Lab Report 0

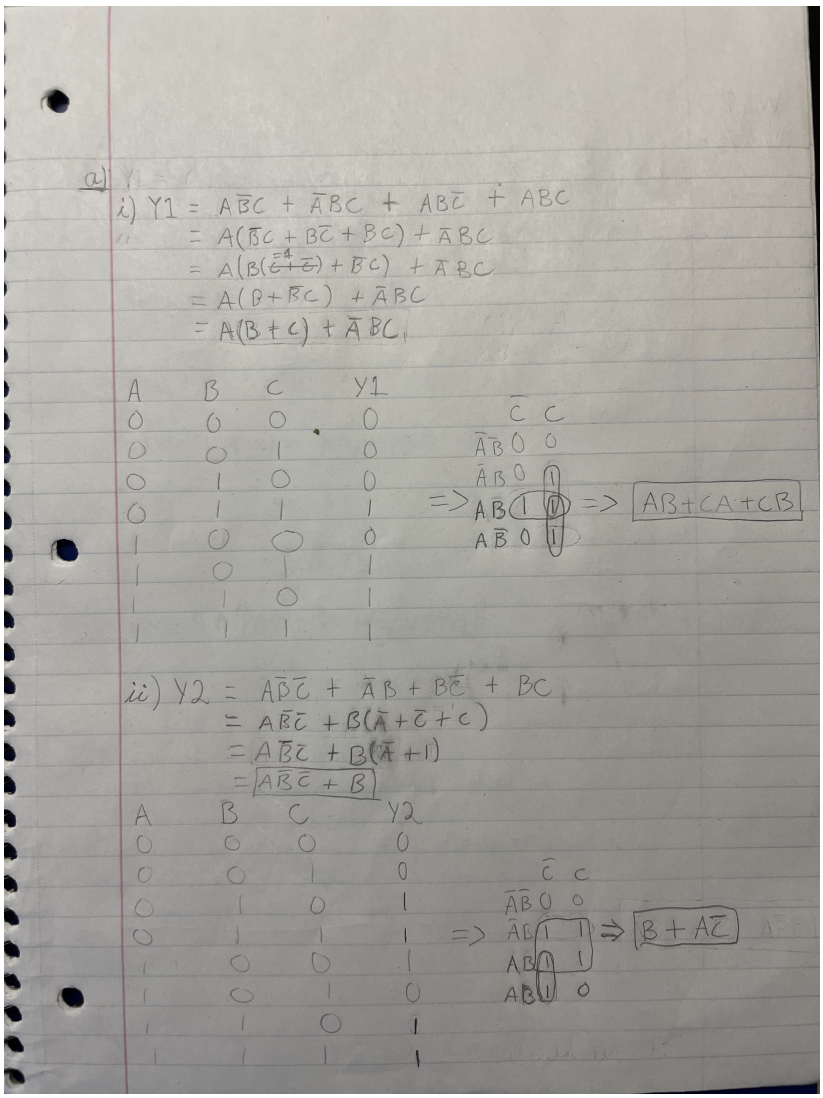
EEL3701C - Digital Logic & Computer Systems

Benjamin Ruddy
UFID 33609976

Pre-lab

Pre-lab Questions + Homework

a) Boolean Equation Minimization:



b) Implementation of minimized equation Y1:

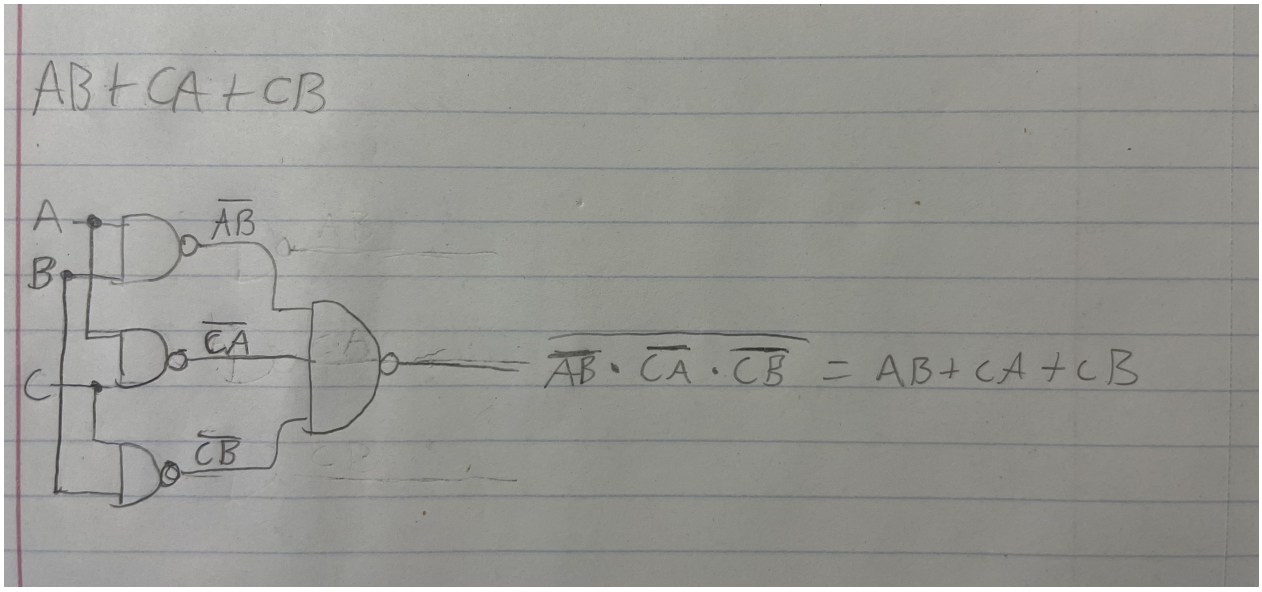


Figure 1: Schematic of minimized Y1

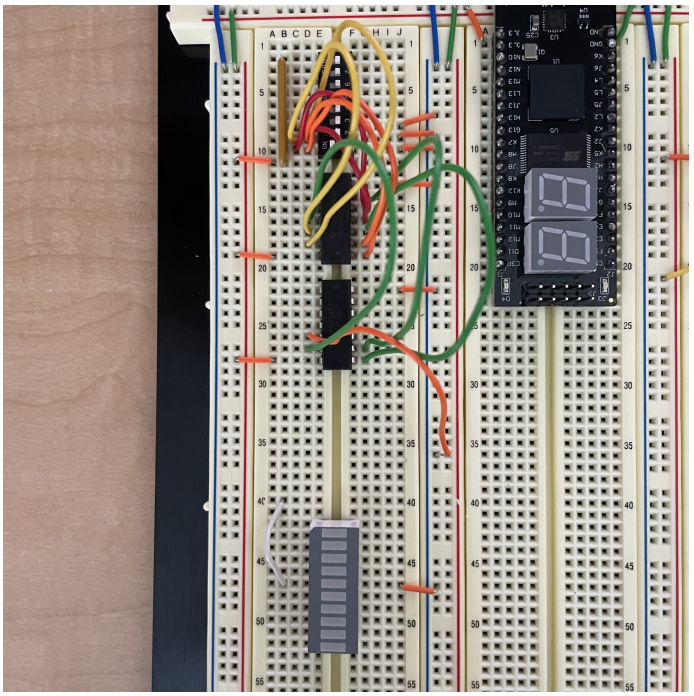


Figure 2: Implementation of equation Y1 with NAND gates (checked off in lab)

c) Implementation of minimized equation Y2:

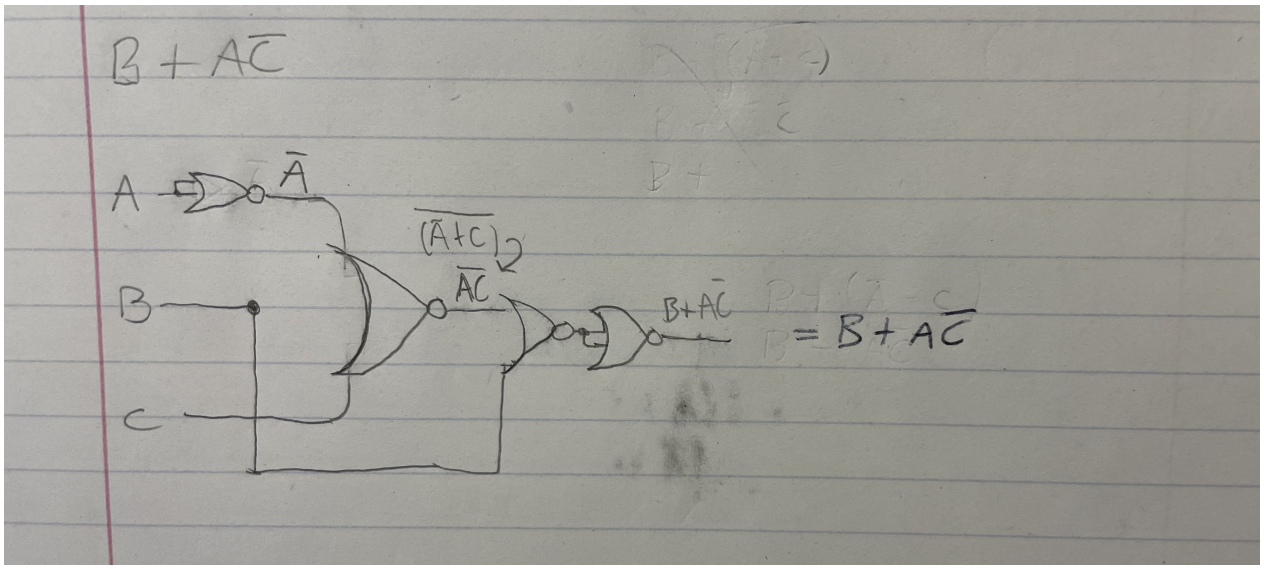


Figure 3: Schematic of minimized Y2

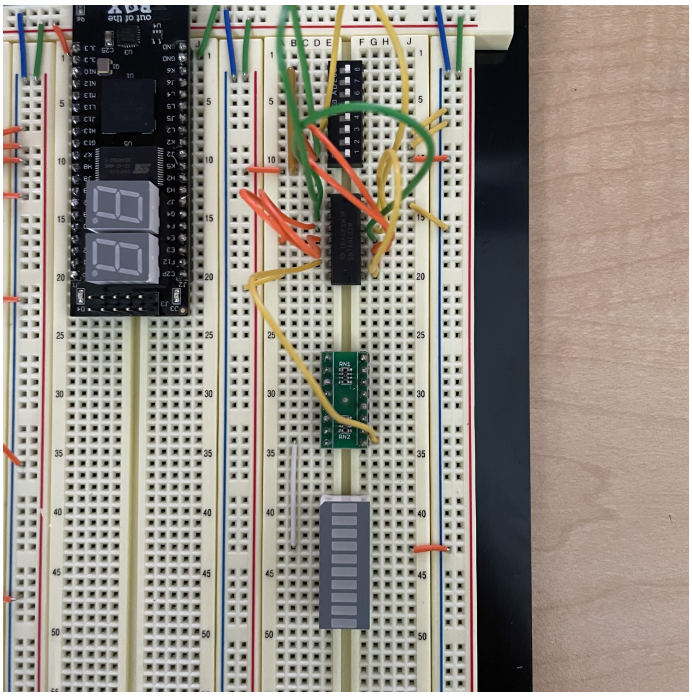


Figure 4: Implementation of equation Y2 with NOR gates (checked off in lab)

d) Installed and set up Quartus ✓

e) Implementation of Y1 and Y2 using NAND gates in Quartus:

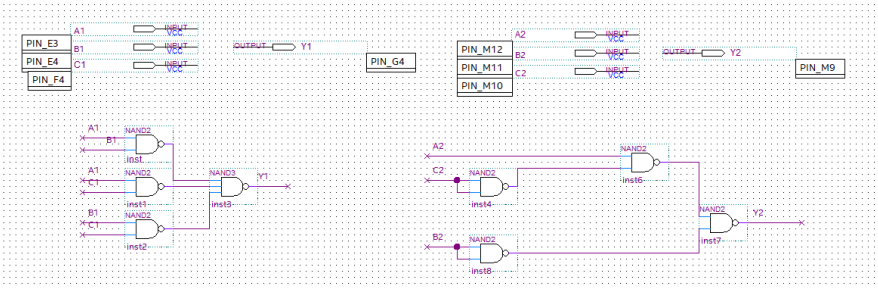


Figure 5: NAND BDF representation of Y1 and Y2

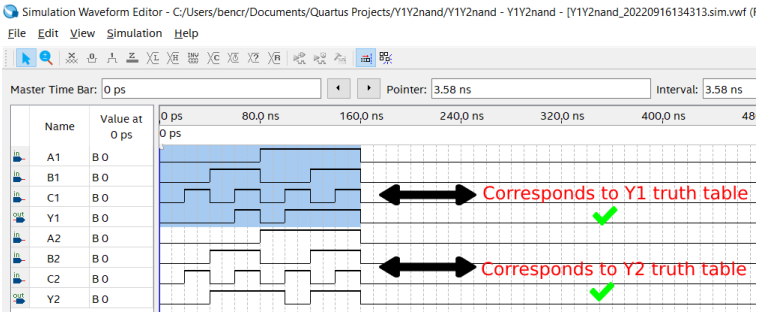


Figure 6: Simulation of the above BDF design (Low represents 0, high represents 1)

f) Implementation of Y1 and Y2 using NOR gates in Quartus:

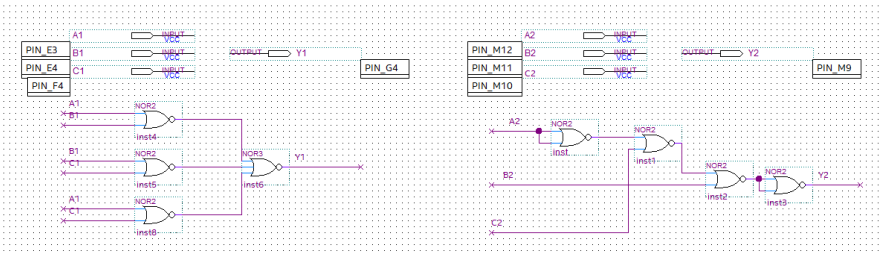


Figure 7: NOR BDF representation of Y1 and Y2

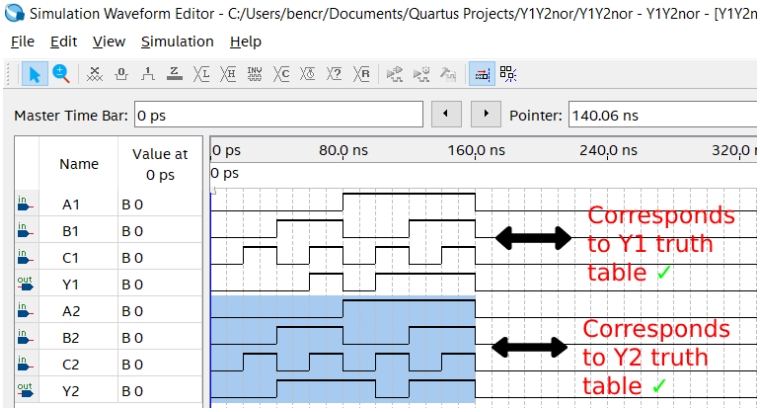


Figure 8: Simulation of the above BDF design (Low represents 0, high represents 1)

g) MXDB implementation of part (f):

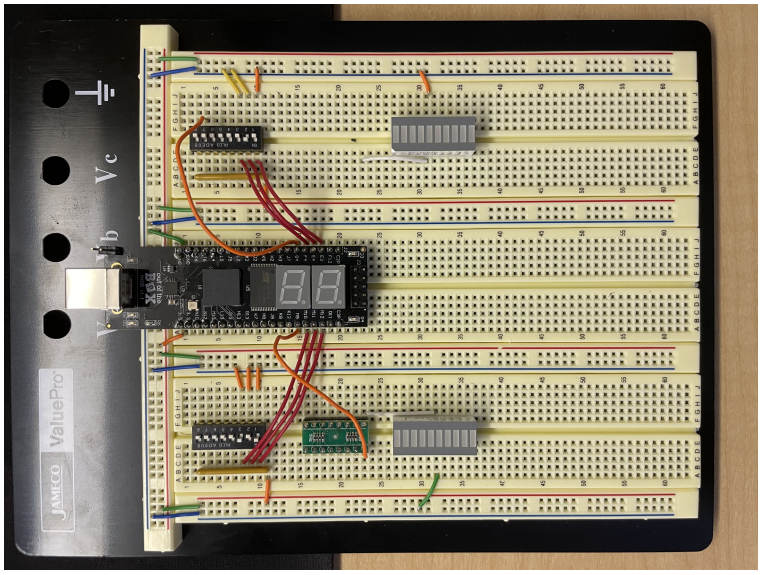


Figure 9: Breadboard after part (f) has been programmed onto the MXDB (checked off in lab)

h) Demonstration of parts C and G (verified by TA in lab) ✓

Pre-lab Design and Implementation

The design consisted of the schematics in figures (1), (3), (5), and (7), which were arrived at after minimizing equations Y1 and Y2, and translating them over to their minimized forms using only NAND and NOR gates. In this first part of the process, DeMorgan's law was of special importance as it allowed us to undo the negation of NAND and NOR gates as well as change from OR to AND or vice versa, when needed. The implementation of the pre-lab consisted of both using the gates within physical ICs, as well as programming the gates ourselves onto the MXDB using BDF diagrams in Intel Quartus. Both methods gave the same final product, as they came from the same fundamental logical expressions, and as such their output was the same for equivalent inputs.

Pre-lab Reflection

Firstly, this pre-lab solidified my conceptual understanding of logical expressions as well as their associated theorems – especially DeMorgan's law. But beyond refreshing my understanding, it also gave me an introduction to Karnaugh maps, which were significantly helpful in arriving at the minimized form of our two boolean equations. I had an especially hard time converting these expressions into NAND-only and NOR-only circuits, but I was eventually able to overcome this diffuculty through experimentation and through studying the ways in which other gates can be constructed with NAND and NOR gates. Towards the end of the pre-lab, my practical skills were put to use through the programming of the previously mentioned NAND and NOR circuits onto the MXDB using Intel Quartus. I appreciated this part of the pre-lab because it emphasized that the same outcome can be achieved in multiple ways, some easier than others.

Post-Lab

Problem Statement

This lab consisted of implementing the aforementioned equations Y1 and Y2 in a VHDL program. That is to say, we aimed to write code in the VHDL language, which we then programmed onto the MXDB to achieve a circuit that took inputs A, B, and C, and depending on their input values, gave an output corresponding to the value that Y1 and Y2 would give in their equations.

Design

The design of this VHDL implementation was straightforward. Since we already have the minimized equations for Y1 and Y2, the design was only a matter of implementing the necessary IEEE libraries, declaring the the entity and the architecture with the correct syntax, and writing down the equations with each variable being represented as a signal (A, B, and C as input signals, Y1 and Y2 as output signals). The code consisted of the following:

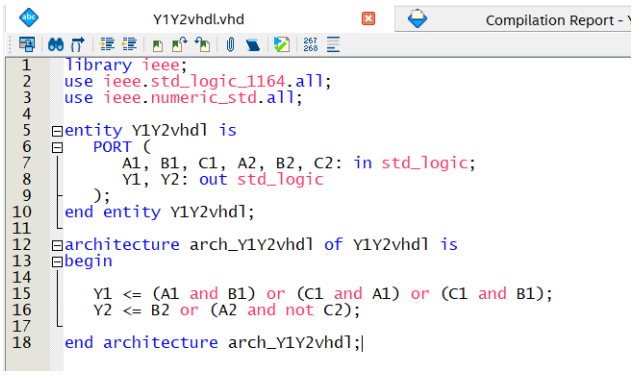


Figure 10: VHDL code for the equations Y1 and Y2

Implementation

In the design of the lab, inputs A,B,C for Y1 and A,B,C for Y2 were distinguished by adding a 1 or a 2 after their letters. An alternative implementation would involve simply using the same three inputs for both Y1 and Y2, which would lead to the same result with less hardware usage (less cables, less switches).

The final part of the design consisted of the pin assignment. For both the pre-lab BDF implementations and this VHDL implementation, the following pin assignments were chosen:

- Pins E3, E4, F4, and G4 for inputs A_1, B_1, C_1, and output Y_1, respectively.
- Pins M12, M11, M10, and M9 for inputs A_2, B_2, C_2, and output Y_2, respectively.

SIP resistors were used to transmit power to a series of switches with jumper cables, which then were fed as inputs to the MXDB pins. The output pins were hooked up to DIP resistors to eventually transmit power to an LED.

Testing

Testing was done, first and foremost, through a waveform simulation, which resulted as follows:

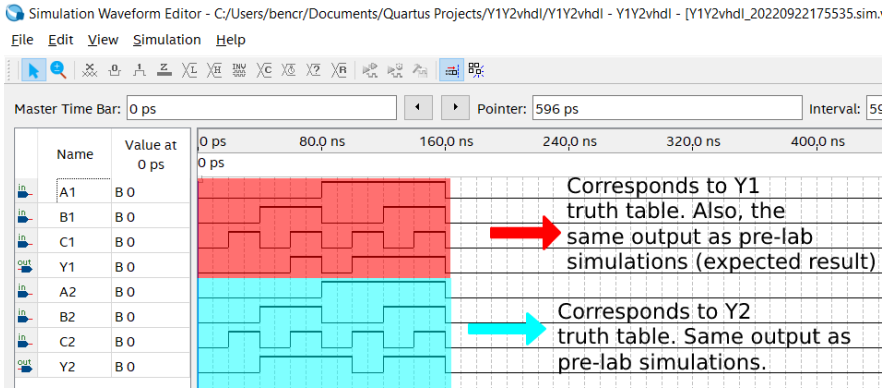


Figure 11: Waveform simulation of the VHDL implementation of Y1 and Y2

As mentioned in the above figure annotations, the output of the simulation not only corresponded to the truth tables for their respective equations as seen in pre-lab section (a), but they are also identical to the previous simulations in parts (e) and (f) of the pre-lab. This makes sense, given that each part is implementing the same two equations – just in different ways.

The second set of testing was done after having programmed the code onto the MXDB. This testing, which was administered by the TA/lab instructor, involved flipping the switches for each input and ensuring that the expected ouput was obtained for a given configuration. In our case, this involved checking whether or not the LED in our breadboard lit up or remained off when it was supposed to.

Conclusion

As a whole, the overall lab procedure was a success. By ironing out both software and hardware problems in the pre-lab, the lab was able to be carried out with little to no confusion or mistakes, and was approached with a set of fundamental knowledge that made the process a lot easier. As for potential improvements, there exists one, which is to reuse inputs A, B, and C for both equations Y1 and Y2, as opposed to declaring A1, B1, C1, A2, B2, and C2, as was done in this lab. This would cut down on time as well as reduce the hardware needs for the lab implementation.

Appendix

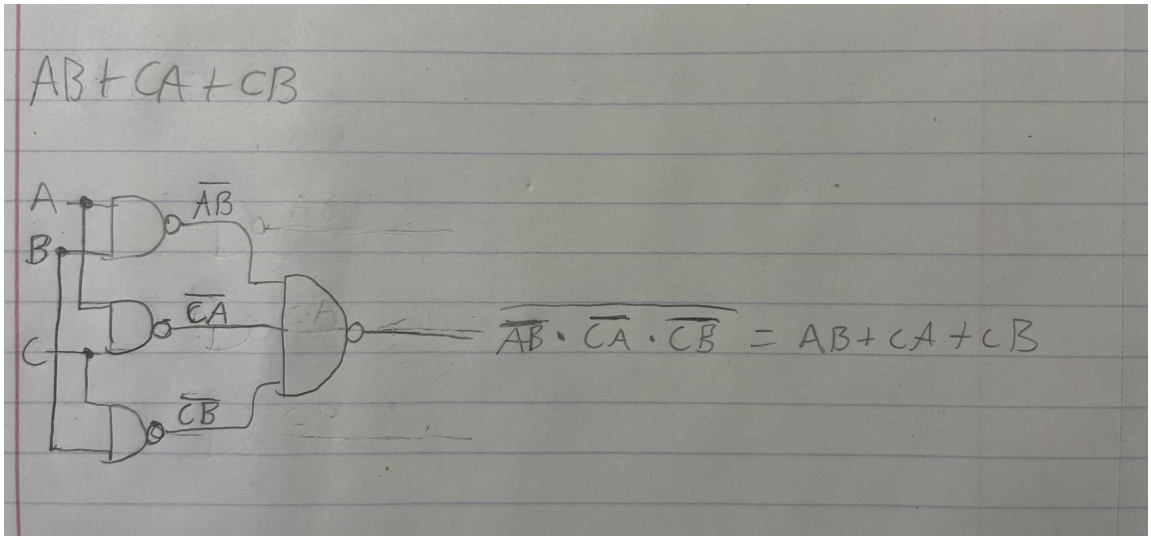


Figure 1: Schematic of minimized Y1

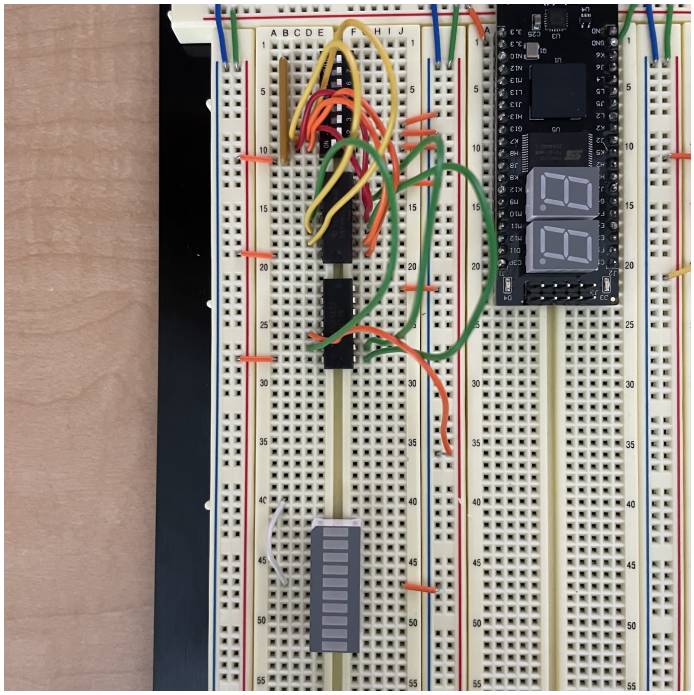


Figure 2: Implementation of equation Y1 with NAND gates (checked off in lab)

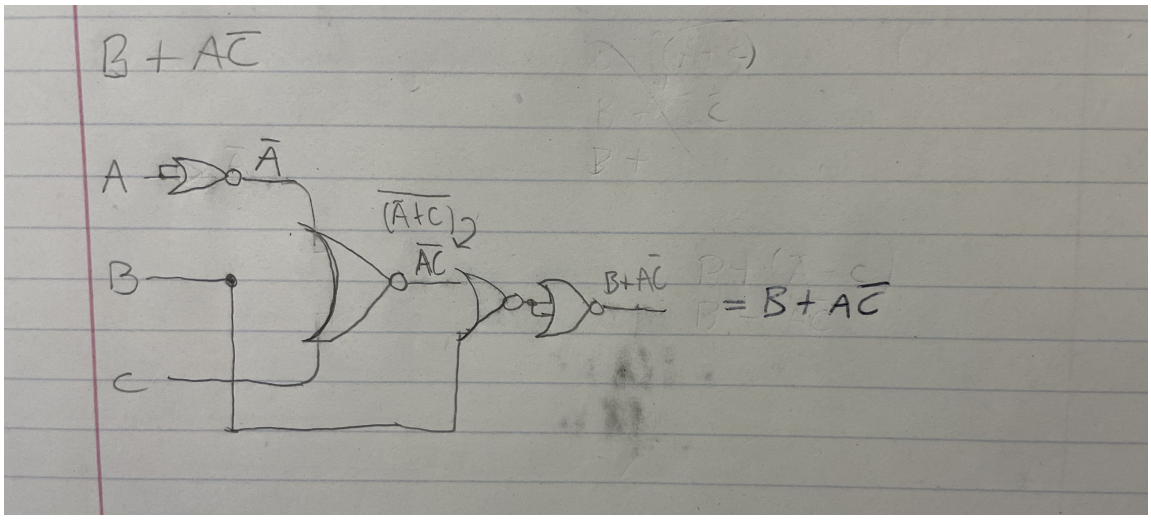


Figure 3: Schematic of minimized Y2

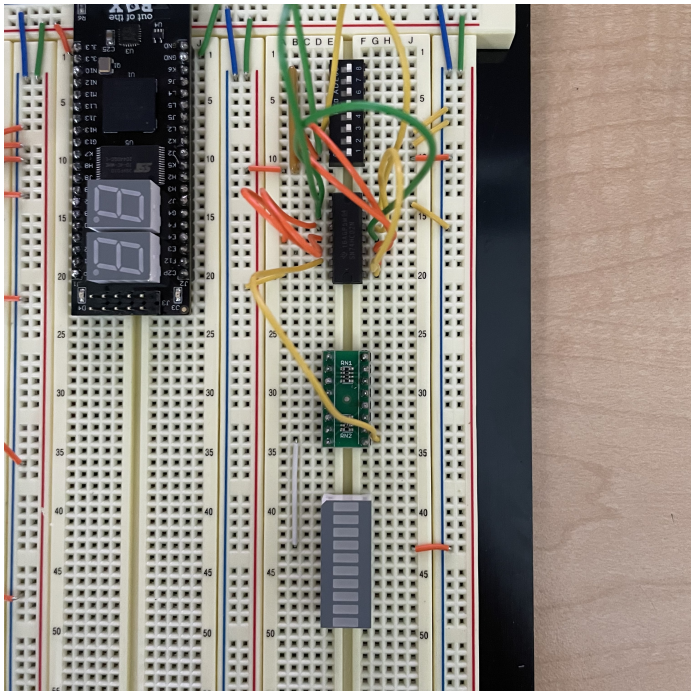


Figure 4: Implementation of equation Y2 with NOR gates (checked off in lab)

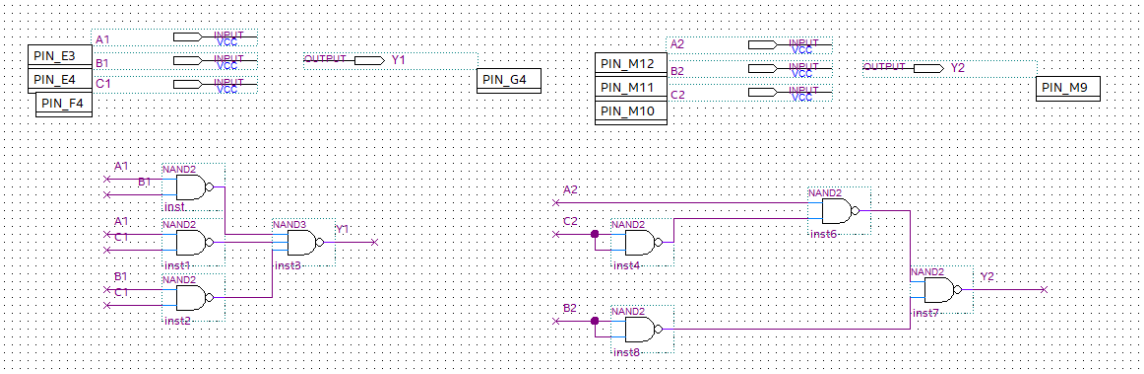


Figure 5: NAND BDF representation of Y1 and Y2

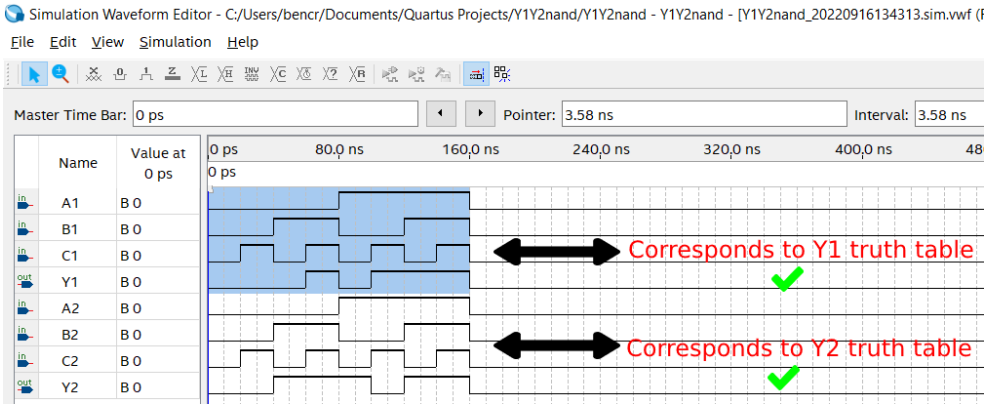


Figure 6: Simulation of the above BDF design (Low represents 0, high represents 1)

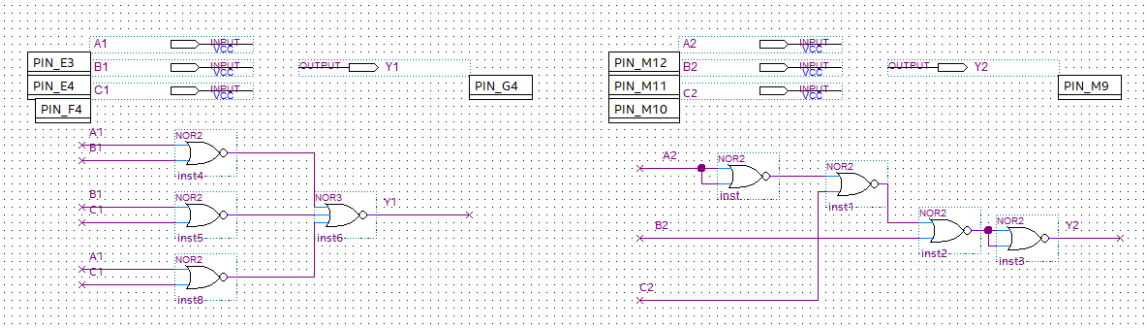


Figure 7: NOR BDF representation of Y1 and Y2

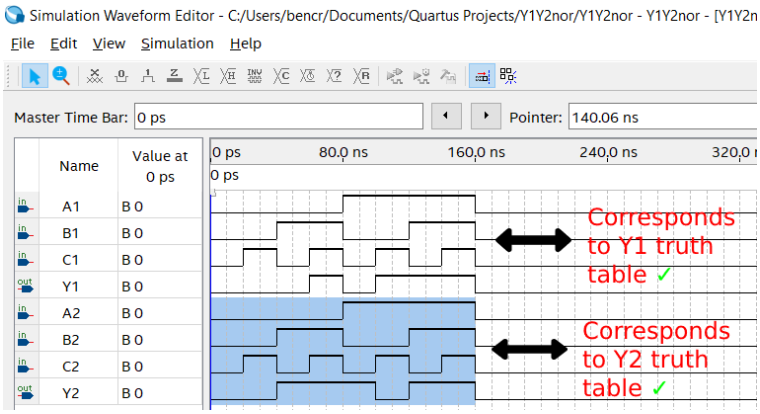


Figure 8: Simulation of the above BDF design (Low represents 0, high represents 1)

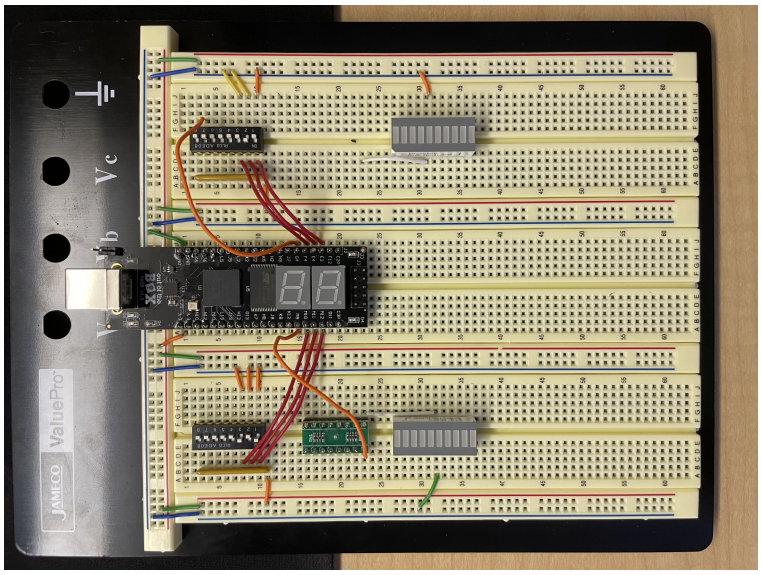


Figure 9: Breadboard after part (f) has been programmed onto the MXDB (checked off in lab)

Y1Y2vhdl.vhd — Compilation Report -

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity Y1Y2vhdl is
   PORT (
      A1, B1, C1, A2, B2, C2: in std_logic;
      Y1, Y2: out std_logic
   );
end entity Y1Y2vhdl;

architecture arch_Y1Y2vhdl of Y1Y2vhdl is
begin

   Y1 <= (A1 and B1) or (C1 and A1) or (C1 and B1);
   Y2 <= B2 or (A2 and not C2);

end architecture arch_Y1Y2vhdl;
```

Figure 10: VHDL code for the equations Y1 and Y2

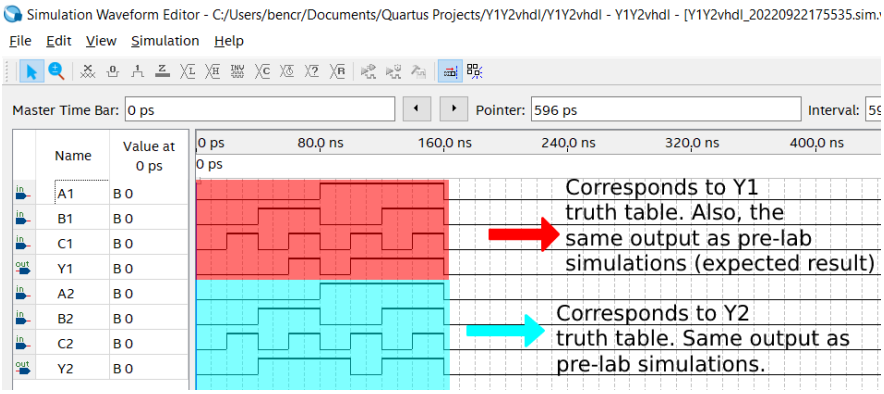


Figure 11: Waveform simulation of the VHDL implementation of Y1 and Y2