To:	Dr. Larry Bland
From:	Jake Caudle and Grant Bruner
Date:	11/21/2016
Re:	Design #2 – Coffee Machine

PART 1: CIRCUIT DESCRIPTION

The purpose of this project was to design a coffee machine that would perform various operations depending on what the user specified to the system. Specifically for this project, the objective was to design a custom coffee vending machine with a minimum amount of gates and the machine was to be able to serve a 12 oz. cup of coffee for 40 cents, with or without sugar and creamer. The machine would include switches indicating the type of coffee wanted, the use of pushbuttons for the input of change, LEDs to represent the type of coffee and time of coffee being poured, as well as the 8 segment Hex lights.

The first part of the design was to design a circuit that would control the coffee machine. The red LEDs would be used to indicate which type of coffee was chosen, and based off of the type of coffee chosen, different lights would illuminate. Switches were used in this circuit to differentiate the choices of coffee.

The second part of the design was to make a controller that accepts nickels, dimes, and quarters as inputs. To do so, pushbuttons were used to represent the coin inputs, Key0 was used for nickels, Key1 was used for dimes, and Key2 was used for quarters. As the change was entered into the circuit, a display of the sum was shown onto the DE2 board on the 8 segment display. When the input exceeded 40 cents, the display would change from showing the sum of the change to the amount of change to be dispersed. The display was shown by writing code in a VHDL file.

Once the correct amount of change has been input into the circuit, the circuit would then flow through a few different functions. The first function that it would flow through is to dispense the cup for 10 seconds. This was indicated by using an LEDG light turned on for 10 seconds. After the cup has been dispensed, the machine would flow into the next function which was to dispense the correct mix of coffee into the cup. This action lasted 5 seconds indicated by up to 3 different LEDGs at the same time. Once the coffee mix was finished dispensing, a new LEDG light would have turned on that would indicate that the process was complete.

The last part of the project was to design the machine to be able to know when the coffee machine was out of coffee, sugar, and/or cream. When any of those situations occurred, the machine would not accept any funds or dispense any coffee mix. Table 1 represents the switches and pin locations of the three different situations. The red LEDs were used to show the type of coffee that you want, while the green LEDs are used to show what product is being dispensed at that time. When the machine is out of a product, all the LEDs will turn on except for the product that is out.

PART 2: BLOCK LEVEL DIAGRAM

Figure 1 below shows the coffee machine circuit at its highest level. Part 3 will go into detail on how each block in Figure 1 was designed and implemented in Quartus II.

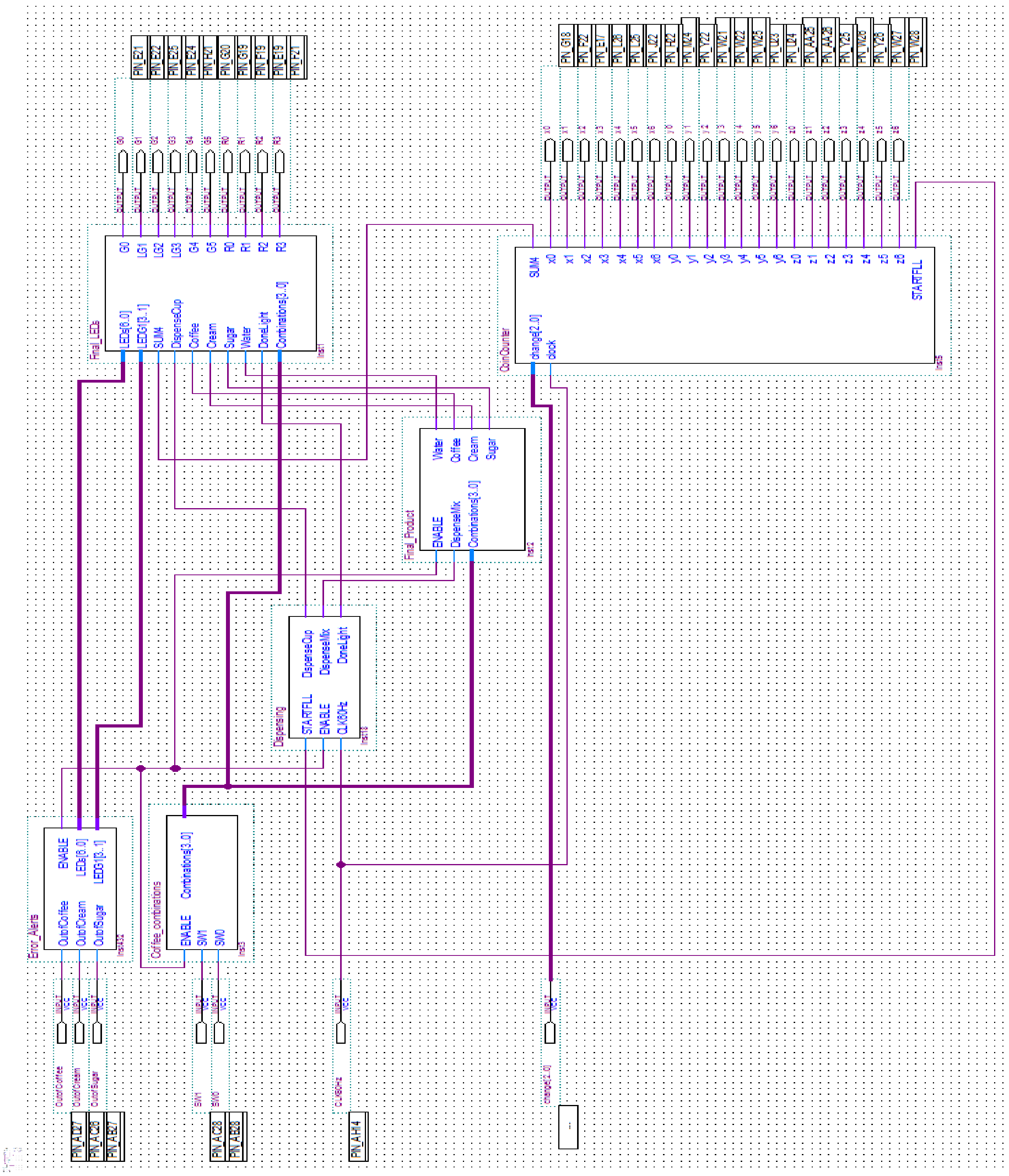


Figure 1: Block Level Diagram Circuit Schematic

The Tables below show the Pin assignments for each of the inputs and outputs for the completed circuit in Figure 1. Table I shows all the outputs while Table II shows each of the inputs.

Table I: Output Pin Assignments of the Coffee Machine

Outputs	Circuit Identity	DE2-115 Assignment	Pin Assignment
	G0	LEDG[0]	PIN_E21
	G1	LEDG[1]	PIN_E22
	G2	LEDG[2]	PIN_E25
	G3	LEDG[3]	PIN_E24
	G4	LEDG[4]	PIN_H21
	G5	LEDG[5]	PIN_G20
	R0	LEDR[0]	PIN_G19
	R1	LEDR[1]	PIN_F19
	R2	LEDR[2]	PIN_E19
	R3	LEDR[3]	PIN_F21
	X0	HEX0[0]	PIN_G18
	X1	HEX0[1]	PIN_F22
	X2	HEX0[2]	PIN_E17
	X3	HEX0[3]	PIN_L26
	X4	HEX0[4]	PIN_L25
	X5	HEX0[5]	PIN_J22
	X6	HEX0[6]	PIN_H22
	Y0	HEX1[0]	PIN_M24
	Y1	HEX1[1]	PIN_Y22
	Y2	HEX1[2]	PIN_W21
	Y3	HEX1[3]	PIN_W22
	Y4	HEX1[4]	PIN_W25
	Y5	HEX1[5]	PIN_U23
	Y6	HEX1[6]	PIN_U24
	Z0	HEX2[0]	PIN_AA25
	Z1	HEX2[1]	PIN_AA26
	Z2	HEX2[2]	PIN_Y25
	Z3	HEX2[3]	PIN_W26
	Z4	HEX2[4]	PIN_Y26
	Z5	HEX2[5]	PIN_W27
	Z6	HEX2[6]	PIN_W28

Table II: Input Pin Assignments of the Coffee Machine

Inputs	Circuit Identity	DE2-115 Assignment	Pin Assignment
	Change[2]	KEY[2]	PIN_N21
	Change[1]	KEY[1]	PIN_MN1
	Change[0]	KEY[0]	PIN_M23
	Clock	CLOCK2_50	PIN_AH14
	OutofCoffee	SW[3]	PIN_AD27
	OutofCream	SW[5]	PIN_AC26
	OutofSugar	SW[4]	PIN_ AB27
	SW0	SW[0]	PIN_AB28
	SW1	SW[1]	PIN_AC28

PART 3: CIRCUIT DESIGN

3.1 COFFEE COMBINATION CIRCUIT

There were several steps that were taken in the process of designing the circuit that determined the combinations of the coffee mixtures that were to be dispensed. This circuit had to be designed in such a way that the inputs Switch 0 and 1 and the ENABLE output from the Out of Product circuit would lead to the four desired outputs coffee combinations of Black, White, Black with Sugar, and White with Sugar. If the ENABLE output from the Out of Product circuit was low it would disable the Switch circuit. The truth table, Table III, shows the exact combinations of the two assigned input switches that would result in the desired coffee. The Enable added very little complexity to the circuit only allowing an output when it was high. A high ENABLE input meant the coffee machine was full of product and therefore the switches could be activated to dispense coffee. The equations for the outputs were easily found through simple logic and did not require any truth tables or Karnaugh maps. The equations for the circuit can be seen below. Each input combination corresponded to a LED output.

Table III: Coffee Combinations

SW0	SW1	Mix	LED
1	1	Black	LEDR0
1	0	White	LEDR1
0	1	Black w/ sugar	LEDR2
0	0	White w/ sugar	LEDR3

$$\text{Black} = \text{ENABLE} * \text{SW1} * \text{SW0} \quad (1)$$
$$\text{White} = \text{ENABLE} * \text{SW1} * \overline{\text{SW0}} \quad (2)$$
$$\text{Black w/ Sugar} = \text{ENABLE} * \overline{\text{SW1}} * \text{SW0} \quad (3)$$
$$\text{White w/ Sugar} = \text{ENABLE} * \overline{\text{SW1}} * \overline{\text{SW0}} \quad (4)$$

Once the equations had been found Quartus II was used to design and then simulate the outcomes. The circuit schematic for the coffee combinations can be seen below in Figure 2. Figure 3 shows the low-level block that was created for the circuit schematic to implement into the higher-level block diagram of Figure 1. A simulation of the circuit was done for a white coffee combination to ensure the correct output. The simulation can be seen below in Figure 4. This simulation shows how the circuit was correct and could be trusted in the final circuit. When tested on the DE2 board the circuit proved to be true as seen from Figures 5 to 6. The

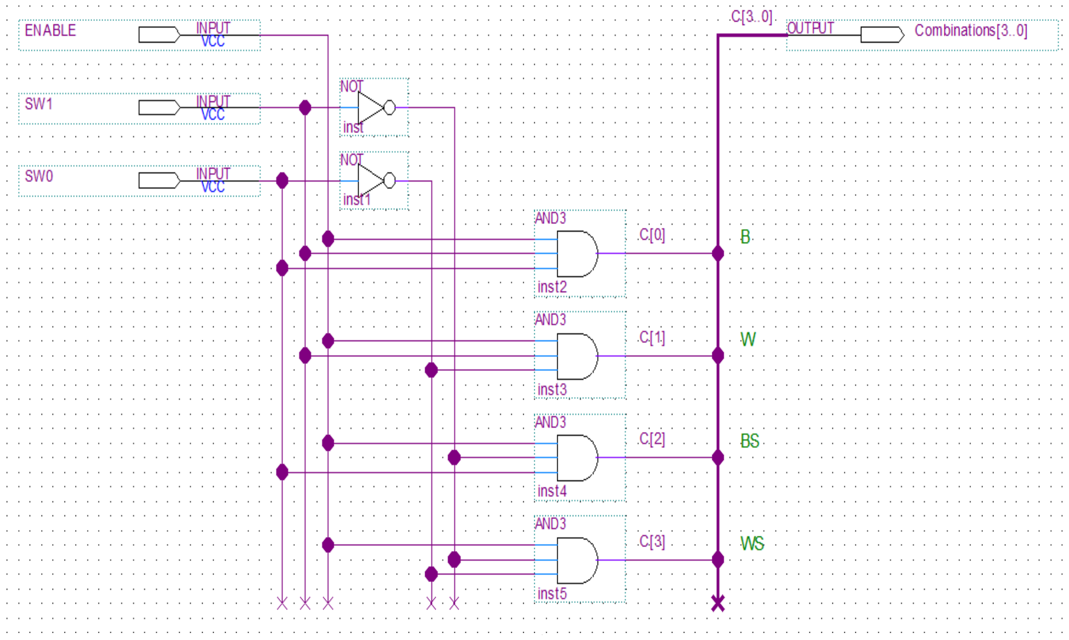


Figure 2: Coffee Combination Circuit Schematic

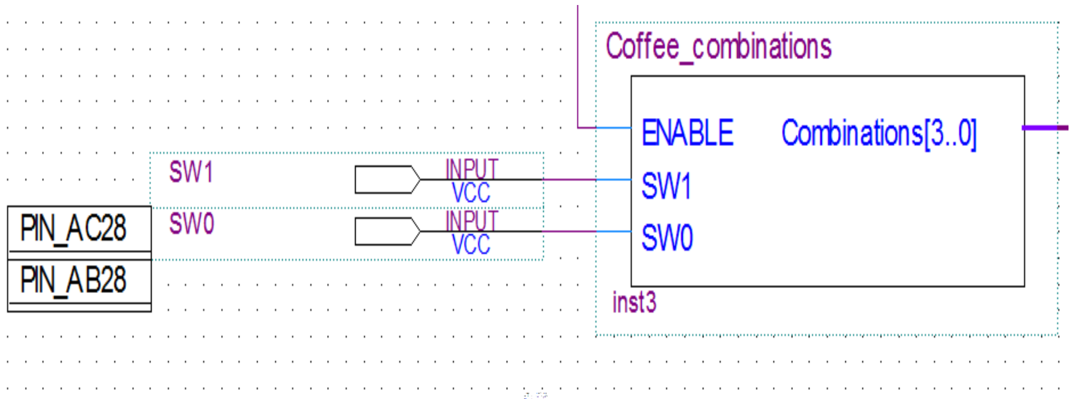


Figure 3: Coffee Combination Block Diagram

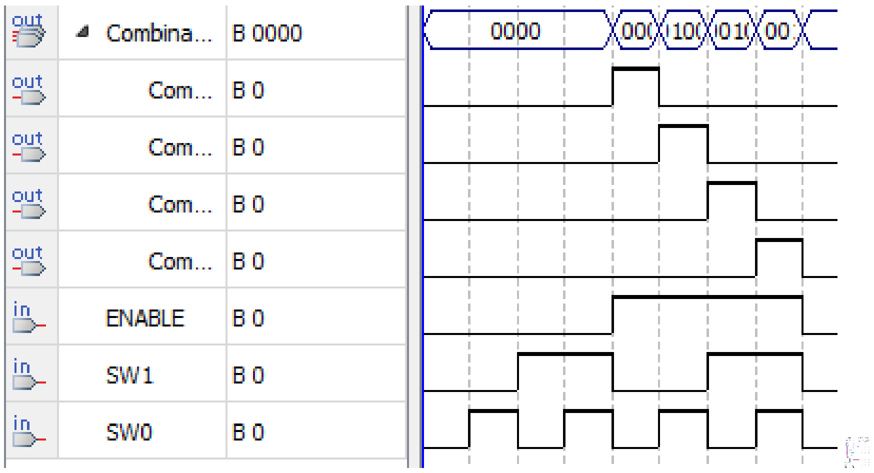


Figure 4: Coffee Combination Simulation for White Coffee

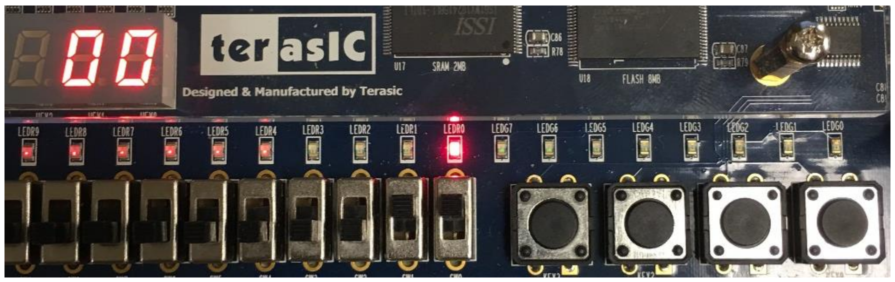


Figure 5: Black Coffee Mix

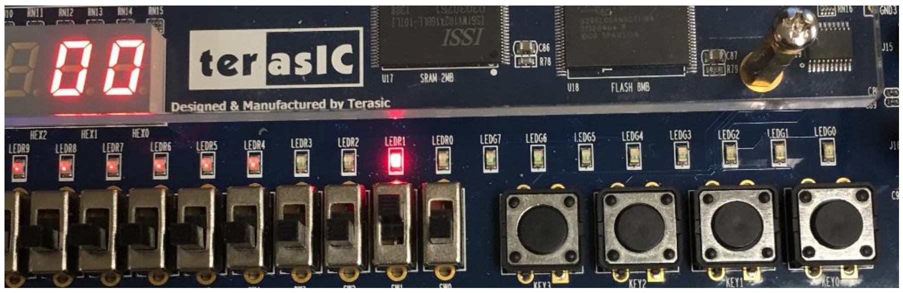


Figure 6: White Coffee Mix

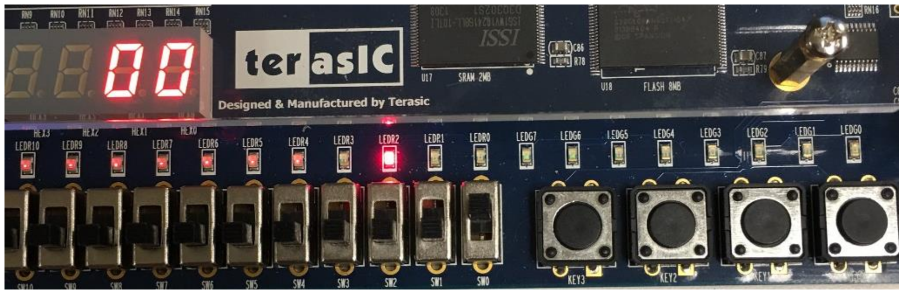


Figure 6: Black with Sugar Coffee Mix

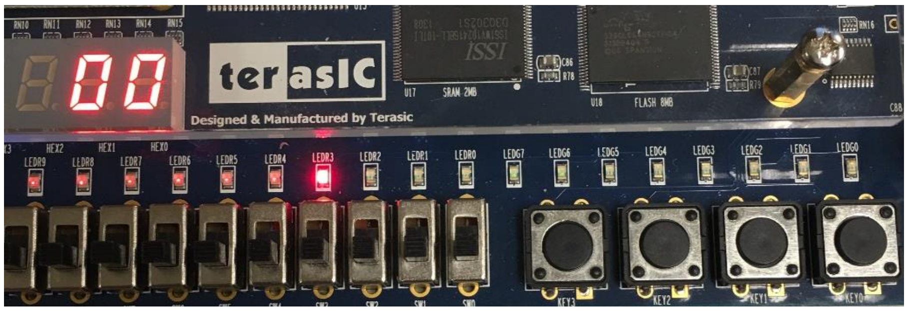


Figure 7: White with Sugar Coffee Mix

3.2 Out of Product

The Out of Product circuit had two purposes, the first was to enable the circuit to run when the coffee machine was full of product, and the second was to alert the user if the machine was out of coffee, cream, or sugar. The circuit contained 3 inputs, OutofCoffee, OutofCream, and OutofSugar. If any of these inputs went high, signaling that the coffee machine was out of a product, the ENABLE output, as seen in the schematic for the circuit in Figure 8, would output low and turn off the rest of the circuits seen in Figure 1. At the same time, this circuit would turn on all the product LEDs (LEDG0-LEDG5) and combination LEDs (LEDR0-LEDR3) except for the LED corresponding to the product that was out. Table IV below shows the combinations for which LEDs would turn on and off for an out of product situation. It was through this table that the equations for the circuit were derived through simple logic, as seen in Figure 8. Figure 9 shows the low-level block diagram that was later used in the final block diagram in Figure 1. As it can be seen from the Table IV, the only LEDs that turn off when out of product are LEDG1, 2, and 3. The equations for these outputs were the same and the equations for the rest of the output LEDs were the same as depicted in Figure 8.

Table IV: Product Alerts

	LEDR0	LEDR1	LEDR2	LEDR3	LEDG0	LEDG1	LEDG2	LEDG3	LEDG4	LEDG5
Out of Coffee	on	on	on	on	on	off	on	on	on	on
Out of Cream	on	on	on	on	on	on	off	on	on	on
Out of Sugar	on	on	on	on	on	on	on	off	on	on

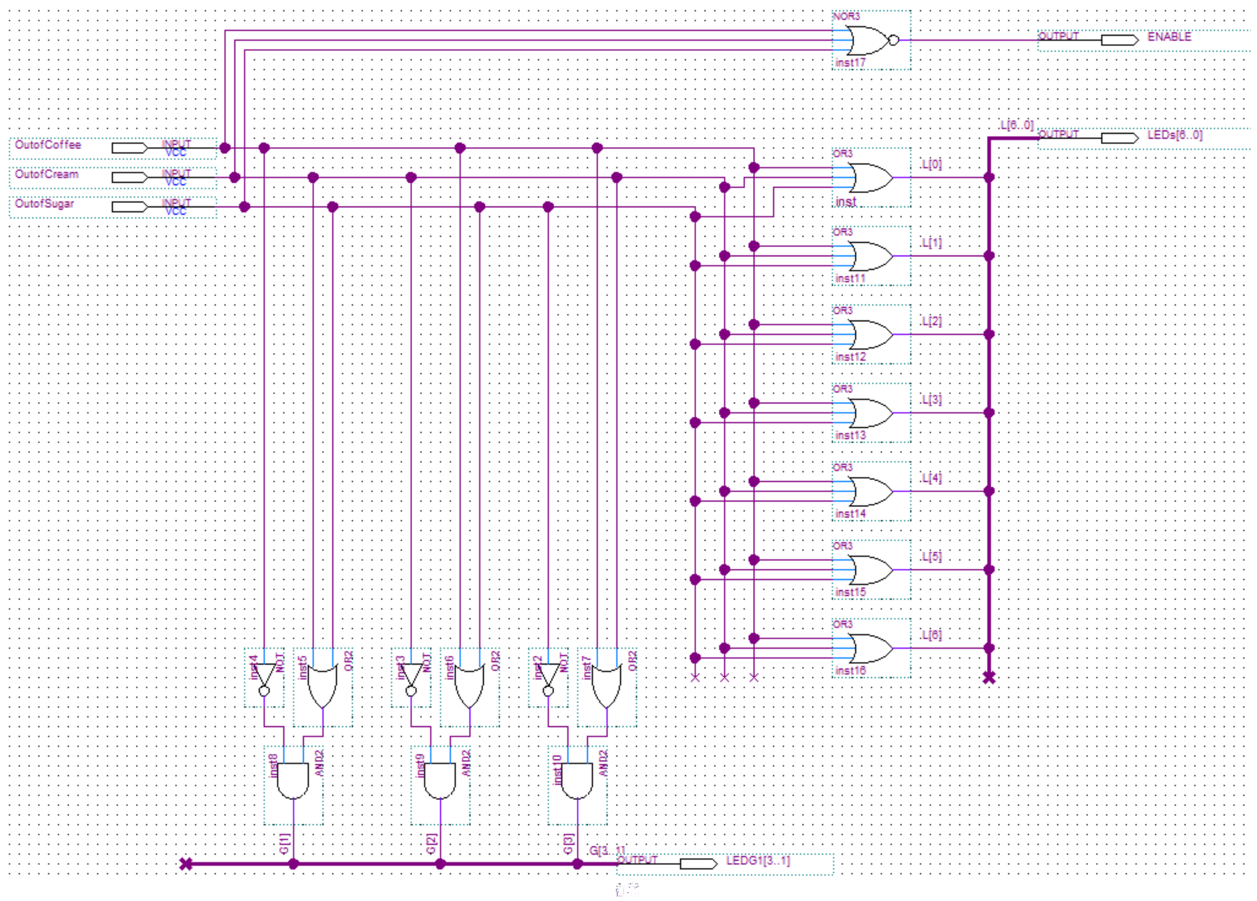


Figure 8: Error Alerts Circuit Schematic

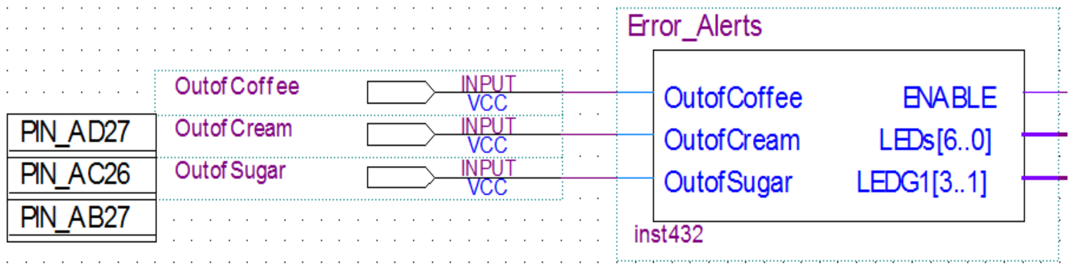


Figure 9: Error Alerts Block Diagram

This circuit was also tested through simulations in Quartus to make sure they were correct before implementation in the final circuit. The simulation for an out of product alert for coffee can be see below in Figure 10. The circuit was also test on the DE2 as can be seen from Figures 11 through 13.

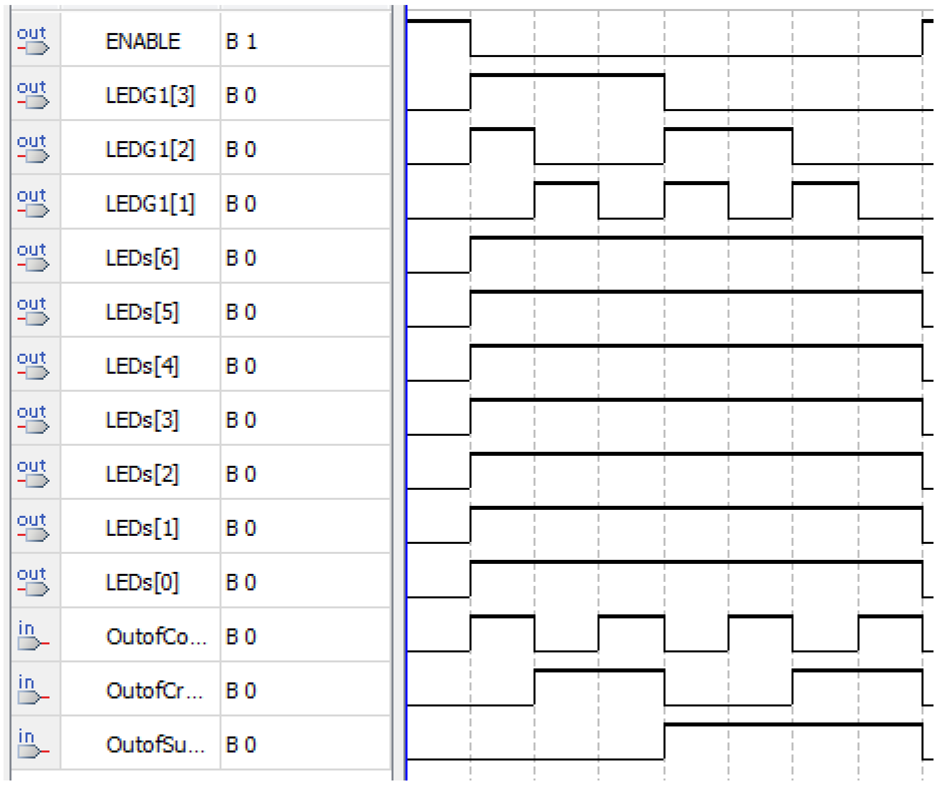


Figure 10: Error Alert for Coffee Simulation

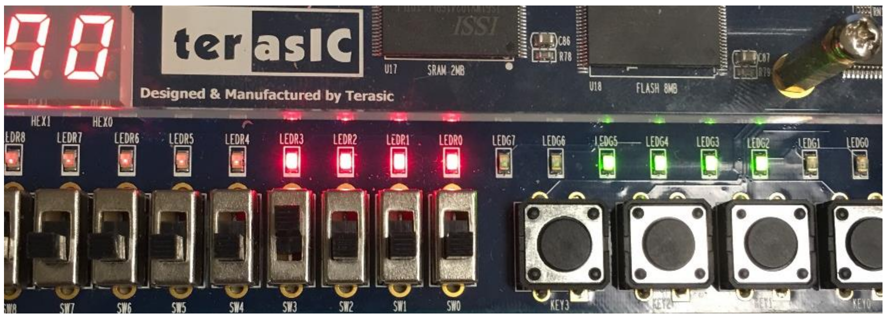


Figure 11: Out of Coffee

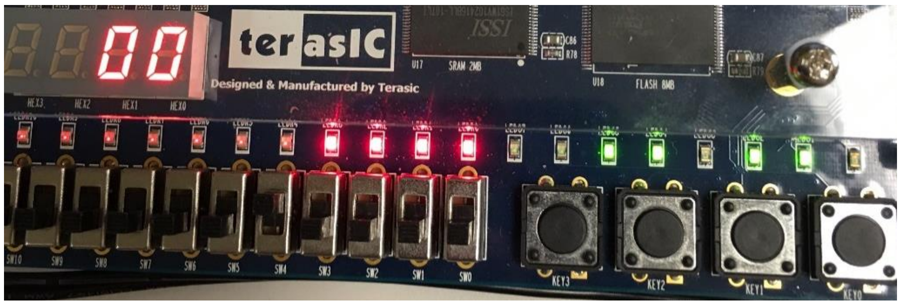


Figure 12: Out of Sugar

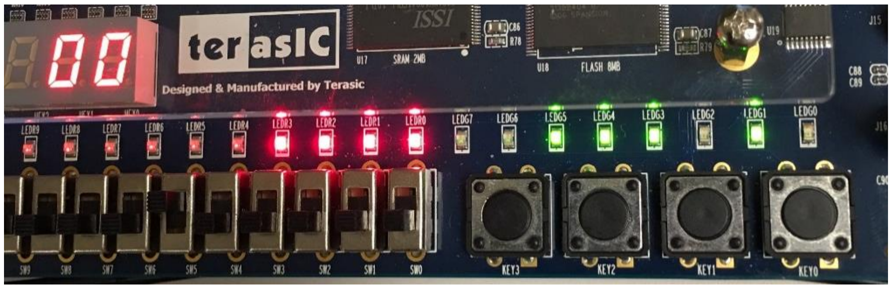


Figure 13: Out of Cream

3.3 Change Counter Circuit

Below in Figure 14 is the schematic for the change counter. The circuit consists of the pushbutton inputs, the nickels, dimes and quarters, as the 74283 4 bit adder, the storage register, a clock and VHDL file for the display. Instead of using 5, 10, and 25 representing the nickel, dime and quarter, the input was divided by 5 to make the circuit simpler, therefore inputs represented 1, 2, and 5 respectively, represented by Table V. Since the pushbuttons are active low, the inputs were each led into a not gate before being fed into the 4 bit adder. The other input in the 4 bit adder was the storage register, which was originally at 0 when the circuit starts. The sum of the 4 bit adder was fed into the 4 inputs of the register. The 4 outputs of the register were fed back into the adder so that the original input would be stored into the register and the output would go back into the adder to be ready for the next input. The output of the register was also connected to an output bus node which was fed into the 4 input VHDL file for display on the DE2 board. The clock in this circuit was fed into an AND gate with an inverted quarter input which was then fed into the register clock. This meant that when the output would become 8 (1000) or greater, the quarter input would become an inverted HIGH which would cause the clock to stop taking inputs.

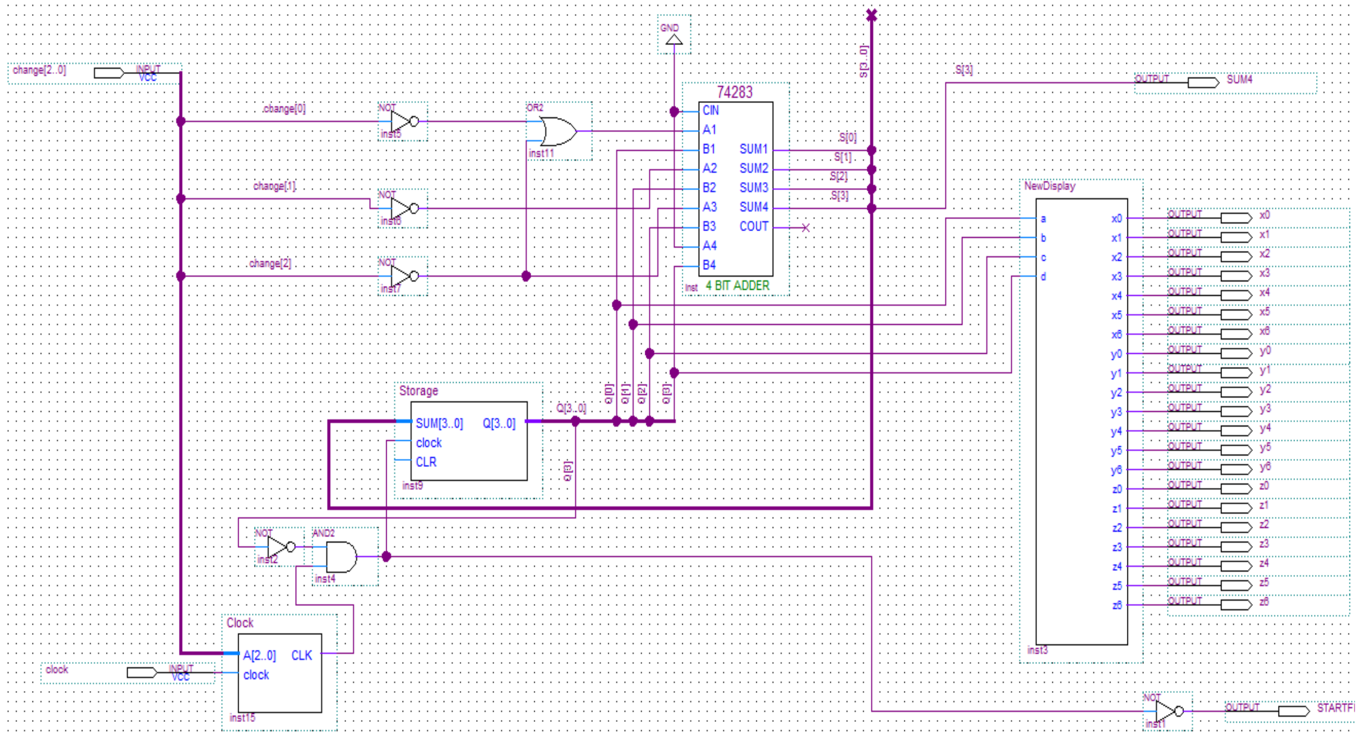


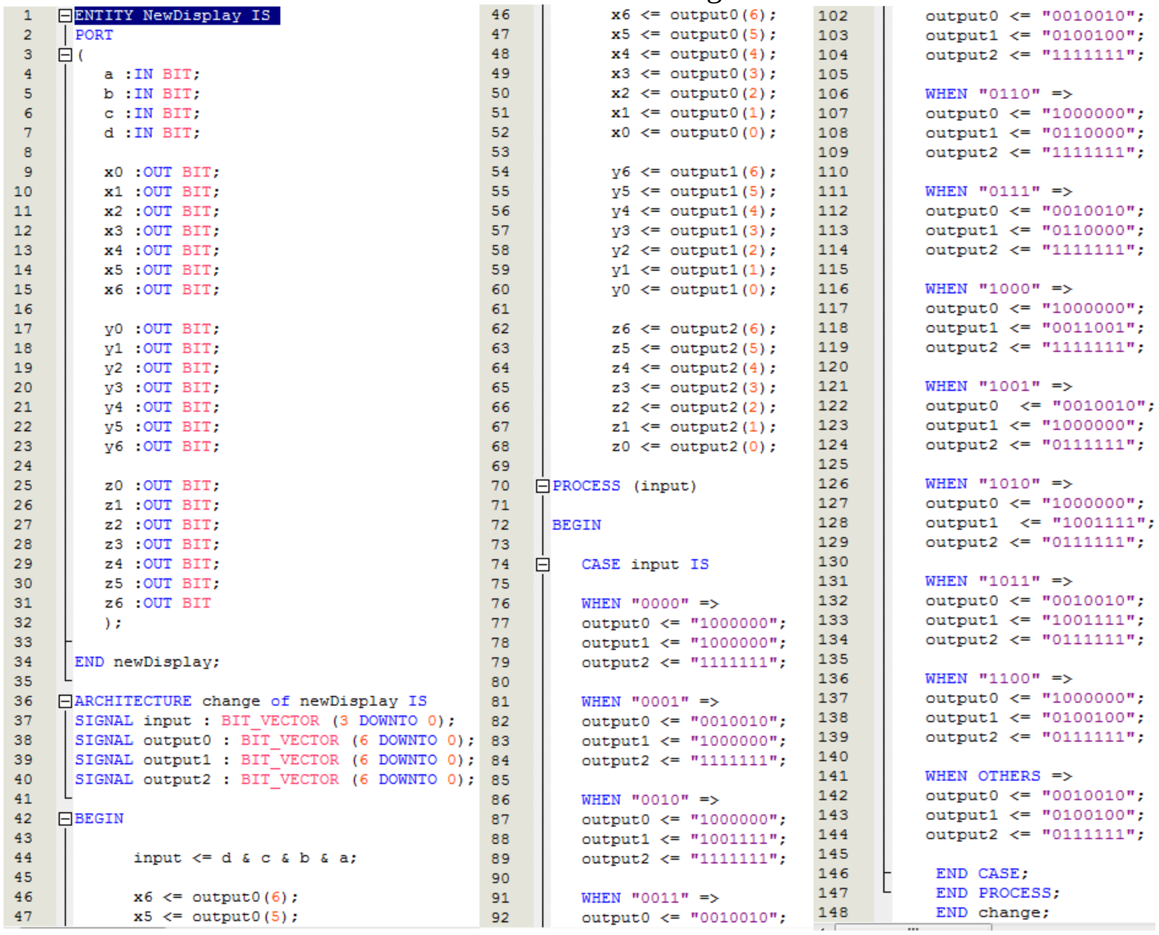
Figure 14: Coin Counter Circuit Schematic

Table V: Input binary numbers

Inputs	**Binary Input**
Nickel	0001
Dime	0010
Quarter	0101

3.3.A VHDL CODE

For the 8 segment LED display on the DE2 board, a VHDL file was created and used for the display of the amount of change or amount of change to be dispersed. The file was named NewDisplay and took 4 inputs and produced a total of 21 outputs. Each output represented a certain LED light on the Hex segment display. The purpose of the code was to produce the intended output based on the current input, which for this circuit meant that it would multiply the input by 5 to reach the correct output. If a “0001” was received in the register (the nickel button was pushed), then the circuit would display the number 5, meaning that the user has inserted 5 cents. The VHDL code can be seen below in Figure 15.



```vhdl
ENTITY NewDisplay IS
PORT
(
    a :IN BIT;
    b :IN BIT;
    c :IN BIT;
    d :IN BIT;

    x0 :OUT BIT;
    x1 :OUT BIT;
    x2 :OUT BIT;
    x3 :OUT BIT;
    x4 :OUT BIT;
    x5 :OUT BIT;
    x6 :OUT BIT;

    y0 :OUT BIT;
    y1 :OUT BIT;
    y2 :OUT BIT;
    y3 :OUT BIT;
    y4 :OUT BIT;
    y5 :OUT BIT;
    y6 :OUT BIT;

    z0 :OUT BIT;
    z1 :OUT BIT;
    z2 :OUT BIT;
    z3 :OUT BIT;
    z4 :OUT BIT;
    z5 :OUT BIT;
    z6 :OUT BIT
    );

END newDisplay;

ARCHITECTURE change of newDisplay IS
SIGNAL input : BIT_VECTOR (3 DOWNTO 0);
SIGNAL output0 : BIT_VECTOR (6 DOWNTO 0);
SIGNAL output1 : BIT_VECTOR (6 DOWNTO 0);
SIGNAL output2 : BIT_VECTOR (6 DOWNTO 0);

BEGIN

        input <= d & c & b & a;

        x6 <= output0(6);
        x5 <= output0(5);
```

```vhdl
        x6 <= output0(6);
        x5 <= output0(5);
        x4 <= output0(4);
        x3 <= output0(3);
        x2 <= output0(2);
        x1 <= output0(1);
        x0 <= output0(0);

        y6 <= output1(6);
        y5 <= output1(5);
        y4 <= output1(4);
        y3 <= output1(3);
        y2 <= output1(2);
        y1 <= output1(1);
        y0 <= output1(0);

        z6 <= output2(6);
        z5 <= output2(5);
        z4 <= output2(4);
        z3 <= output2(3);
        z2 <= output2(2);
        z1 <= output2(1);
        z0 <= output2(0);

PROCESS (input)

BEGIN

    CASE input IS

    WHEN "0000" =>
    output0 <= "1000000";
    output1 <= "1000000";
    output2 <= "1111111";

    WHEN "0001" =>
    output0 <= "0010010";
    output1 <= "1000000";
    output2 <= "1111111";

    WHEN "0010" =>
    output0 <= "1000000";
    output1 <= "1001111";
    output2 <= "1111111";

    WHEN "0011" =>
    output0 <= "0010010";
```

```vhdl
    output0 <= "0010010";
    output1 <= "0100100";
    output2 <= "1111111";

    WHEN "0110" =>
    output0 <= "1000000";
    output1 <= "0110000";
    output2 <= "1111111";

    WHEN "0111" =>
    output0 <= "0010010";
    output1 <= "0110000";
    output2 <= "1111111";

    WHEN "1000" =>
    output0 <= "1000000";
    output1 <= "0011001";
    output2 <= "1111111";

    WHEN "1001" =>
    output0  <= "0010010";
    output1 <= "1000000";
    output2 <= "0111111";

    WHEN "1010" =>
    output0 <= "1000000";
    output1  <= "1001111";
    output2 <= "0111111";

    WHEN "1011" =>
    output0 <= "0010010";
    output1 <= "1001111";
    output2 <= "0111111";

    WHEN "1100" =>
    output0 <= "1000000";
    output1 <= "0100100";
    output2 <= "0111111";

    WHEN OTHERS =>
    output0 <= "0010010";
    output1 <= "0100100";
    output2 <= "0111111";

     END CASE;
     END PROCESS;
     END change;
```

Figure 15: VHDL code

Table X: Hex representations on DE2 board

0	1000000
1	100111
2	0100100
3	0110000
4	0011001
5	0010010
-	0111111

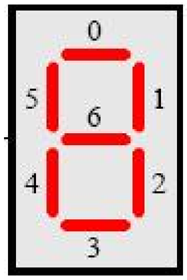


Figure 16: Hex representation on DE2 board

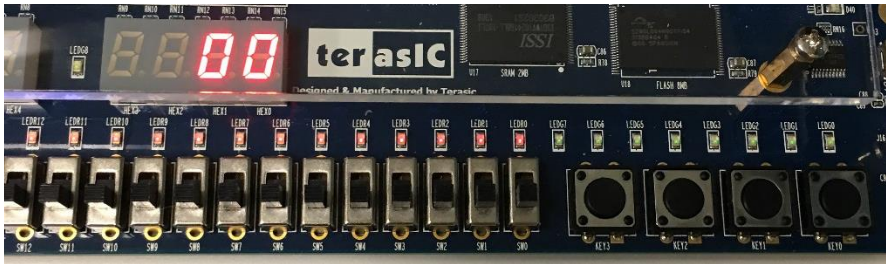


Figure 17: Display starting at 0000

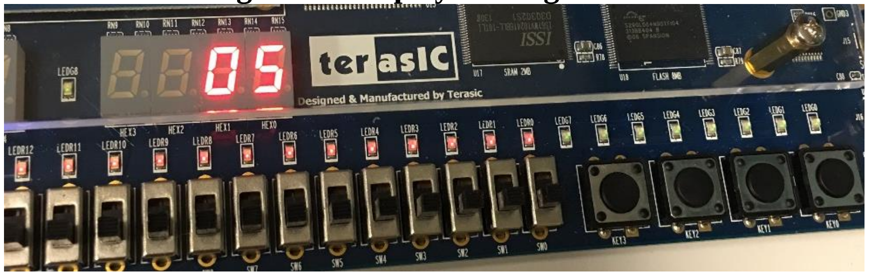


Figure 18: Display when a nickel is pressed

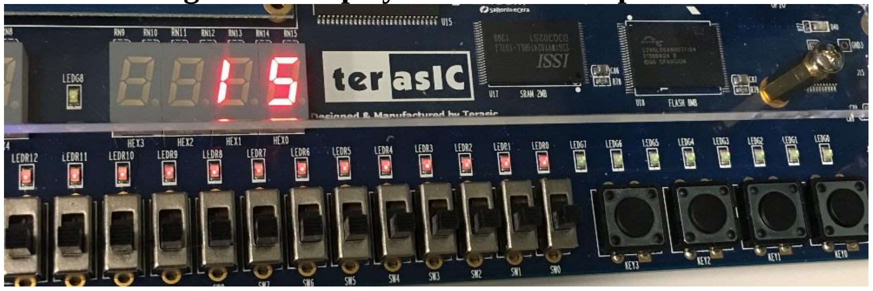


Figure 19: Display when a dime is added (0011)

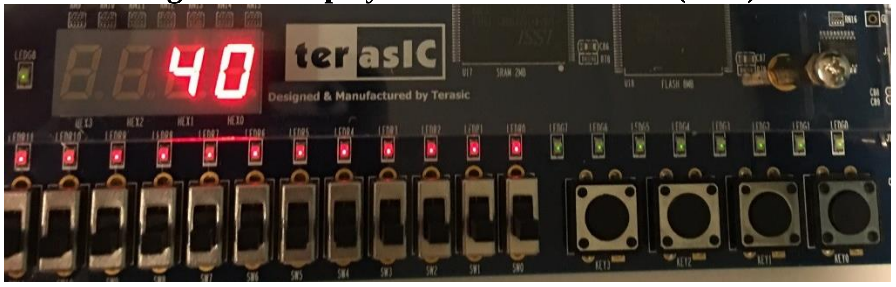


Figure 20: Display when a quarter was added (1000)

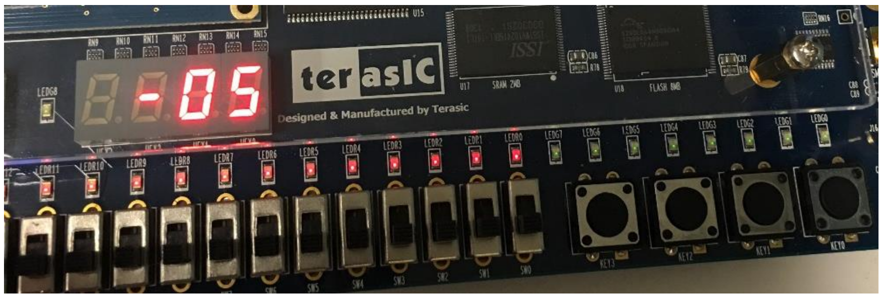


Figure 21: Display when change is needed back (1001)

3.3.B Storage/Register circuit

Below is the Storage/Register that was created by using the Quartus II software. For this specific register, an input of 4 bits – the 4 bits coming from the adder – was created, as well as an output of 4 bits. The output of this register was connected to the 4 bit adder so that it would add the previous input with the next input. The block itself was made up of 4 D-Flip Flops and a synchronous clock connected to each clock input.

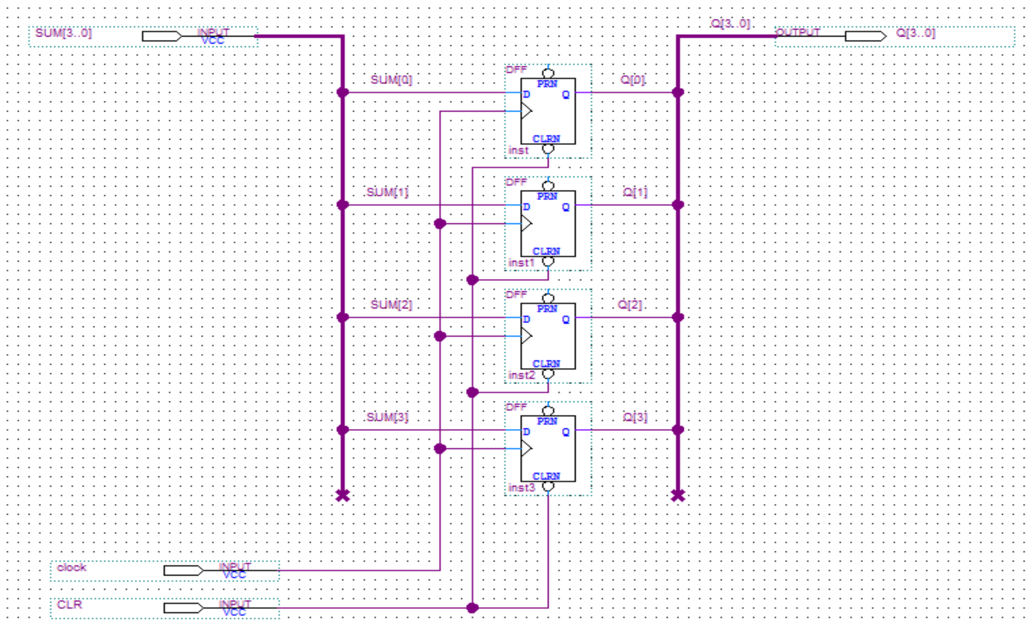


Figure 22: Storage/Register circuit schematic

3.3.C 60Hz Clock

The clock for this part of the circuit was made up of two D-Flip Flops. The first DFF is connected straight to the clock and the other is connected to the NOT gate and then the clock, both are connected synchronously. The inverter is used to capture the negative edge of the clock pulse. The clock has three inputs – the pushbuttons - going into a NAND gate. This meant that whenever a pushbutton was hit, the clock would pulse and capture that pulse to represent that certain input being hit.

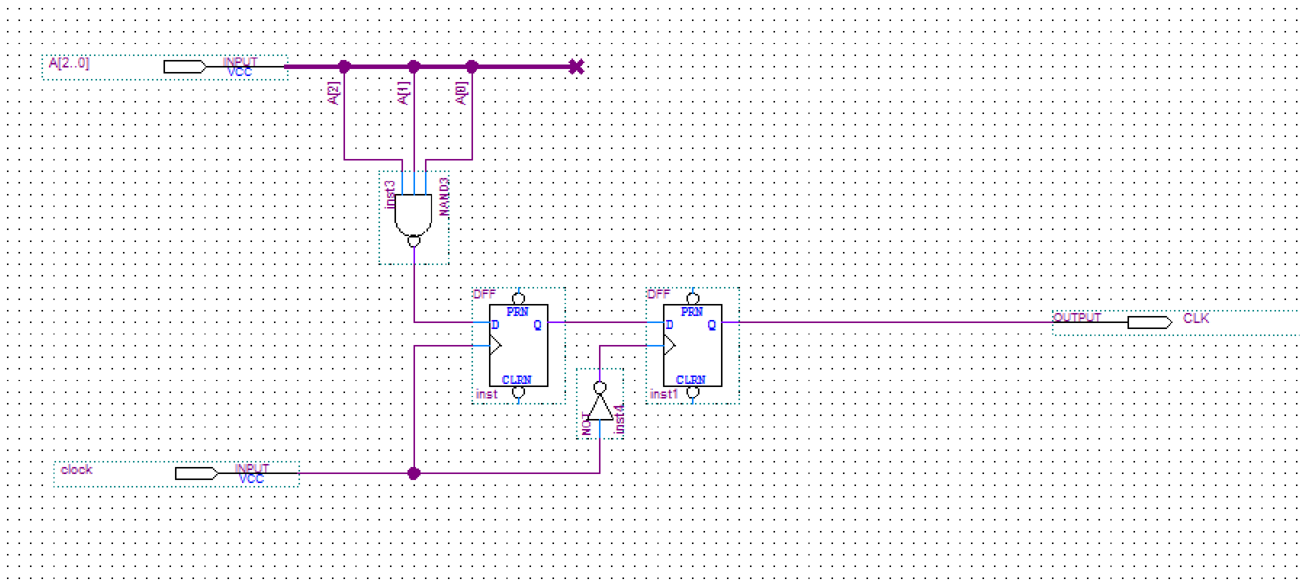


Figure 23: Clock circuit schematic

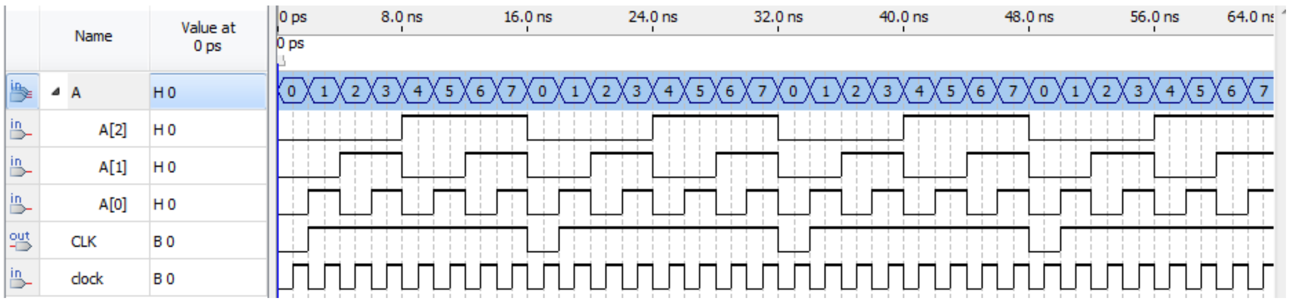


Figure 24: Clock waveform simulation

Figure 24 shows the clock circuit that was constructed in the waveform simulation model on Quartus II. The clock pulses every time the inputs are LOW.

3.4 Dispensing Circuit

This circuit was designed to get the timing for the dispensing cycles of the coffee machine. The design specified that an input clock of 60 Hz would be used to power the machine. The human eye cannot see 60 cycles per second. Therefore, a Mod 60 counter as seen in Figure 25 below was used to reduce the 60 Hz to 1 Hz. The Mod 60 was designed following a diagram found in the Digital Systems book. The 1 Hz outputting from the Mod 60 was immediately inputted into the Mod 16 counter shown in Figure 26. With this sequence of counters and the circuitry coming from the Mod 16 the circuit output DisperseCup took the first 10 states of the counter and the output DispenseMix took the next 5, with the output DoneLight ending the sequence. Once the counter reached 15 or 1111, it would shut off due to the NAND gate which would input back into the Mod 16. The input at the beginning of the Mod 16 was also ANDed to ENABLE which would also shut off the counter if an error occurred with the product. This circuitry can be seen in Figure 26. As seen in Figure 27 the Mod 16 also has the input STARTFILL. This input comes from the counter circuit, and turns the Mod 16 on when the counter has received sufficient payment for a cup of coffee. The simulation for the Dispensing circuit can be seen in Figure 28 below to verify that this circuit would allow the DispenseCup light to be on for 10 seconds followed by 5 seconds of the DispenseMix light and finished by the DoneLight.

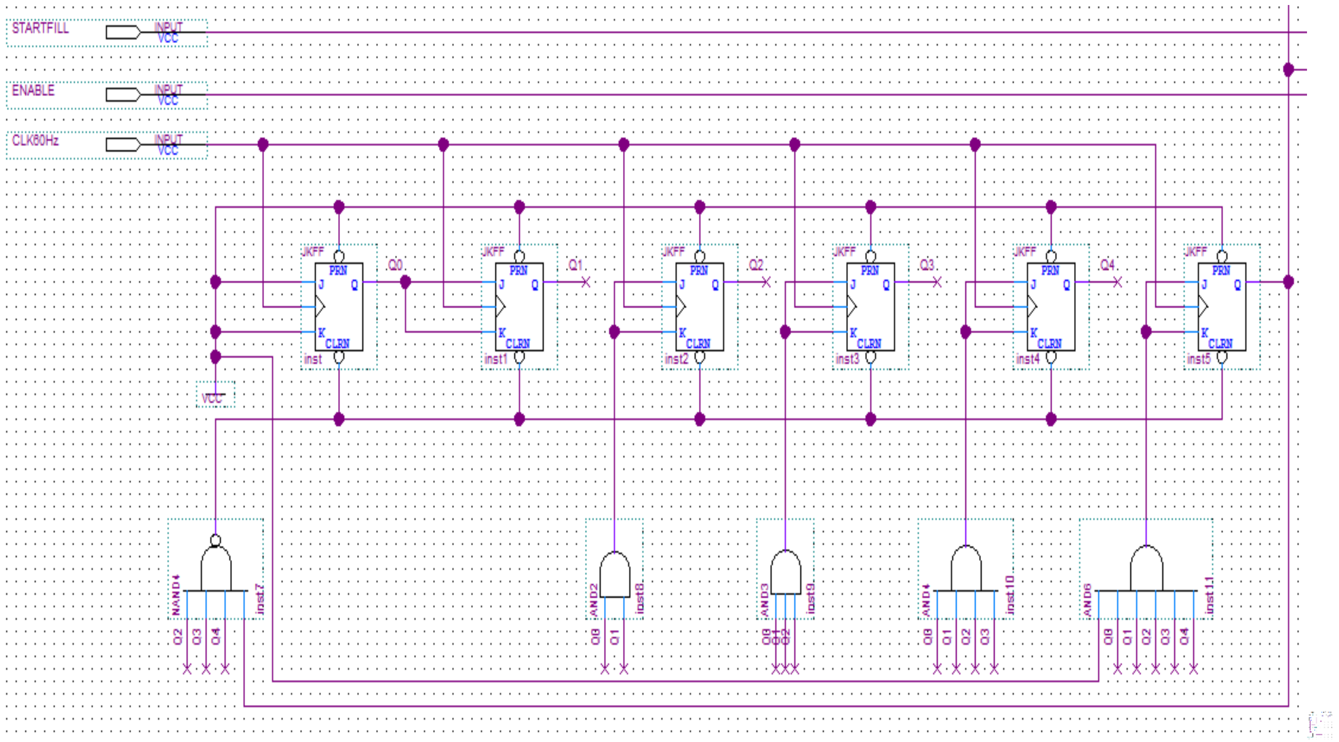


Figure 25: Mod 60 Counter Circuit

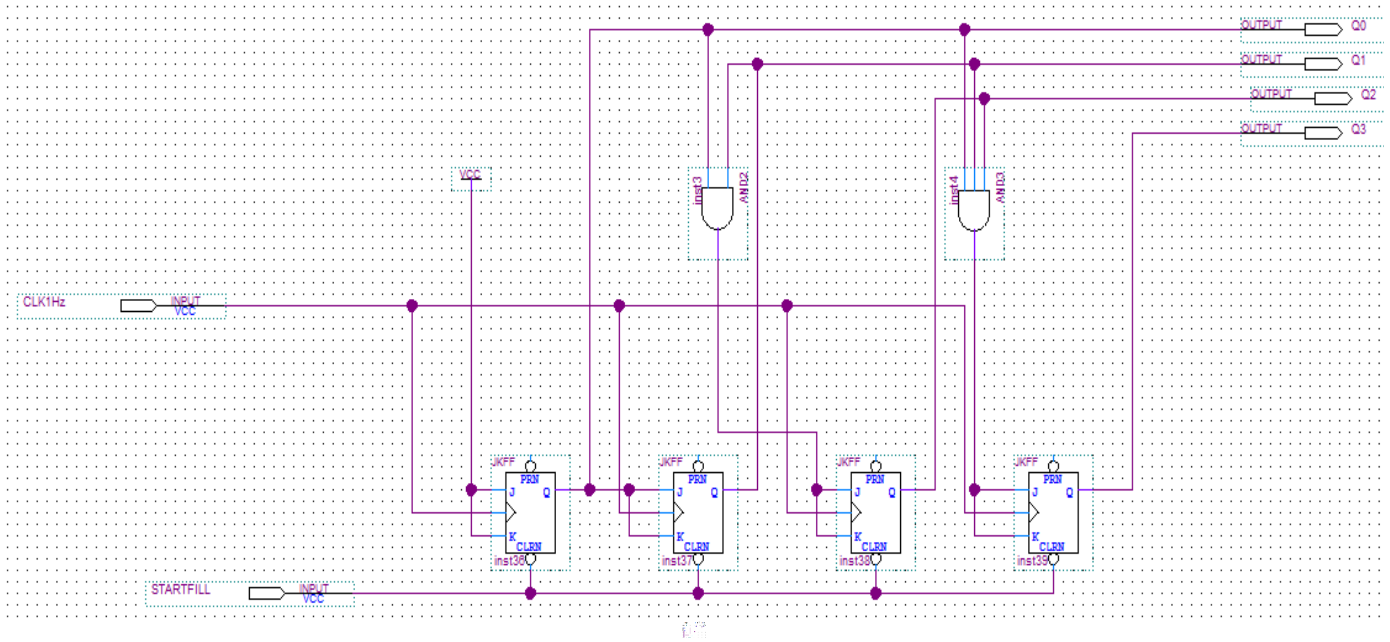


Figure 26: Mod 16 Counter Circuit

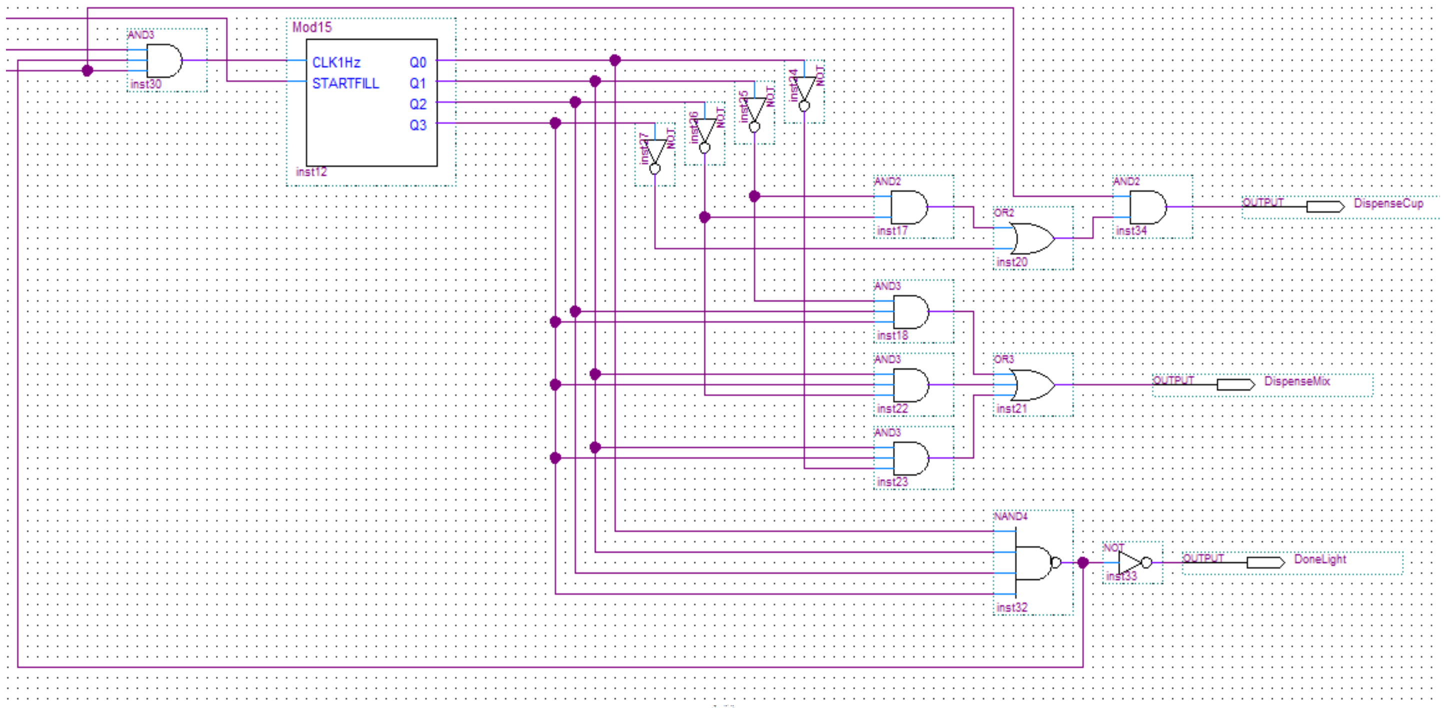


Figure 27: Mod 16 Counter Circuit

3.5 Final Product Circuit

The Final Product circuit was designed to combine the timing of the Dispensing circuit with the switch combinations to determine which LEDs would light up during the DispenseMix cycle. The DispenseMix output was inputted into this circuit along with, ENABLE, and the combination outputs from the Coffee Combinations circuit. The outputs Coffee, Water, Sugar, and Cream would be illuminated during the 5 seconds of the mixture dispensing depending on what combination of coffee was selected at the beginning of the process. The logic process for this circuit was also very simple and did not require any tables or Karnaugh maps. The circuit schematic as well as the block diagram for this circuit can be seen in Figures 28 and 29 below.

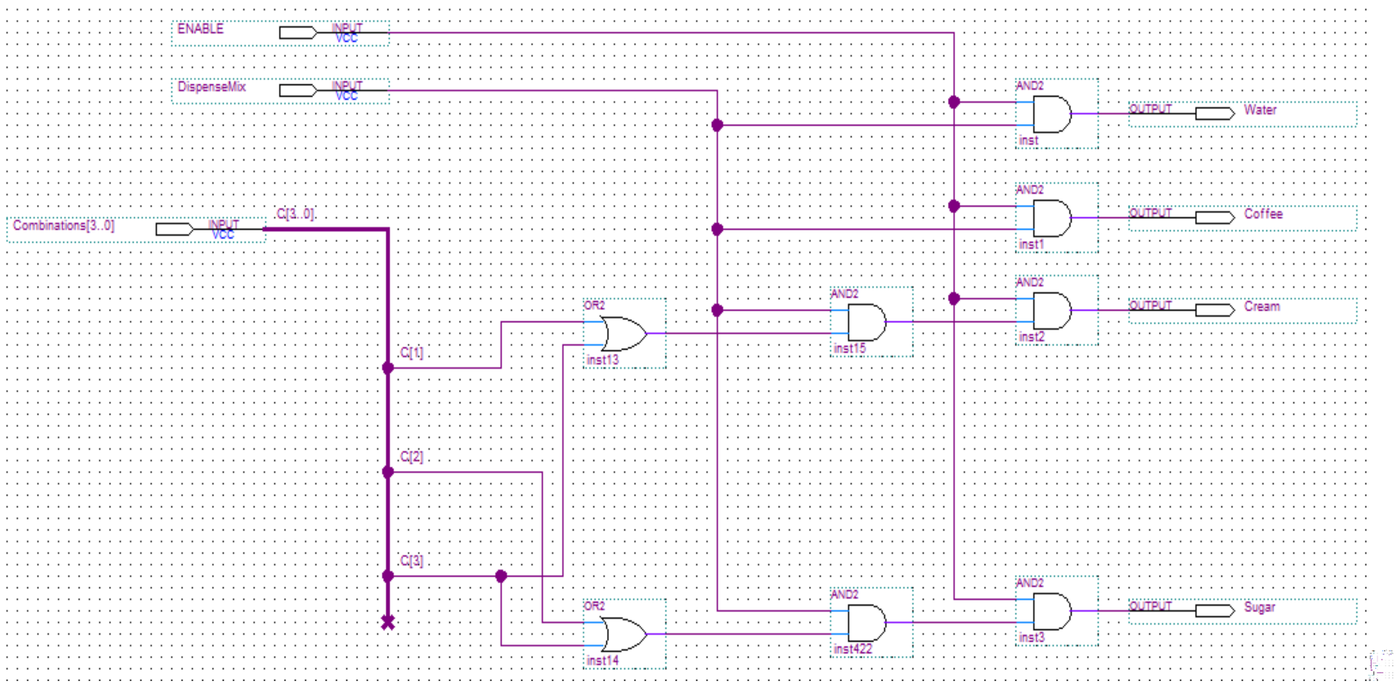


Figure 28: Final Product Circuit Schematic

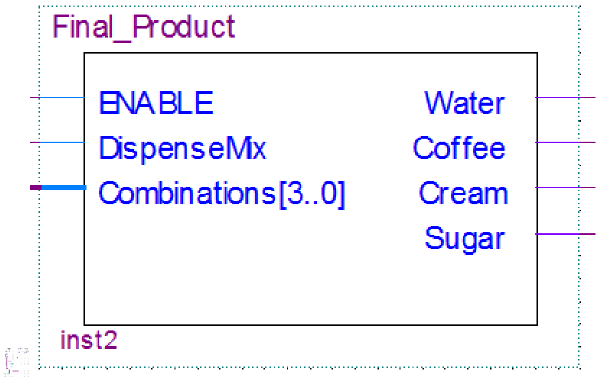


Figure 29: Final Product Circuit Schematic

3.6 Final LEDs Circuit Design

For all the outputs from each of the individual circuit described above to come together and output through only 10 LED lights, a circuit had to be made that would determine whether the coffee machine had product or not and then allow the dispense sequence or active the error alerts. As seen below in Figures 30 and 31 the circuit inputted all the outputs from the Out of Product circuit, Coffee Combinations circuit, Dispensing circuit, and Final Product circuit. The circuit then decide which input to allow out based on the ENABLE that was already built into the inputs. The outputs were based on simple or expressions as seen in Figure 30.

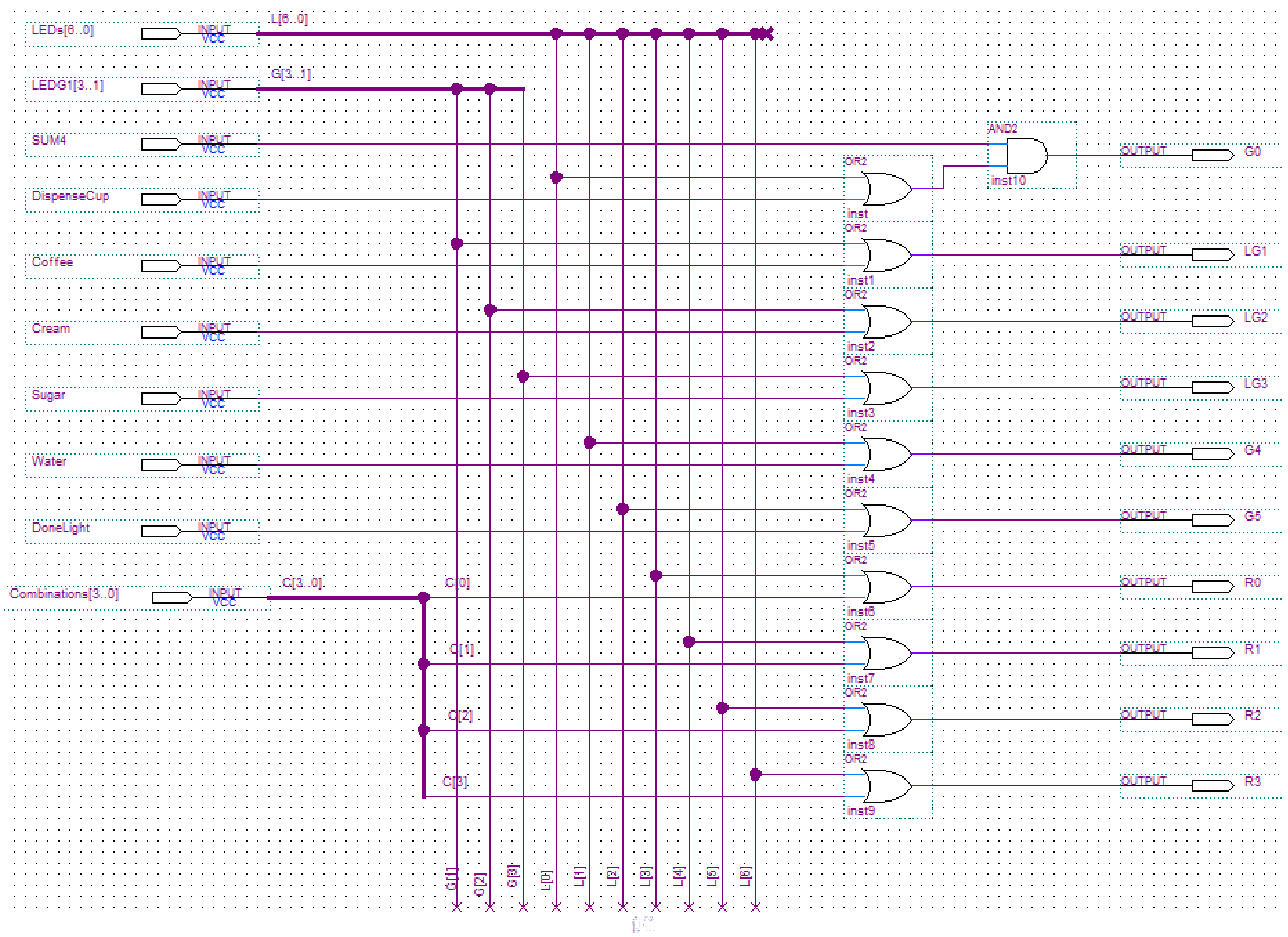


Figure 30: Final LEDs Circuit Schematic

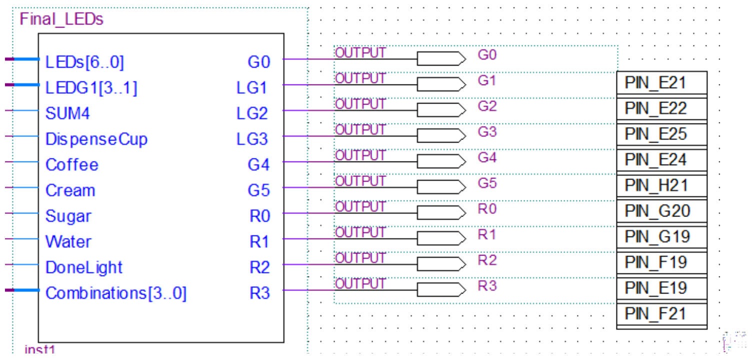


Figure 31: Final LEDs Block Diagram

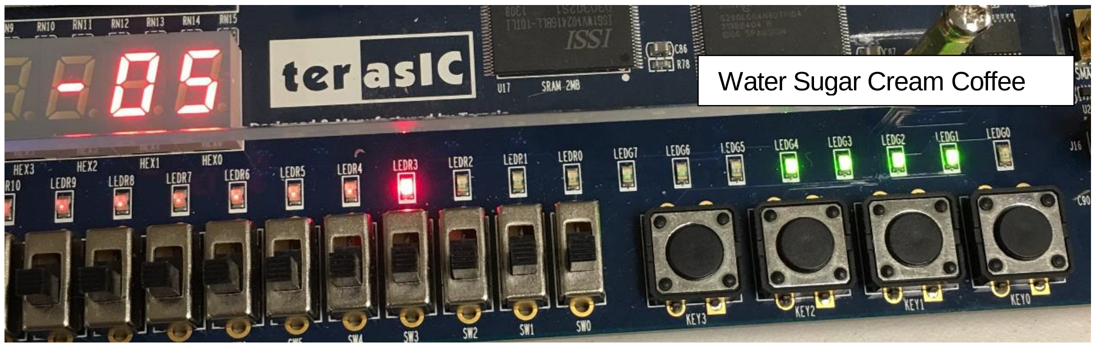


Figure 32: White Coffee with sugar being dispensed

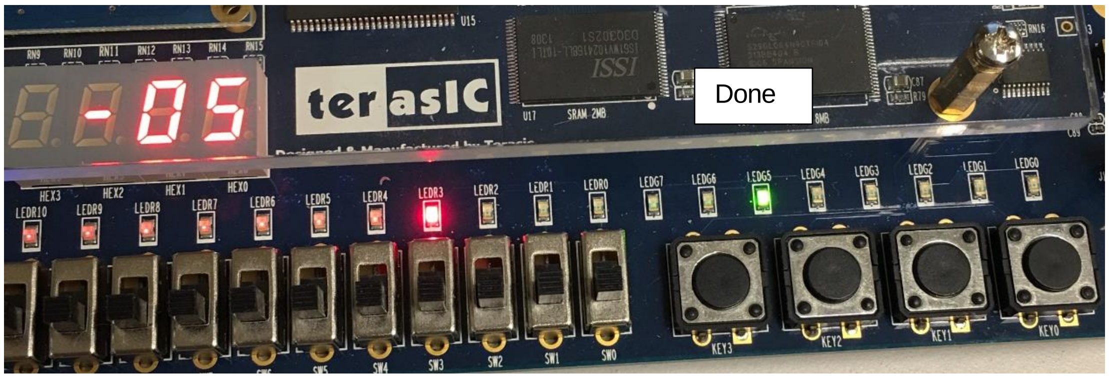


Figure 33: Done LEDG light when circuit is done dispensing coffee mix

PART 4: CONCLUSIONS

We learned quite a few important things that will be used for future labs.

- Learned to start the design as early as possible.
- Learned to work on the project in small parts rather than large scale.
- Learned to run many simulations and test multiple ideas.
- Learned to make sure the output from the waveform generator was on before changing circuitry.
- Learned to make bus wires as they can help make the circuit look cleaner.
- Learned to make hierarchy blocks to make overall schematic cleaner.
- Learned to make truth tables and Karnaugh maps to help solve problems.
- Learned to keep trying even when you do not know the answer.