ASHESI
UNIVERSITY

CS 330 Hardware and Systems Fundamentals

Spring 2024: Final Project

Due Date: Monday, 8th April 2024.

INTRODUCTION

In the realm of digital circuit design and simulation, Logisim Evolution stands as a powerful tool for constructing and analyzing various electronic circuits. This term project delves into the multifaceted functionalities of Logisim Evolution through the development of four distinct modules: Binary to Gray Code Converter, Gray Code to Binary Converter, BCD to 7-segment Decoder, and the Ashesi Vending Machine. Each module represents a crucial aspect of digital logic design, ranging from fundamental conversions to practical applications. Through this project, we aim to explore the intricate workings of these modules while honing our skills in digital circuit design, simulation, and analysis.

Key Objectives:

1. **Binary to Gray Code Converter:**
    - Design a robust binary to Gray code conversion circuit within Logisim Evolution.
    - Ensure the circuit accurately translates binary inputs into corresponding Gray code outputs.
    - Optimize the circuit for efficiency and minimal resource utilization.
    - Validate the functionality of the converter through rigorous testing and simulation.
    - Document the design process and findings comprehensively for future reference.
2. **Gray Code to Binary Converter:**
    - Develop a reliable Gray code to binary conversion circuit using Logisim Evolution.
    - Implement algorithms that accurately decode Gray code inputs into binary outputs.
    - Incorporate error-checking mechanisms to detect and handle invalid inputs effectively.
    - Evaluate the performance of the converter under various scenarios and input conditions.
    - Prepare detailed documentation outlining the design rationale, simulation results, and any potential limitations.
3. **BCD to 7-segment Decoder:**
    - Construct a BCD (Binary Coded Decimal) to 7-segment decoder circuit utilizing Logisim Evolution.
    - Define mappings between BCD inputs and corresponding 7-segment display outputs.
    - Implement efficient logic to facilitate seamless conversion and display of decimal digits.
    - Validate the decoder's functionality through systematic testing and simulation.
    - Explore opportunities for optimization and enhancements to improve performance and versatility.
4. **Ashesi Vending Machine:**
    - Create a virtual vending machine model, named "Ashesi Vending Machine," employing Logisim Evolution.
    - Define user interfaces for selecting products and processing transactions.

- Integrate payment mechanisms and inventory management functionalities into the vending machine simulation.
- Ensure the vending machine operates reliably and efficiently in diverse usage scenarios.
- Conduct thorough testing and validation to guarantee the robustness and accuracy of the vending machine's operation.

By addressing these key objectives, this term project aims to deepen our understanding of digital circuit design principles while showcasing the practical applications of Logisim Evolution in real-world scenarios. Through diligent exploration and implementation, we aspire to develop functional and optimized solutions for each module, enriching our knowledge and proficiency in digital logic design.

Part 1: Binary to Gray Code Converter

Introduction:

In the realm of digital electronics, data representation plays a pivotal role. Binary code, with its simplicity and effectiveness, is widely used to represent numerical and textual data in various digital systems. However, in certain applications, such as signal processing and communication systems, gray code presents advantages over binary code due to its property of minimizing errors during transitions, as being used in Karnaugh Map for Boolean simplification. The transition from binary to gray code is a fundamental operation in digital design. In this project, you will delve into the intricacies of binary-to-gray code conversion and explore how this conversion can be realized through a digital circuit.

Objective:

The primary objective of this project is to understand the concept of binary to gray code conversion and implement a digital circuit that facilitates this conversion. Through this project, you will:

1. Familiarize yourself with the binary and gray code representations.
2. Analyze the relationship between binary and gray code conversions.
3. Develop a strategy to convert binary numbers to gray code systematically.
4. Design a digital circuit using logic gates to perform the conversion from binary to gray code.
5. Gain hands-on experience in constructing and testing digital circuits for data transformation.

Question:

Your task is to design a digital circuit that converts 4-bit binary numbers to their corresponding gray code equivalents. Below is the truth table showing the conversion from binary to gray code for 4-bit numbers:

I. Identify the bit pattern changes and derive an expression with justification.

II. Draw the Binary to Gray Converter circuit in Logism evolution.

Binary (B3 B2 B1 B0)	Gray Code (G3 G2 G1 G0)
0000	0000
0001	0001
0010	0011
0011	0010
0100	0110
0101	0111
0110	0101
0111	0100
1000	1100
1001	1101
1010	1111
1011	1110
1100	1010
1101	1011
1110	1001
1111	1000

Fig 1. Binary to Gray Code Conversion

Using the provided truth table as a reference, design and draw the digital circuit that converts 4-bit binary inputs to their corresponding gray code outputs. Clearly label the components and connections in your circuit diagram.

Part 2: Gray Code to Binary Converter

Using the same truth table (Fig 1.) in Part A:

I. Identify the bit pattern changes and derive an expression with justification.
II. Draw the Gray to Binary Converter circuit in Logisim evolution.

Part 3: BCD to 7-segment Decoder

Introduction:

BCD: BCD stands for Binary Coded Decimal. It's a binary encoding of decimal numbers where each decimal digit is represented by a fixed number of binary bits. In BCD, each decimal digit

is encoded using a 4-bit binary number, allowing for easy conversion between binary and decimal representations.

For example, in BCD:

- Decimal 0 is represented as 0000.
- Decimal 1 is represented as 0001.

BCD is often used in digital systems where decimal arithmetic is required, such as calculators, digital clocks, and electronic meters. It allows for efficient and straightforward manipulation of decimal numbers in binary form.

7-segment display: A 7-segment display is a form of electronic display device commonly used to represent numbers, letters, and sometimes other characters. It consists of seven individual LED (Light Emitting Diode) segments arranged in a specific pattern, plus an additional optional eighth segment for displaying a decimal point. Each segment in a 7-segment display is labeled with a letter from a to g, with each letter corresponding to a specific LED segment. The arrangement of segments allows for the display of numerals 0 through 9, as well as a few additional characters.

In digital electronics, Binary Coded Decimal (BCD) is a commonly used binary encoding for decimal numbers. However, in many applications, it's essential to display BCD numbers in a human-readable format. A common way to achieve this is by using a 7-segment display, which consists of seven LED segments arranged in a specific pattern to represent decimal digits. In this project, you will explore the concept of BCD to 7-segment conversion and design a digital circuit to decode BCD numbers and display them on a 7-segment display.

Objective:

The primary objective of this project is to understand the BCD representation of decimal numbers and design a digital circuit that converts BCD input to the corresponding output for a 7-segment display. Through this project, you will:

1. Learn about the Binary Coded Decimal (BCD) encoding for decimal numbers.
2. Understand the principle of operation of a 7-segment display.
3. Explore the mapping between BCD digits and their representations on a 7-segment display.
4. Design a BCD to 7-segment decoder circuit using logic gates.
5. Gain practical experience in implementing and testing digital circuits for data conversion and display.

Consider the truth tables below:

BCD (BCD3 BCD2 BCD1 BCD0)	Segment Outputs (abcdefg)
0000	1111110
0001	0110000
0010	1101101
0011	1111001
0100	0110011
0101	1011011
0110	1011111
0111	1110000
1000	1111111
1001	1111011

Fig 2: BCD to 7-segment

Decimal Digit	a	b	c	d	e	f	g
0	1	1	1	1	1	1	0
1	0	1	1	0	0	0	0
2	1	1	0	1	1	0	1
3	1	1	1	1	0	0	1
4	0	1	1	0	0	1	1
5	1	0	1	1	0	1	1
6	1	0	1	1	1	1	1
7	1	1	1	0	0	0	0
8	1	1	1	1	1	1	1
9	1	1	1	1	0	1	1

Fig 3: Decimal to 7-segment

From the truth tables above:

I. Observe patterns for each segment and derive Boolean expressions for them based on the binary inputs (BCD digits).
II. Obtain the minimized expressions for each segment using the Karnaugh Map approach.
III. Design a logical circuit with minimized expressions. Your circuit should take 4-digit binary inputs (BCD) and outputs: a to g.
IV. Create a mask (or module) of your circuit in the previous part in a new worksheet. Using the 7-segment display in the Input/Output folder in Logisim, connect the outputs of your circuit to the 7-segment display to display the values in decimal equivalent for verification. The 7-segment display is illustrated below:

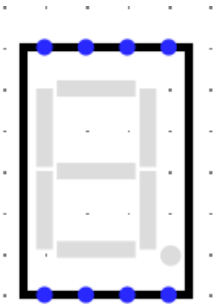

Fig 4: 7-segment display in Logisim Evolution.

V. From the figure above, each pin on the display circuit is assigned to a specific segment from a to g. Therefore, connect the outputs of your circuit to the appropriate segment pins for display.

Part 4: Ashesi Vending Machine

Introduction:

In today's modern world, vending machines have become an integral part of our daily lives, providing quick access to a variety of goods and services. From snacks and beverages to tickets and electronic gadgets, vending machines offer convenience and efficiency in transactions. In this part, you will leverage the power of Logisim Evolution to design and simulate a vending machine for Ashesi University. By creating a virtual representation of a vending machine, you understand the underlying digital logic and sequential processes involved in its operation. Through this simulation, you will explore the intricate details of vending machine functionality, from item selection to transaction processing.

Objective:

The primary objective of this project is to design and simulate a vending machine using Logisim Evolution. Specifically, we aim to achieve the following goals:

1. **Digital Logic Implementation:** Develop the digital logic circuits necessary to emulate the behavior of a vending machine, including coin detection, item selection, and transaction processing.
2. **User Interaction:** Enable user interaction with the virtual vending machine, allowing them to select items and input coins to initiate transactions.
3. **Transaction Handling:** Implement the transaction handling mechanism to process user requests, deduct appropriate amounts from the deposited coins, and dispense selected items.
4. **Error Handling:** Incorporate error detection and recovery mechanisms to handle situations such as insufficient funds, invalid item selection, or coin jams gracefully.
5. **Simulation and Testing:** Conduct thorough simulations and testing to verify the correctness and robustness of the vending machine design under various scenarios and input conditions.

Task:

You will be provided with a starter file (Logisim circuit file), from which you will update to realize a fully functional vending machine. Let us briefly go through each circuit in the starter file:

Bank:

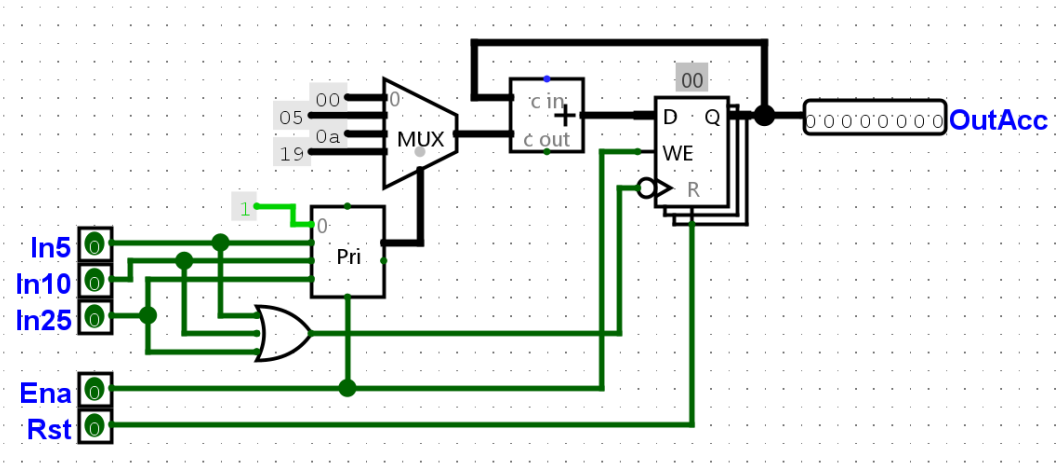

Fig 5: Bank circuit in Vending machine

The bank circuit is where the money is deposited and totaled. It contains the following key features:

- Nickel input – 5 cents
- Dime input – 10 cents
- Quarter input – 25 cents
- Priority Encoder: This is a 4-input encoder that feeds a 2-bit (address/control/signal) of the highest input turned on into the multiplexer. It feeds the 2-bit control when the enable input is turned on.
- Multiplexer: The 4-input mux takes the inputs corresponding to the amount of the coins. It takes the 2-bit control from the priority encoder to determine which input (coin amount in hex) should be fed into the adder.
- Adder (built-in Logisim from the Arithmetic folder)c: The adder accepts the amount from the mux as one input and feedback from the D-Flip Flop (which holds the current amount stored) as the second input, so that our D-Flip flop used our present state and the new input to determine the next state (new value).
- Enable input: Turns on the 4:2 priority encoder to feed to select the appropriate coin. It also enables the D-Flip Flop to store the current amount.
- Reset input: Clears the memory (D-Flip Flop) to zero when high.
- The clock is activated anytime a coin button is pushed. From the figure, the coin inputs are ORed, and the output activates the clock, meaning that the clock is activated anytime at least one coin button is activated.
- D-Flip Flop: This acts as the memory element to store the total coins deposited. The clock is triggered at the falling edge (when the input coin is pushed on and off)

Dispenser:

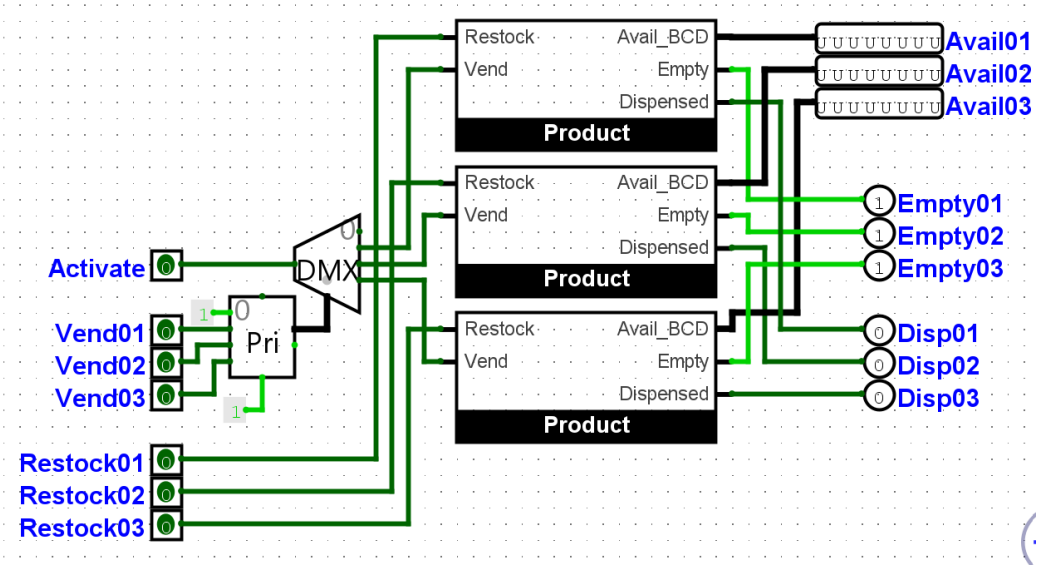

Fig 5: Dispenser circuit in Vending machine

The dispenser circuit is what dispenses the product. There are 3 products, and based on the user's activation, the corresponding product will be released. It contains the following key features:

- Activate signal: When high, is fed into the demultiplexer.
- Vend01, Vend02, Vend03: These are the input buttons representing each product. When Vend01 is activated, then we are interested in dispensing the first product.
- Priority encoder: Like its behavior in the bank circuit, it sends the control signal to the ***demux***.
- Demultiplexer: The ***demux*** accepts the control signal from the priority encoder and determines which of the 3 products should be activated and dispensed.
- Restock1, Restock2, Restock3. Allows for restocking of each of the products automatically with 15 products.
- Product mask/module: 3 product circuit components are used in the creation of the dispenser. The product circuit takes the restock and vend buttons as inputs and outputs the available number for that product left, an empty signal, and a dispenser signal.
- Avail01, Avail02, Avail03: Determines the number of available items in each product category.

Product:

The product circuit in the figure below contains the following key features:

- Counter: Contains the number of products available to vend. When you click on the counter, the maximum value is 0xf, meaning that it can only count up to 15, just as indicated earlier that our products are at most 15. The data loaded into the counter is f (1111) and the output is a 4-bit binary number of the current value in the counter.

- Restock button: When the restock is activated, the counter is automatically reset to f (15).
- Vend: When activated, the counter subtracts 1 from its current value to reduce the total number for a specific product.
- Empty button (output): Activated whenever the counter is empty.
- The output of the counter must be fed into a Binary to BCD converter for easy reading.
- The machine is set up so that every product is dispensed at 75 cents.

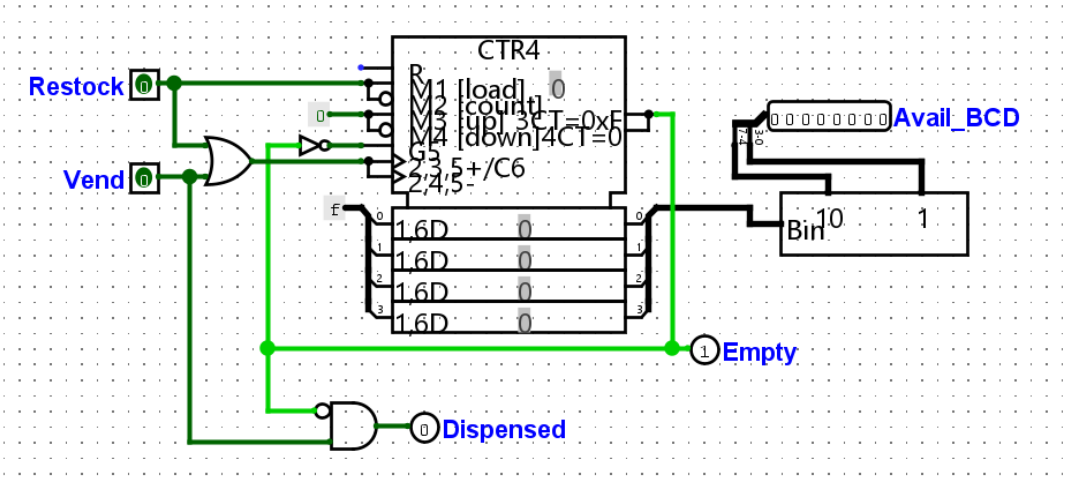

Fig 6: Product Circuit in Vending Machine

Activator:

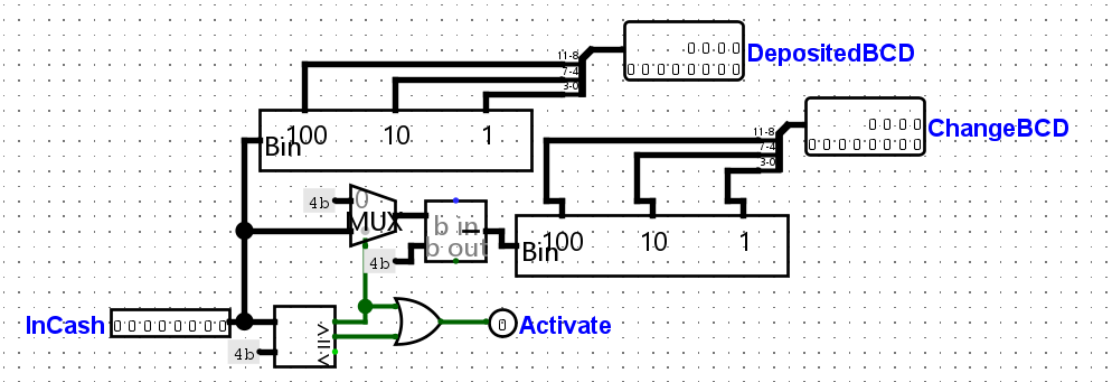

Fig 7: Activator circuit in Vending Machine

The activator is what turns on the vending machine so that the user can select a product. It has the following key features:

The cash input is compared with 4b (75) using a comparator. Since every product costs 75 cents, then the customer needs at least 75 cents to be able to select a product. Therefore, the activate button is activated when at least 75 cents is deposited into the machine.

If the amount deposited is more than 75 cents, it sends the control (1) to the 2-input multiplexer, which selects the amount deposited as output and subtracts 75 cents (4b) from the amount

deposited using the subtractor (from the arithmetic folder) circuit. The output of the subtractor circuit is then fed into a binary to BCD converter to display the change due (as a 12-bit output).

Also, the cash deposited is fed into a binary to BCD converter to display how much money is deposited (as a 12-bit output).

Vending:

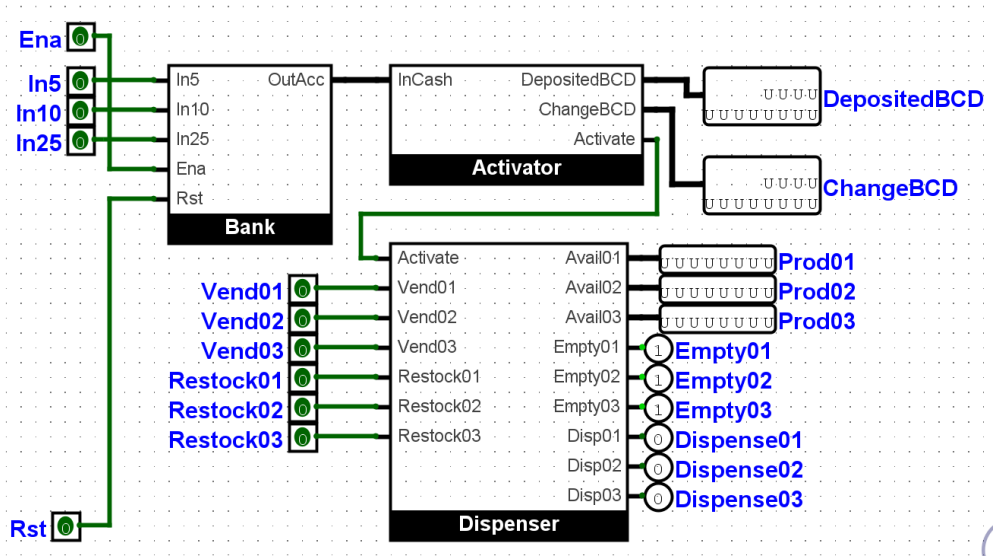

Fig 8: Vending Circuit

- The vending circuit contains all the sub-circuits described above as illustrated in the figure above. All the components are assembled to make a functional vending machine.
- The current amount deposited in the bank circuit is fed in as the cash to activate which activates the dispenser to allow the user to select a product to be dispensed.

Main:

This is where we simulate the vending machine in a human-friendly interface by the addition of hex-digit displays to display the amounts in human-readable forms as shown in the figure below:

- The enable button (Ena) must be activated to turn on the vending machine.
- Restocking product inserts automatically inserts 15 into the vending machine.
- Every product costs 75 cents.
- When a product is dispensed, it's dispense button lights momentarily.

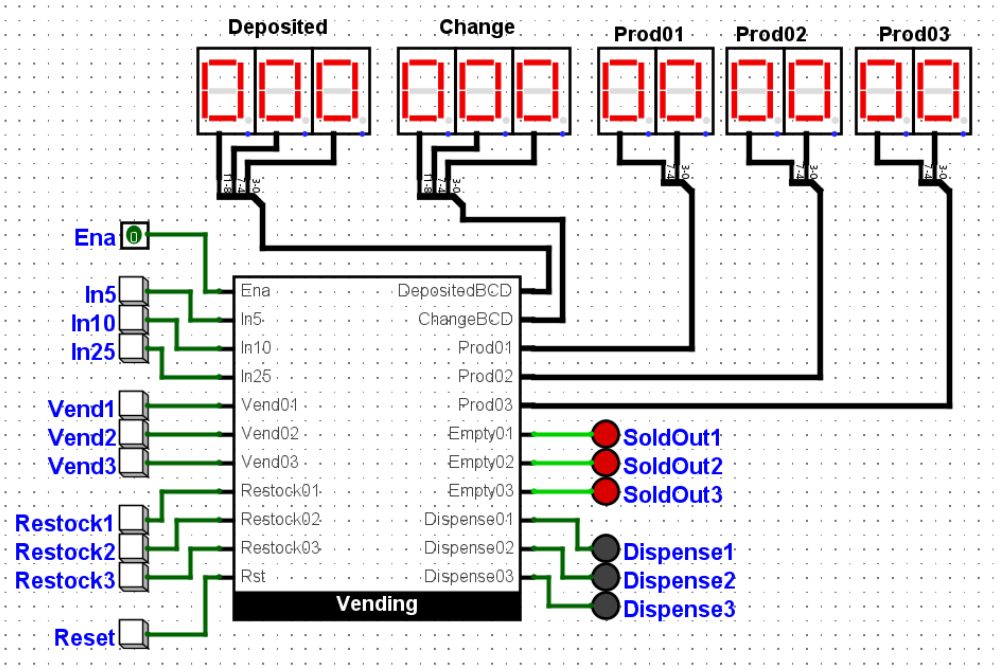

Fig 9: Main Circuit for Vending Simulation

Challenges:

In this section, we outline 3 problems associated with the overall vending machine that you need to correct as your activity for this part.

Issue 1: When an amount of more than 75 cents is deposited, the machine displays the amount and the change due. An example is when you deposit 100 cents and the change due is 25 cents. The issue here is that anytime a user dispenses a product, the product count reduces, but the deposited amount and change do not update. This means that a customer can keep vending more products without inputting any more money. The customer may enjoy that, but of course, your team may not like that. Your task is to:

- Update the circuit such that when a customer vends a product, the deposited amount is reset to zero. This will in turn inactivate the vend buttons because we have seen that the vend is not activated until the amount deposited is at least 75 cents. Therefore, the customer can no longer vend more products but only receive his/her change due.

Issue 2: When a customer deposits 85 cents, the machine allows him/her to continuously deposit additional coins. For instance, if he/she clicks on 10 again, then the amount deposited increases to 95. Of course, your team may enjoy that since the customer can keep putting in more money. However, the customer is not going to like that. Your task is to:

- Update the circuit such that there is a stop once 75 cents has been deposited and the customer cannot keep putting money into the machine.

Issue 3: We also want your machine to display the total amount of money deposited so that when the service technician goes to service the machine, the technician can see the total amount

of money deposited in the coin can. This is simply a check on the system so that your team always knows the total amount of money in the machine periodically. Your task is to:

- Update the circuit to have a display for the total amount (16 bits) deposited. Therefore, you will need 4 hex-digit displays to show the total amount in the machine, as illustrated in the image below:

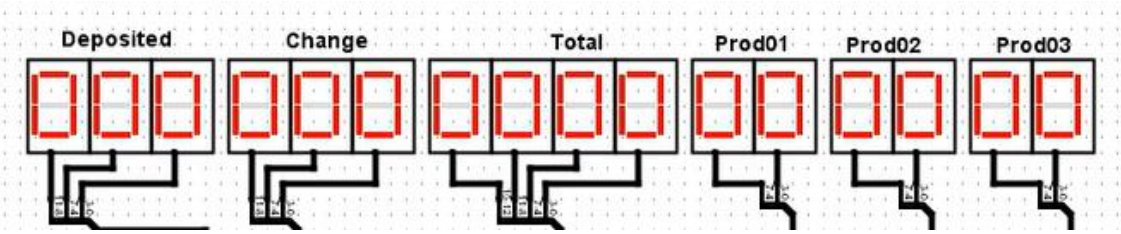

Fig 10: Total amount display added to the machine.

Summary:

You have been provided with the starter circuit file on Canvas, named **Ashesi_Vending_Machine,** that simulates a functional vending machine. Your task is to fix the 3 issues explained above in the circuit.

Part 5: Reflection Questions

1. What challenges did you encounter during the design of the Binary to Gray Code Converter, and how did you overcome them?
2. What considerations influenced the choice of error-checking mechanisms in the Gray Code to Binary Converter?
3. How did you define the mapping between BCD inputs and 7-segment display outputs in the BCD to 7-segment Decoder?
4. What strategies did you employ to simulate and validate the functionality of the Ashesi Vending Machine?
5. How did you prioritize optimization efforts in each module to ensure efficient resource utilization?
6. In what ways did this project enhance your understanding of digital circuit design principles and the practical applications of Logisim Evolution?

Submission:

Submit a zip file (named as ***Group_Number_Project***) containing:

- Separate circuit files for each part. Therefore, your zip file should contain 4 Logisim-evolution files for parts 1 – 4, respectively.
- PDF of responses to reflection questions

You are encouraged to distribute the work among the group members and ensure that you complete the project with minimum delay. All the best!