VIET NAM NATIONAL UNIVERSITY HO CHI MINH CITY – UNIVERSITY OF TECHNOLOGY

FACULTY OF ELECTRICAL AND ELECTRONICS ENGINEERING

DEPARTMENT OF ELECTRONICS

MILESTONE 1

COMPUTER ARCHITECTURE

Instructor: Dr. Tran Hoang Linh

Student name	**Student ID**
Nguyen Dinh Huy	2211208
Nguyen Ha Phong Thinh	2151143
Dao Nguyen Khoi	2151215

Ho Chinh Minh, September 2025

Contents

1. Introduction

In this project, we designed a vending machine using SystemVerilog. The machine accepts coin inputs of 5¢ (nickel), 10¢ (dime), and 25¢ (quarter). Once the accumulated value reaches or exceeds 20¢, the machine outputs one soda and returns the remaining amount as change.

2. Design methodology

2.1. Coding style

- Followed the lowRISC Verilog style guide.
- Clear and consistent signal naming.
- Clock-based operation

2.2. Design Approach

Instead of using an FSM (Finite State Machine), we implemented the design using an accumulator register and combinational logic.

Reasons for not using FSM:

- The problem is relatively simple: it only requires accumulating money and comparing it against a fixed threshold (20¢).
- An FSM would require at least one additional clock cycle to move from the "accumulation" state to the "dispense" state. This delays the soda and change output unnecessarily.
- In contrast, by directly using a comparator on the accumulated value, the design can immediately output soda and change as soon as the total amount is greater than or equal to 20¢.
- Compare 2 waveforms:

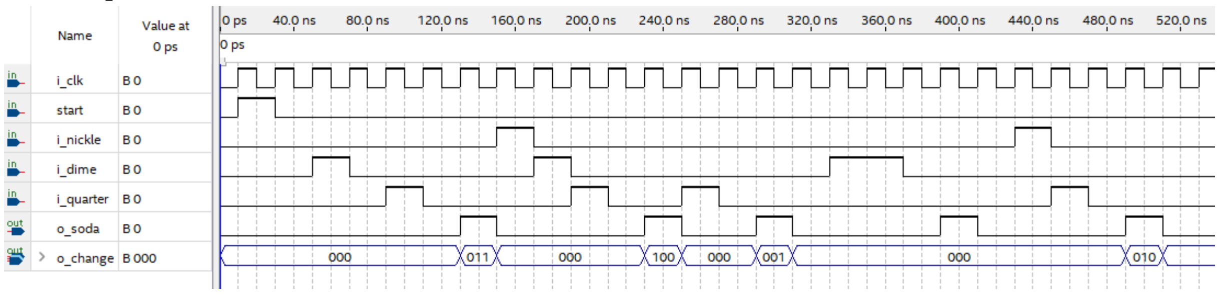


Figure 1: With FSM

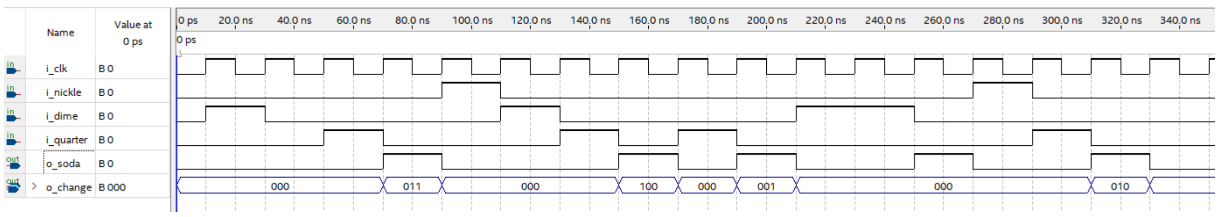


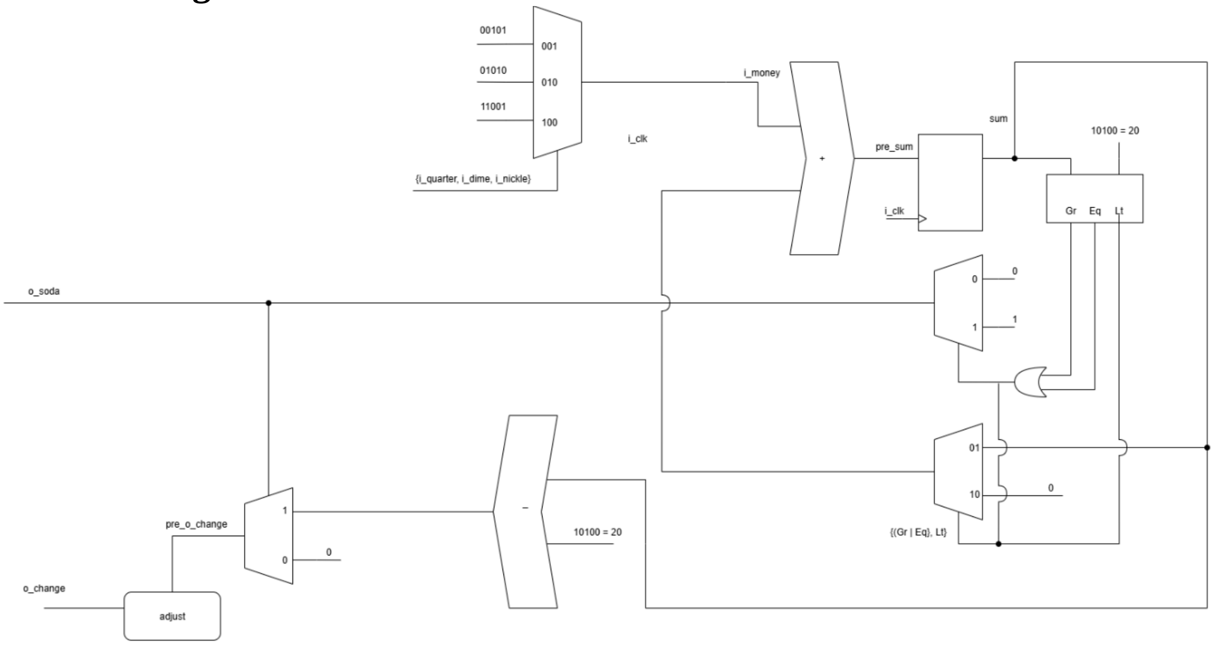
Figure 2: Without FSM

3. I/O Description

Signal	I/O	Width	Description
i_clk	Input	1 bit	System clock signal
i_nickle	Input	1 bit	Insert nickel (5¢)
i_dime	Input	1 bit	Insert dime (10¢)
i_quarter	Input	1 bit	Insert quarter (25¢)
o_soda	Output	1 bit	Dispense soda (1 = soda released)
o_change	Output	3 bits	Value of returned change

Table 1: Inputs and Outputs

4. Block diagram



First, the system uses a multiplexer to differentiate between nickel, dime, and quarter inputs. The selected value is then accumulated step by step until the comparator detects that the total is greater than or equal to 20.

If the total is greater than or equal to 20, the output o_soda will be set to 1, and the system will reset the accumulated value back to 0. At the same time, the inserted amount is subtracted by 20 using a full adder to calculate the remaining balance, which is then assigned to o_change.

5. Simulation result

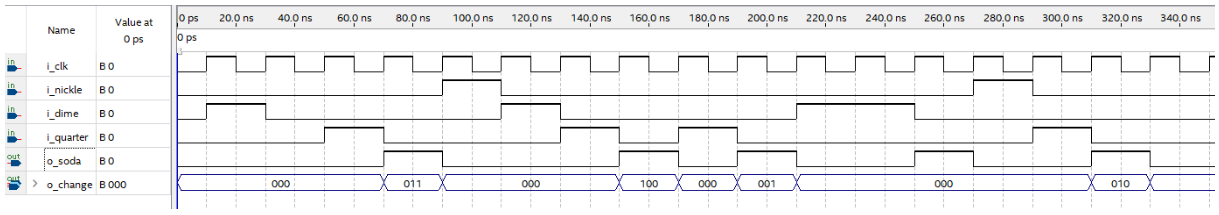


Figure 3: Waveform

To verify the correctness of the design, five test cases were simulated. In each case, the inputs represent the sequence of inserted coins, and the outputs were checked for both o_soda and o_change.

1. Dime + Quarter → Change = 011 (3¢) → Correct
2. Nickel + Dime + Quarter → Change = 100 (4¢) → Correct
3. Quarter → Change = 001 (1¢) → Correct
4. Dime + Dime → Change = 000 (0¢) → Correct
5. Nickel + Quarter → Change = 010 (2¢) → Correct

The simulation confirms that the system correctly dispenses soda and calculates change across all test scenarios.

6. RTL View check

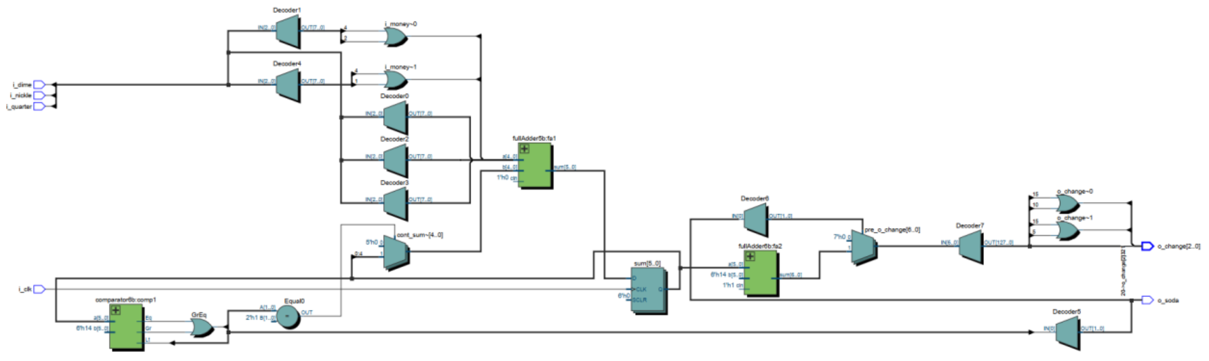


Figure 4: RTL view

7. Conclusion

In this project, a soda vending machine was successfully designed without using an FSM, instead relying on an accumulator and comparator to simplify the logic and reduce unnecessary clock delays. The system was verified through simulation with multiple test cases, all producing correct outputs for soda dispensing and change calculation. This demonstrates that the design works reliably and efficiently.

For further details and to review the source code, please visit the GitHub repository: [huynguyendinhhcmut/Milestone1_Vending-Machine]