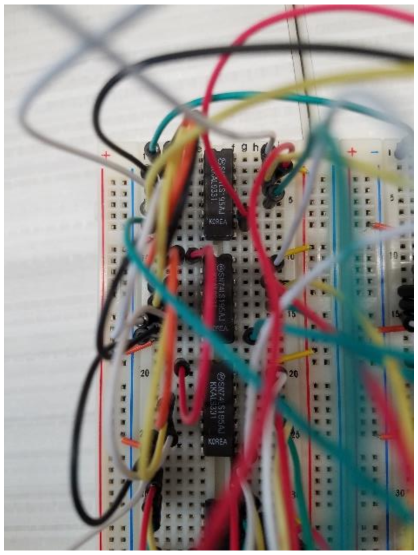
II. Changes to the Procedure

There were no changes to the written lab procedure.

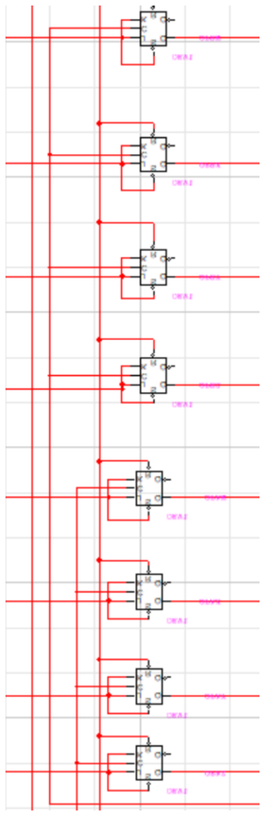
III. Building a Sequential Logic Circuit Questions

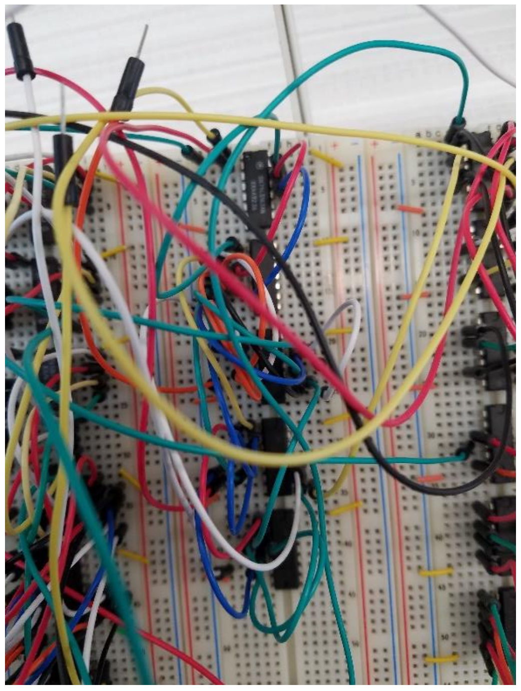
1. ***Using a 74163 4-bit counter, modify it to count modulo-8 continuously. It is reset at startup with Reset−. The three counter bits must be output to the decoder gates and also (initially) to the LED's. You will have to use a trick to make the 4-bit counter to a 3-bit counter. It would not normally be a good idea, as it makes the counter count slightly off (by a few microseconds) but in this case, it is okay.***



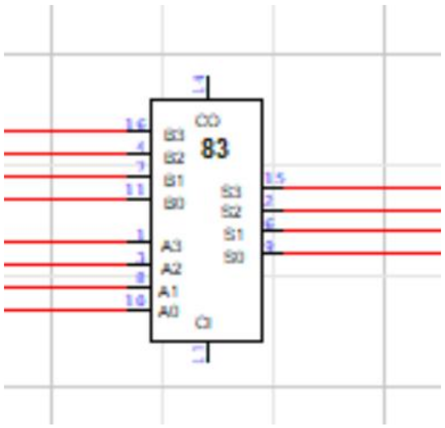


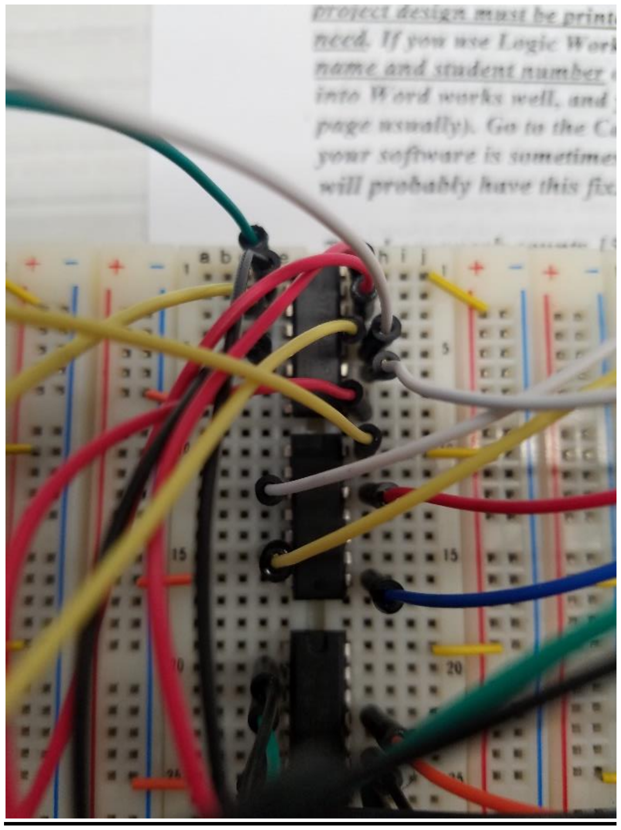
2. ***Use 74195 shift/storage registers as the 4-bit registers to hold the op code that is transmitted on count 1, Operand A that is transmitted on count 3, and Operand B that is true on count 4.***



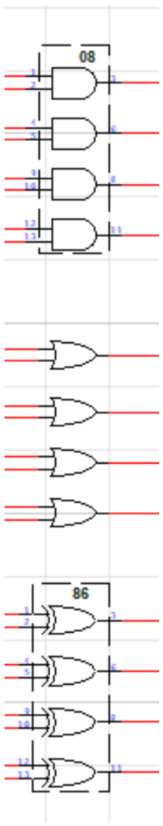


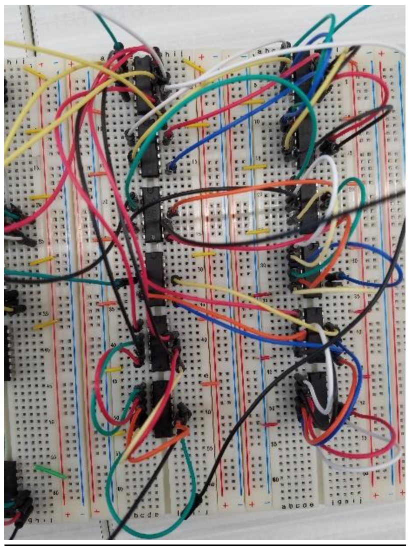
3. ***Use the 7483 4-bit adder chip (shown below) as the 4-bit adder to add the contents of the two registers. Since you are adding two 4-bit positive numbers (which are NOT 2's complement), that means that the entire range of operation for addition and subtraction is from 0 + 0 to 15 +15, for a total of +30 with a carry, maximum.***



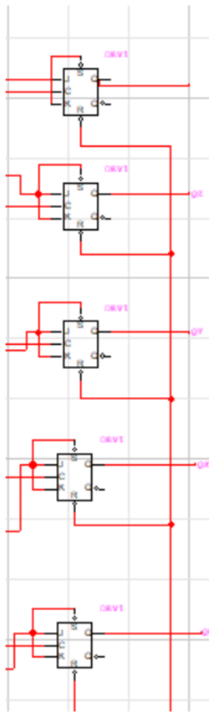


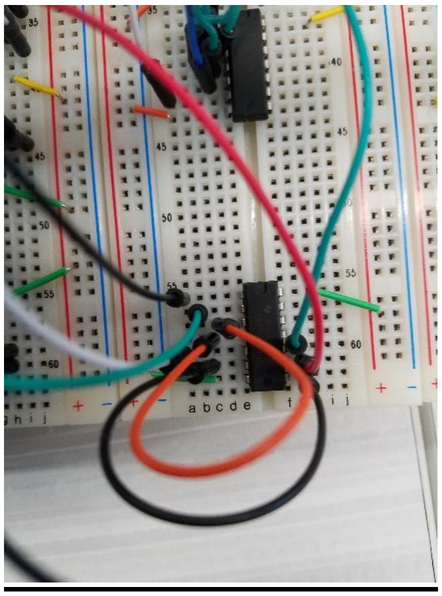
4. ***Design 4-bit AND, OR, and XOR circuits to perform those operations. Note that when you are doing logical operations, the 4-bit numbers are considered plain binary numbers, not 2's complement.***



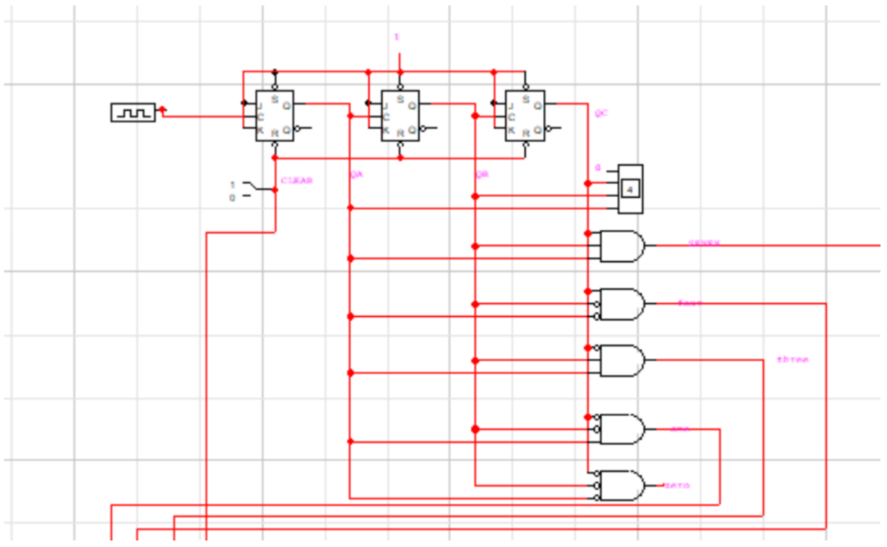


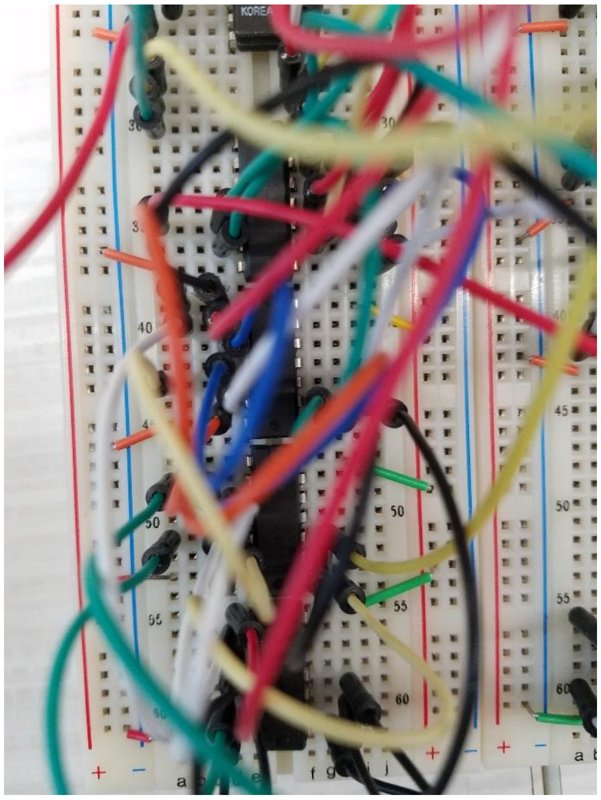
5. ***Design a 5-bit register to hold the results (labeled V', W', X', Y', and Z'). Adder output digits (Sum) use outputs W'-Z' and carry out (overflow) uses output V'. AND, OR, and XOR use output registers W'-Z'. Note that V' is only used to hold the Carry Out bit for the Add function. Count 7 is decoded and used as a clock to gate the answer into the output register.***



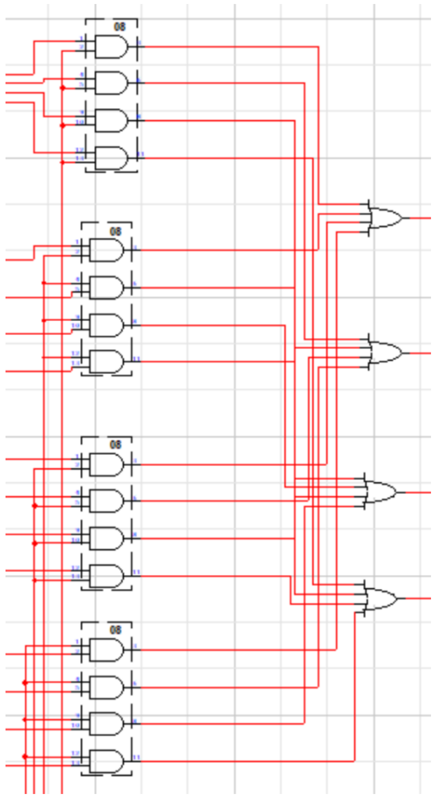


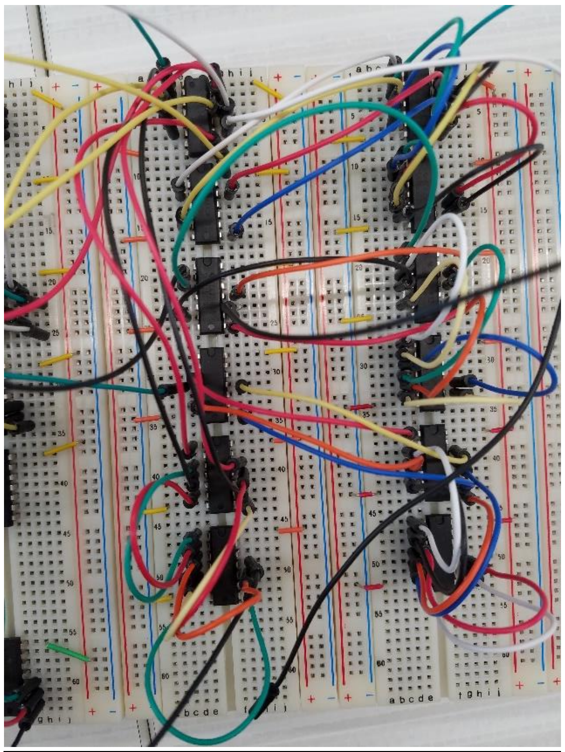
6. ***Build a timer circuit that decodes counts 0, 1, 3, 4, and 7 to generate the appropriate timing signals.***





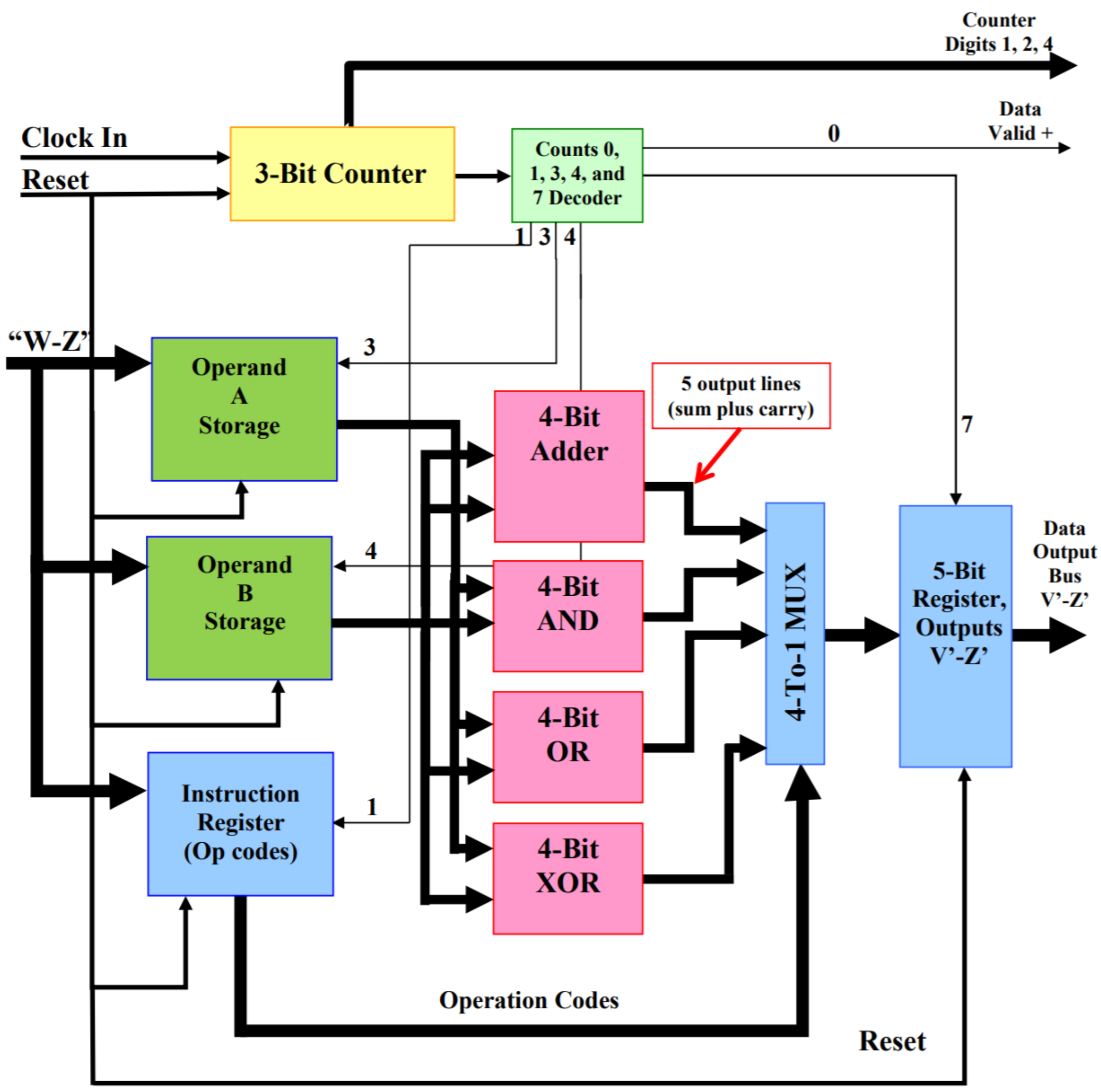
7. ***Build a MUX that chooses the correct result by decoding the Op codes 4, 6, 7, and 10 (0xa). This will allow storage of the correct result in the Output Register.***



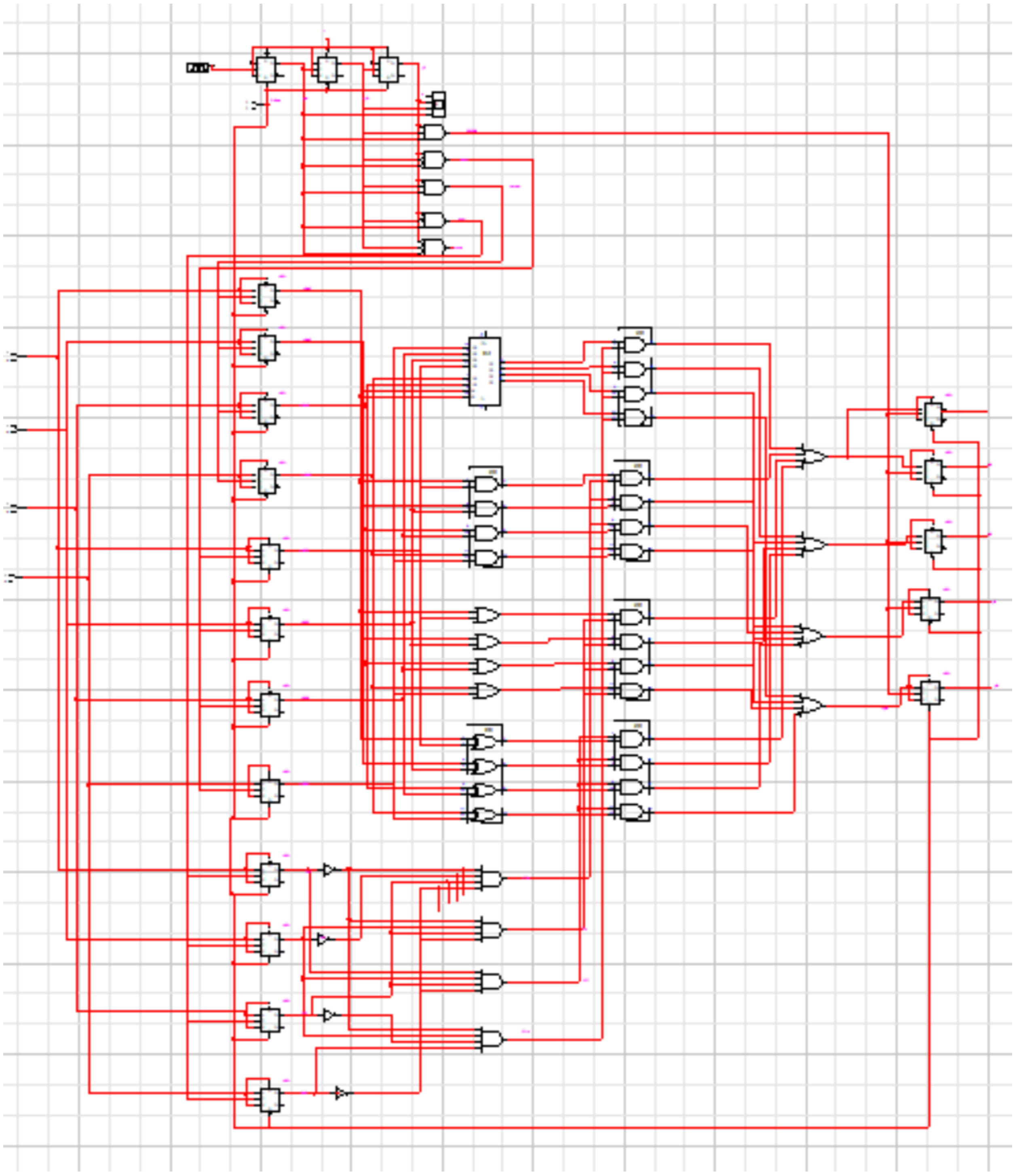


IV. Diagrams of Logic Circuits Assembled

Block Diagram of 4-Bit Processor



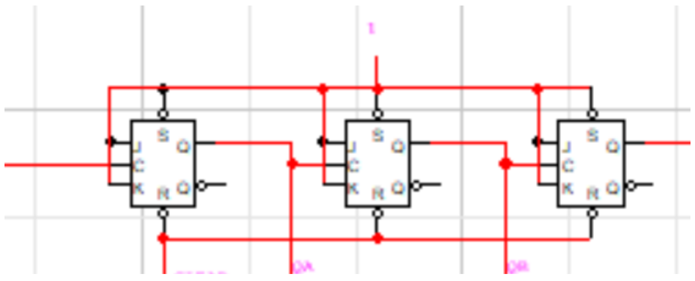
LogicWorks Schematic of 4-Bit Processor



V. Statements of Results

Design Problem 1: 3-Bit Counter Modulo-8

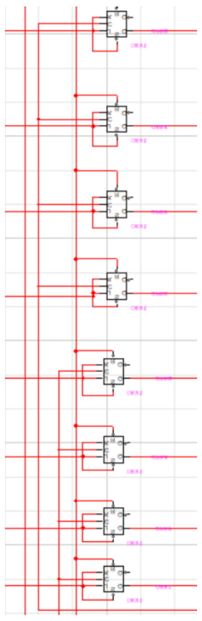
3-Bit Counter Modulo-8 Design



For the first design, we were tasked with creating a 3-bit counter modulo-8 with a 4-bit counter and one flip-flop connected to the clock and switches. Before creating the actual circuit on the breadboard, we designed the 3-bit counter modulo-8 on LogicWorks (Section 4) and labeled the variables (V, W, X, Y, Z). Then we constructed the designed circuit component on the breadboard by placing the chips vertically from each other; however, we did not match the wire colors for each input since there was an insufficient quantity of matching color wires in the kit for this experiment. Additionally, another issue we encountered was placing the wires in the wrong pins. The 4-bit counter chip has only one counter and the flip-flop chip only have one flip-flop. Despite these issues, we fixed our circuit component and the output matched the expected result of the design problem as the lab TA checked it.

Design Problem 2: Operand A & B Storage Registers

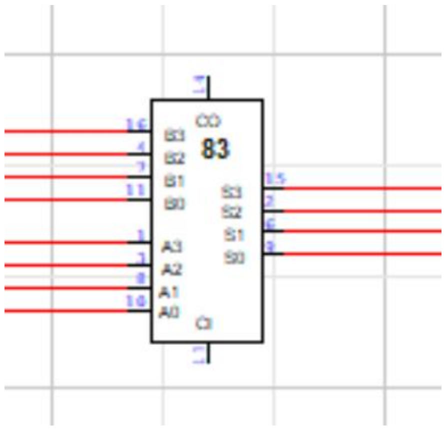
Operand A & B Design



For the second design, we were tasked with designing two 4-bit storage registers, which would store the op code that is transmitted on count 1, operand A that is transmitted on count 3, and operand B that is true on count 4. We designed how we were going to make the circuit component on LogicWorks first, before creating it on the breadboard. We set up the circuit the same way as design problem 1, by placing the chips vertically from each, except they are 4-bit registers instead of a 4-bit counter and flip-flop. Moreover, there was a different complication in this design problem. We had to take the outputs from the decoder and counter and input them into the storage registers, then output them to the adder, AND gates, OR gates, and XOR gates. Despite this complication, we were able to complete the storage registers and the output matched the expected result of the design problem as the lab TA checked it.

Design Problem 3: 4-Bit Adder

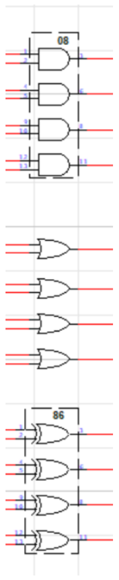
4-Bit Adder Design



For the third design, we were tasked with designing a 4-bit adder, which would add the contents of the two registers. We designed how we were going to make the circuit component on LogicWorks first, before creating it on the breadboard. We set up the circuit the same way as design problem 1, by placing the chips vertically from each other, except it is a 4-bit adder instead of a 4-bit counter and flip-flop. There were not many problems in building the 4-bit adder since we successfully inputted and added the contents of the storage registers and connected the outputs to the MUX. Hence, we were able to complete the 4-bit adder and the output matched the expected result of the design problem as the lab TA checked it.

Design Problem 4: 4-Bit AND, OR, and XOR Circuits

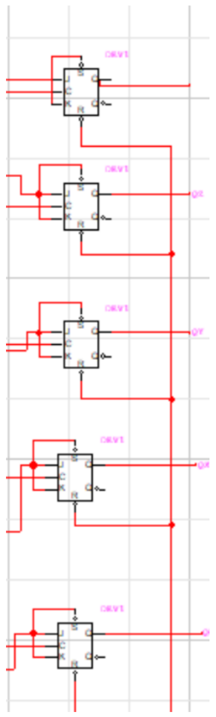
4-Bit AND, OR, and XOR Circuit Design



For the fourth design, we were tasked with designing the 4-bit AND, OR, and XOR circuit to perform the operations with the storage registers. We designed how we were going to make the circuit component on LogicWorks first, before creating it on the breadboard. We set up the circuit the same way as design problem 1, by placing the chips vertically from each other, except they are 4- bit AND, OR, and XOR chips, instead of a 4-bit counter and flip-flop. There were not many problems in building the gate circuit design since the design was alike the previous, 4-bit adder. We took the inputs and connected the outputs to the MUX similar to the 4-bit adder design. Hence, we were able to complete the 4-bit AND, OR, and XOR circuits and the output matched the expected result of the design problem as the lab TA checked it.

Design Problem 5: 5-Bit Register

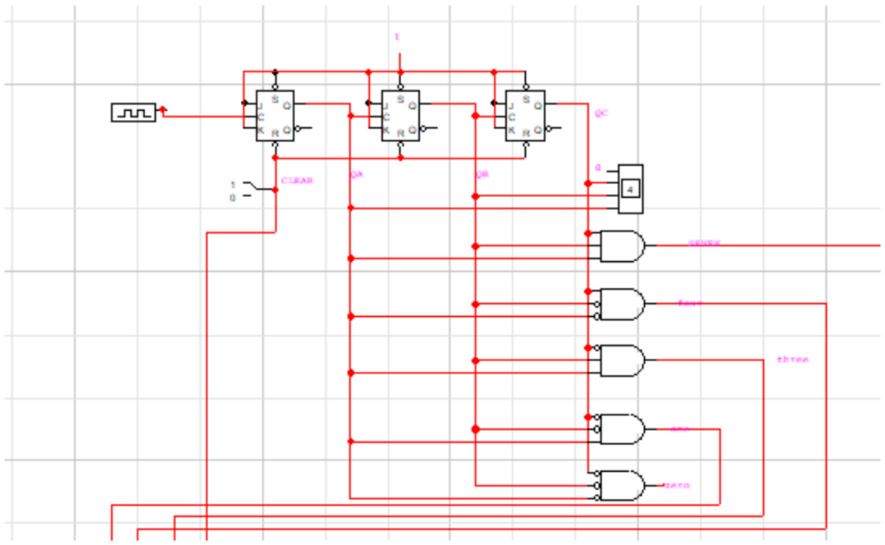
5-Bit Register Design



For the fifth design, we were tasked with designing a 5-bit register to hold the results from the adder and gates (AND, OR, XOR). We designed how we were going to make the circuit component on LogicWorks first, before creating it on the breadboard. We set up the circuit the same way as design problem 1, by placing the chips vertically from each other, except it is a 4-bit register and a flip-flop, instead of a 4-bit counter and flip-flop. The design as tricky since we did not have experience with building a 5-bit register, as a result the TA guided us to the correct build. We took the outputs from the MUX and adder carry to the 5-bit register. Hence, we were able to complete the 5-bit register and the output matched the expected result of the design problem as the lab TA checked it.

Design Problem 6: Decoder

Decoder Design



The sixth design initially came after the first one, we were tasked with designing a decoder that decodes counts 0, 1, 3, 4, and 7 to generate the appropriate timing signals. We designed how we were going to make the circuit component on LogicWorks first, before creating it on the breadboard. We set up the circuit the same way as design problem 1, by placing the chips vertically from each other, we added an AND chip and inverter chip to the counter, to conclude the outputs of the counter to the decoder. We experience the same problems as building the counter, connecting to the wrong pins. However, we were able to fix the mistakes and finish the decoder. Hence, we were able to complete the decoder and the output matched the expected result of the design problem as the lab TA checked it.

Design Problem 7: MUX

For the last design, we were tasked with designing a 4 to 1 MUX that chooses the correct result by decoding the op codes 4, 6, 7, and 10 (0xa). We designed how we were going to make the circuit component on LogicWorks first, before creating it on the breadboard. We set up the circuit the same way as design problem 1, by placing the chips vertically from each other, except there were 5 AND chips utilized, instead of a 4-bit counter and flip-flop. The build was simple since we were just connecting the inputs to the AND gates and outputting them to the 5-bit register. The only complication that occurred was organizing the wires since many of them intersected each other. We completed the MUX and the output matched the expected result of the design problem as the lab TA checked it.

VI. Discussion of Results

The circuit designs we were instructed to create in lab followed the same process of design. We had to first design the 4-bit mini-CPU circuit on LogicWorks and then create the circuit on the protoboard. If we did not follow this process, the outcome of the circuit will not match the correct results wanted, and it would be inefficient to not record the designs. Through creating the circuit on the breadboard, we encountered a problem that occurred in both design problems. There was a lot of wiring necessary to connect the chips together; thus, the wires sometimes were not connected to the correct pins, allowing for error in our design. Furthermore, some design problems, more specifically the 5-bit register, had a different error than design problem 1, where we sought help from the TA and the internet to complete the build since we had no experience in building a 5-bit register. Other than connecting some of the wires to the wrong pins, we had no other discrepancies and the results matched the expected outcomes. Both my partner and I learned that recording designs and seeking aid is important to developing a successful project such as the 4-bit mini CPU.

VII. Summary and Conclusions

The lab exercise allowed us to apply conceptual knowledge about combinational logic, sequential logic circuits, and CPU components over the semester to build a functioning mini-processor and see it light up the LEDs to check that they are correct. Not only did we apply conceptual knowledge, but we understood the necessity to have a design process before creating the 4-bit mini-CPU. It is essential to design the circuit on LogicWorks. Moreover, through the lab process, we learned that there are various potential mistakes we can make it if we do not perform the lab carefully. For example, since we were not careful enough on a minority of design problems, we connected a wire to a wrong pin, which led to the entire circuit not functioning correctly.

Through this mistake we experienced the phenomena of design a CPU, all components must work before building the next component. Therefore, the lab exercise allowed us to demonstrate our digital systems knowledge from over the semester and familiarize ourselves with building a 4-bit mini- CPU and learn about common mistakes in creating such a system.