FACULTY OF ENGINEERING

EE415/BEE3053/BER3063 DIGITAL SYSTEM & HDLS

ASSIGNMENT

MAY - AUGUST 2024

Date :

Student name & ID : Zaid Mustafa Zaidan, 1001955964

Fuad Muheeb Sawaked, 1001953463

Ibrahim Dayyah, 1002059744

Assessment	**CLO1**	**CLO2**	**CLO3**	**CLO4**	**CLO5**
	PLO2	**PLO2**	**PLO2**	**PLO2**	**PLO2**
Assignment		√	√	√	√
Course Learning Outcome	CLO1: Evaluate the Hardware Description Language (HDL) of a digital logic design. CLO2: Evaluate the arithmetic circuit design with Verilog HDL. CLO3: Evaluate the combinational circuit design with Verilog HDL. CLO4: Evaluate registers and counters design with Verilog HDL. CLO5: Analyze sequential circuit for the finite state machine (FSM) with HDL.				
Programme Learning Outcome	PLO2: ***PROBLEM ANALYSIS*** - Identify, formulate, conduct research literature and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.				

Department of Electrical and Electronic Engineering
Assignment Marking Rubric

Criteria	Unsatisfactory	Average	Good	Excellent	Score
Introduction & Objective [5 Marks]	0	1 – 2	3 - 4	5	
	No introduction and objectives.	Weak explanation on introduction and objectives.	Appropriately structured introduction and objectives.	Excellent explanation on introduction and objectives.	
Methodology & Coding [25 Marks]	0 – 6	7 – 12	13 – 18	19 – 25	
	Unclear methodology. Errors in coding.	Average description of methodology. Only partial coding was done.	Clear description of methodology, with flow chart. All coding was done and running with the software.	Excellent description of the methodology, with clear flow chart. All coding was efficiently written and running with the software.	
Result [25 Marks]	0 – 6	7 – 12	13 – 18	19 – 25	
	No figures, graphs, tables are provided.	Most figures, graphs, tables OK, some still missing some important or required features.	All figures, graphs, tables are correctly drawn, but some have minor problems or could still be improved.	All figures, graphs, tables are correctly drawn, are numbered and contain titles/captions.	
Discussion & Analysis [20 Marks]	0 – 5	6 – 10	11 - 15	16 – 20	
	No discussion or incomplete and incorrect interpretation of results.	Some of the results have been correctly interpreted and discussed; Partial but incomplete understanding of results is still evident.	Almost all of the results have been correctly interpreted and discussed; only minor improvements are needed.	Important trends and results have been interpreted correctly and discussed; good understanding of results is conveyed.	
Conclusion [10 Marks]	0 – 2	3 – 5	6 – 8	9 – 10	
	Incomplete and incorrect conclusion.	Major points are drawn, but many are misstated, Indicating a lack of understanding.	Conclusions have been drawn, could be better stated.	Conclusions have been clearly made; student shows good understanding.	
Written Skills, Format, References [5 Marks]	0	1 – 2	3 - 4	5	
	Poor in report writing and no references included.	Moderate in report writing and the references provide a limited discussion of the literature available and may be out dated.	Good in report writing and satisfactory referencing has been carried out.	Excellent in report writing and very good use of references with a highly appropriate coverage which is relevant, and properly referenced.	
Presentation [10 Marks]	0 – 2	3 – 5	6 – 8	9 – 10	
	Did not present.	Presentation unconvincing & not methodical; able to answer < 50% of questions posed.	Well groomed; appropriately presented key points; able to answer > 50% of questions posed.	Well groomed; convincing in presentation; appropriately presented key points; able to answer all questions posed.	
				Total Marks (100)	
				Final Score (%) = (Total Marks / 100) * 30%	

Bank Ticket Machine:

1. Design a digital system for a Bank Ticket Machine using Verilog HDL on Quartus software, with the following specification:
    - Dispense ticket for at least 3 different types of bank services. i.e., each type of service will dispense a ticket starting with different number or alphabet.
    - Consider at least 4 bank officers on duty.
    - The bank officer press a button to get the next queuing number.
    - Display the next ticket number on 7-segment display with the bank officer's desk number.
    - The ticket shows the number of waiting customers.
2. Use the finite state machine (FSM) to design the Bank Ticket Machine. Your design may require an arithmetic circuit, combinational circuit, register, and counter.
3. Show your designed system with a flow chart.
4. Verify the functionality of your design using ModelSim software.

General Instruction:

- Submit to CN link by **3rd August 2024**.
- Late submission is allowed up to 5 days (latest by 8th August 2024) with a deduction of 5% out of the 30%. After 8th April 2024, no mark for the group.
- This is an open-ended assignment.
- No group should have the same Verilog coding for the circuit design. If plagiarism is found between groups, marks will be deducted depending on the amount of similarity.
- A group assignment with maximum 3 students in a group.
- Submit one report per group, maximum 50 pages.
- The report must include the **Cover Page** and **Marking Rubric** (page 1-2).
- Compress all the following and submit only **ONE** **zip** or **rar** file, by one of the group members:
    - Assignment Report (in **PDF**)
    - Quartus Project File
    - ModelSim Project File
- Label the compressed file with your group members' name, i.e., ***Mark_Nadia_John.rar***
- Group presentation at Week 14 by all group members.

Table of Contents

1. Introduction

A bank ticket machine is an automated device that is employed by financial organizations to expedite and control customer movement within their branches. These machines are also known as queuing systems or token systems. Customers are provided with numbered tickets by these machines upon their arrival, which enables them to wait for their turn lessening the amount of congestion that occurs in banking halls, and making sure that each customer receives attention in a methodical and equitable manner.

It is possible for banks to improve the overall customer experience and increase the efficiency of their operations by organizing and prioritizing customers based on ticket numbers.

When it comes to daily operations, ticket machines give financial institutions the ability to analyze and improve the delivery of their services. It is possible for the data generated from these systems to provide significant insights into peak service periods, client wait times, and staff performance. This enables better resource allocation and greater customer satisfaction. In a more general sense, the installation of ticket machines is a reflection of a bank's dedication to utilizing technology in order to improve the quality of service and the efficiency of operations.

2. Objectives

In this project, Verilog Hardware Description Language is used to program the bank ticket machine. Quartus Prime 20.1 is the software utilized to program the Verilog code and run the analysis. The design of the circuit consists of arithmetic circuits, combinational circuits, registers, and counters. In the assignment noted as qpf, and several Verilog codes are combined as submodules to program the ticket machine. A flow chart and a Finite State Machine diagram are constructed to explain the operations of the machine. Modelsim software is used as a Verilog simulation and verification tool. The inputs, wires, and output waveforms were added to the software for verification purposes. It can also be used to debug and increase the design quality. The objective of the project is to enable the students to understand the relationship between programming and coding on an embedded system. In addition, it allows the students to be exposed to a deeper knowledge of the Verilog language by being practically involved in the assignment. Moreover, the assignment is intended to develop and engage student’s creative thinking skills during the designing circuit phase.

3. Methodology & Coding

3.1 Methodology

The methodology employed in this project involved a structured approach to design, implement, and test the Bank Ticket Machine system. Initially, the overall system architecture was conceptualized, comprising three main modules: the finite state machine (FSM), the seven-segment display controller, and the testbench. The FSM was designed to manage state transitions based on the input from officer buttons, featuring three distinct states: IDLE, DISPENSE_TICKET, and UPDATE_DISPLAY. In the IDLE state, the system remains inactive until an officer button is pressed. Upon detecting a button press, the system transitions to the DISPENSE_TICKET state, where it increments the ticket number and updates the officer number, service type, and waiting customer count. Finally, in the UPDATE_DISPLAY state, the system updates the seven-segment displays with the current values before returning to the IDLE state.

The seven-segment display controller was developed to handle the visualization of the ticket number, officer number, service type, and waiting customer count across four displays. This module converted the values received from the FSM into the corresponding seven-segment encoding, ensuring clear and accurate display of information. Each aspect of the display, including the ticket number, officer number, service type, and waiting customer count, was meticulously encoded using combinational logic in Verilog HDL.

To verify the functionality of the system, a comprehensive testbench was created. This testbench simulated various operational scenarios by providing clock and reset signals and simulating button presses. It ensured the system's responses were as expected by checking the outputs of the seven-segment displays. The simulation was executed using ModelSim, with a 100 MHz clock signal generated to drive the system. The testbench scenarios included no button presses, individual officer button presses, and multiple button presses. The outputs were monitored and displayed for verification purposes.

The simulation process involved compiling the Verilog HDL code and loading it into ModelSim for execution. The waveforms of the clock, reset, officer buttons, and display outputs were scrutinized to ensure correct system behavior. The simulation results confirmed that the system transitioned between states accurately and updated the displays as intended.

Each test scenario's outputs were compared with expected results, validating the system's functionality and accuracy.

In conclusion, the methodology for developing the Bank Ticket Machine system was methodical and thorough, encompassing system design, module development, simulation, and verification. The FSM and display controller were effectively implemented in Verilog HDL, and their functionality was rigorously tested and validated using a detailed testbench in ModelSim. The simulation results demonstrated the system's correct operation, fulfilling the project objectives and requirements.

Table 1. Variables used in the code and their functions.

Variable	*Type*	*Scope*	*Function*
clk	Input	All Modules	Clock signal for synchronization.
reset	Input	All Modules	Signal to reset the system.
officer_button	Input	BankTicketMachine, fsm	4-bit input representing buttons for selecting an officer.
seg_ticket	Output	BankTicketMachine, seven_segment_display	7-bit output for the ticket number display.
seg_officer	Output	BankTicketMachine, seven_segment_display	7-bit output for the officer number display.
seg_service	Output	BankTicketMachine, seven_segment_display	7-bit output for the service type display.
seg_waiting	Output	BankTicketMachine, seven_segment_display	7-bit output for the waiting customers display.
ticket_number	Wire	All Modules	7-bit wire for the ticket number.
officer_number	Wire	All Modules	2-bit wire for the officer number.
waiting_customers	Wire	All Modules	4-bit wire for the number of waiting customers.
service_type	Wire	All Modules	2-bit wire for the service type.
current_state	Reg	fsm	Register for managing the current FSM state.
next_state	Reg	fsm	Register for managing the next FSM state.
IDLE	Parameter	fsm	FSM state representing idle state.
DISPENSE_TICKET	Parameter	fsm	FSM state representing ticket dispensing state.
UPDATE_DISPLAY	Parameter	fsm	FSM state representing display update state.

3.3 Code

BankTicketMachine.v

```v
module BankTicketMachine (
    input clk,
    input reset,
    input [3:0] officer_button,
    output [6:0] seg_ticket,
    output [6:0] seg_officer,
    output [6:0] seg_service, // New output for service type display
    output [6:0] seg_waiting // New output for waiting customers display
);

    wire [6:0] ticket_number;
    wire [1:0] officer_number;
    wire [3:0] waiting_customers;
    wire [1:0] service_type; // New wire for service type

    // Instantiate FSM module
    fsm fsm_inst (
        .clk(clk),
        .reset(reset),
        .officer_button(officer_button),
        .ticket_number(ticket_number),
        .officer_number(officer_number),
        .waiting_customers(waiting_customers),
        .service_type(service_type) // Connect service type
    );

    // Instantiate 7-segment display module
    seven_segment_display display_inst (
        .ticket_number(ticket_number),
        .officer_number(officer_number),
        .service_type(service_type), // Connect service type
        .waiting_customers(waiting_customers), // Connect waiting customers
        .seg_ticket(seg_ticket),
        .seg_officer(seg_officer),
        .seg_service(seg_service), // Connect service type display
        .seg_waiting(seg_waiting) // Connect waiting customers display
    );

endmodule


module fsm (
    input clk,
    input reset,
    input [3:0] officer_button,
    output reg [6:0] ticket_number,
    output reg [1:0] officer_number,
    output reg [3:0] waiting_customers,
    output reg [1:0] service_type // New output to indicate the service type
);

    // State encoding using parameters
    parameter IDLE = 2'b00, DISPENSE_TICKET = 2'b01, UPDATE_DISPLAY = 2'b10;

    reg [1:0] current_state, next_state;

    // State transition
    always @(posedge clk or posedge reset) begin
        if (reset)
            current_state <= IDLE;
        else
            current_state <= next_state;
```

```
63          end
64
65          // Next state logic and outputs
66          always @(*) begin
67              next_state = current_state;
68              case (current_state)
69                  IDLE: begin
70                      if (officer_button != 4'b0000) begin
71                          next_state = DISPENSE_TICKET;
72                      end else begin
73                          next_state = IDLE;
74                      end
75                  end
76                  DISPENSE_TICKET: begin
77                      next_state = UPDATE_DISPLAY;
78                  end
79                  UPDATE_DISPLAY: begin
80                      next_state = IDLE;
81                  end
82                  default: begin
83                      next_state = IDLE;
84                  end
85              endcase
86          end
87
88          // Output logic
89          always @(posedge clk or posedge reset) begin
90              if (reset) begin
91                  ticket_number <= 7'b0000000;
92                  waiting_customers <= 4'b0000;
93                  officer_number <= 2'b00;
94                  service_type <= 2'b00; // Default to Service A
95              end else if (current_state == DISPENSE_TICKET) begin
96                  ticket_number <= ticket_number + 1;
97                  waiting_customers <= waiting_customers + 1;
98                  officer_number <= (officer_button[0]) ? 2'b00 :
99                                    (officer_button[1]) ? 2'b01 :
100                                   (officer_button[2]) ? 2'b10 :
101                                   (officer_button[3]) ? 2'b11 : officer_number;
102                 service_type <= (officer_button[0]) ? 2'b00 : // Service A
103                                 (officer_button[1]) ? 2'b01 : // Service B
104                                 (officer_button[2]) ? 2'b10 : // Service C
105                                 (officer_button[3]) ? 2'b11 : service_type; // Service D
106             end else begin
107                 // Retain current values when no state change
108                 ticket_number <= ticket_number;
109                 waiting_customers <= waiting_customers;
110                 officer_number <= officer_number;
111                 service_type <= service_type;
112             end
113         end
114
115 endmodule
116
117
118
119
120 module seven_segment_display (
121     input [6:0] ticket_number,
122     input [1:0] officer_number,
123     input [1:0] service_type, // New input for service type
124     input [3:0] waiting_customers, // New input for waiting customers
```

BankTicketMachine.v

```
125     output reg [6:0] seg_ticket,
126     output reg [6:0] seg_officer,
127     output reg [6:0] seg_service, // New output for service type display
128     output reg [6:0] seg_waiting // New output for waiting customers display
129 );
130
131     // 7-segment encoding for ticket number
132     always @(*) begin
133         case (ticket_number)
134             7'b0000000: seg_ticket = 7'b1000000; // 0
135             7'b0000001: seg_ticket = 7'b1111001; // 1
136             7'b0000010: seg_ticket = 7'b0100100; // 2
137             7'b0000011: seg_ticket = 7'b0110000; // 3
138             7'b0000100: seg_ticket = 7'b0011001; // 4
139             7'b0000101: seg_ticket = 7'b0010010; // 5
140             7'b0000110: seg_ticket = 7'b0000010; // 6
141             7'b0000111: seg_ticket = 7'b1111000; // 7
142             default: seg_ticket = 7'b0000000;
143         endcase
144     end
145
146     // 7-segment encoding for officer number
147     always @(*) begin
148         case (officer_number)
149             2'b00: seg_officer = 7'b1000000; // 0
150             2'b01: seg_officer = 7'b1111001; // 1
151             2'b10: seg_officer = 7'b0100100; // 2
152             2'b11: seg_officer = 7'b0110000; // 3
153             default: seg_officer = 7'b0000000;
156
157     // 7-segment encoding for service type
158     always @(*) begin
159         case (service_type)
160             2'b00: seg_service = 7'b1110111; // A
161             2'b01: seg_service = 7'b0011100; // B
162             2'b10: seg_service = 7'b0000110; // C
163             2'b11: seg_service = 7'b0100001; // D
164             default: seg_service = 7'b0000000;
165         endcase
166     end
167
168     // 7-segment encoding for waiting customers
169     always @(*) begin
170         case (waiting_customers)
171             4'b0000: seg_waiting = 7'b1000000; // 0
172             4'b0001: seg_waiting = 7'b1111001; // 1
173             4'b0010: seg_waiting = 7'b0100100; // 2
174             4'b0011: seg_waiting = 7'b0110000; // 3
175             4'b0100: seg_waiting = 7'b0011001; // 4
176             4'b0101: seg_waiting = 7'b0010010; // 5
177             4'b0110: seg_waiting = 7'b0000010; // 6
178             4'b0111: seg_waiting = 7'b1111000; // 7
179             default: seg_waiting = 7'b0000000;
180         endcase
181     end
182
183 endmodule
184
```

Figure 1: Project Code; top module, and both sub-modules

The BankTicketMachine system is a digital design that is meant to imitate the functions of a token or a ticket machine. The handling of ticket issuing and the display of relevant information on seven-segment screens are both handled by this system, which integrates a number of different components and operations. The top-level module's name is BankTicketMachine, the FSM module's name is fsm, and the seven-segment display module's name is seven_segment_display. These are the important components of the system.

For the purpose of integrating the FSM and the seven-segment display modules, the top-level module is responsible for connecting the inputs of the system to the FSM and for passing the outputs of the FSM to the display module. The inputs consist of a clock signal (clk), a reset signal (reset), and an officer button input (officer_button) that is four bits in length (4 bits [3:0]). The outputs consist of four seven-segment display signals: seg_ticket, seg_officer, seg_service, and seg_waiting. These signals represent the number of customers waiting, the number of officers handling the ticket, the type of service being provided, and the number of customers waiting.

The systems' primary logic controller is the FSM module. It controls the state transitions and output logic using input signals. The FSM states are IDLE, DISPENSE_TICKET, and UPDATE_DISPLAY. The system then waits for the officer button press in IDLE. The FSM enters the DISPENSE_TICKET state when an officer button is pressed, incrementing the ticket number, waiting customers, officer number, and service type. The system then updates display outputs in UPDATE_DISPLAY and returns to IDLE. This ensures the system issues tickets accurately and refreshes displays.

The seven-segment display module converts the ticket_number, officer_number, service_type, and waiting_customer binary values into its display codes. It employs ‘always’ blocks with ‘case’ statements to convert FSM binary inputs into 7-segment display formats. The ticket_number, officer_number, service_type, and waiting_customers are encoded to display values from 0 to 7.

Testbench code

tb_bank_ticket_machine.v

```v
module tb_bank_ticket_machine;

    // Inputs
    reg clk;
    reg reset;
    reg [3:0] officer_button;

    // Outputs
    wire [6:0] seg_ticket;
    wire [6:0] seg_officer;
    wire [6:0] seg_service;
    wire [6:0] seg_waiting;

    // Instantiate the Unit Under Test (UUT)
    BankTicketMachine uut (
        .clk(clk),
        .reset(reset),
        .officer_button(officer_button),
        .seg_ticket(seg_ticket),
        .seg_officer(seg_officer),
        .seg_service(seg_service),
        .seg_waiting(seg_waiting)
    );

    initial begin
        // Initialize Inputs
        clk = 0;
        reset = 1;
        officer_button = 0;

        // Wait for global reset
        #20;
        reset = 0;

        // Test No button pressed
        #50;
        $display("No button pressed: seg_ticket = %b, seg_officer = %b, seg_service = %b, seg_waiting = %b",
                 seg_ticket, seg_officer, seg_service, seg_waiting);

        // Test Officer 1 pressed
        #50;
        officer_button = 4'b0001;
        #50;
        officer_button = 4'b0000;
        $display("Officer 1 pressed: seg_ticket = %b, seg_officer = %b, seg_service = %b, seg_waiting = %b",
                 seg_ticket, seg_officer, seg_service, seg_waiting);

        // Test Officer 2 pressed
        #50;
        officer_button = 4'b0010;
        #50;
        officer_button = 4'b0000;
        $display("Officer 2 pressed: seg_ticket = %b, seg_officer = %b, seg_service = %b, seg_waiting = %b",
                 seg_ticket, seg_officer, seg_service, seg_waiting);

        // Test Officer 3 pressed
        #50;
        officer_button = 4'b0100;
        #50;
        officer_button = 4'b0000;
        $display("Officer 3 pressed: seg_ticket = %b, seg_officer = %b, seg_service = %b, seg_waiting = %b",
                 seg_ticket, seg_officer, seg_service, seg_waiting);
```

```
63
64          // Test Officer 4 pressed
65          #50;
66          officer_button = 4'b1000;
67          #50;
68          officer_button = 4'b0000;
69          $display("Officer 4 pressed: seg_ticket = %b, seg_officer = %b, seg_service = %b, seg_waiting = %b",
70                   seg_ticket, seg_officer, seg_service, seg_waiting);
71
72          // Finish simulation
73          #50;
74          $stop;
75      end
76
77      always #5 clk = ~clk; // 100MHz clock
78
79  endmodule
80
```

Figure 2: Testbench Code

In order to simulate the behavior of the machine in a variety of different situations, the testbench code that has been given for the Bank Ticket Machine system has been used. A clock (clk), reset (reset), and officer button inputs (officer_button) are all defined by the testbench, which is responsible for initializing the system. The Unit Under Test (UUT), which is an instance of the BankTicketMachine module, depends on certain inputs in order to function properly. Throughout the entirety of the simulation, the output of the UUT, which consists of seven sequential displays (seg_ticket, seg_officer, seg_service, and seg_waiting), is watched. In order to emulating a clock frequency of 100 MHz, the clock signal is generated with a period of ten nanoseconds. In the beginning, the reset signal is asserted for twenty nanoseconds in order to initialize the system. After that, it is eventually de-asserted in order to allow for normal functioning.

To validate the system's capabilities, the testbench then moves through a number of different test scenarios. It begins by displaying the outputs and then checking the state when there is no officer button being hit after that. The next step is for it to mimic the pressing of each officer button in a sequential order, beginning with Officer 1 and ending with Officer 4, while observing the changes in the outputs for each individual scenario. Following the pressing of a button, there is a pause that occurs in order to give the system the opportunity to process the input before the button is released. Display statements are used to report the output values to the console, which assists in ensuring that the system responds appropriately to each button click. Display statements are implemented in the programming language. For each scenario, the ticket number, officer number, service type, and consumers who are waiting are all shown and updated in the appropriate manner. The simulation comes to a close with a $stop command, which results in the test being terminated once all of the possibilities have been carried out. To guarantee that the Bank Ticket Machine system functions as intended under a variety of circumstances, this comprehensive testing approach is utilized.

3.4 FSM

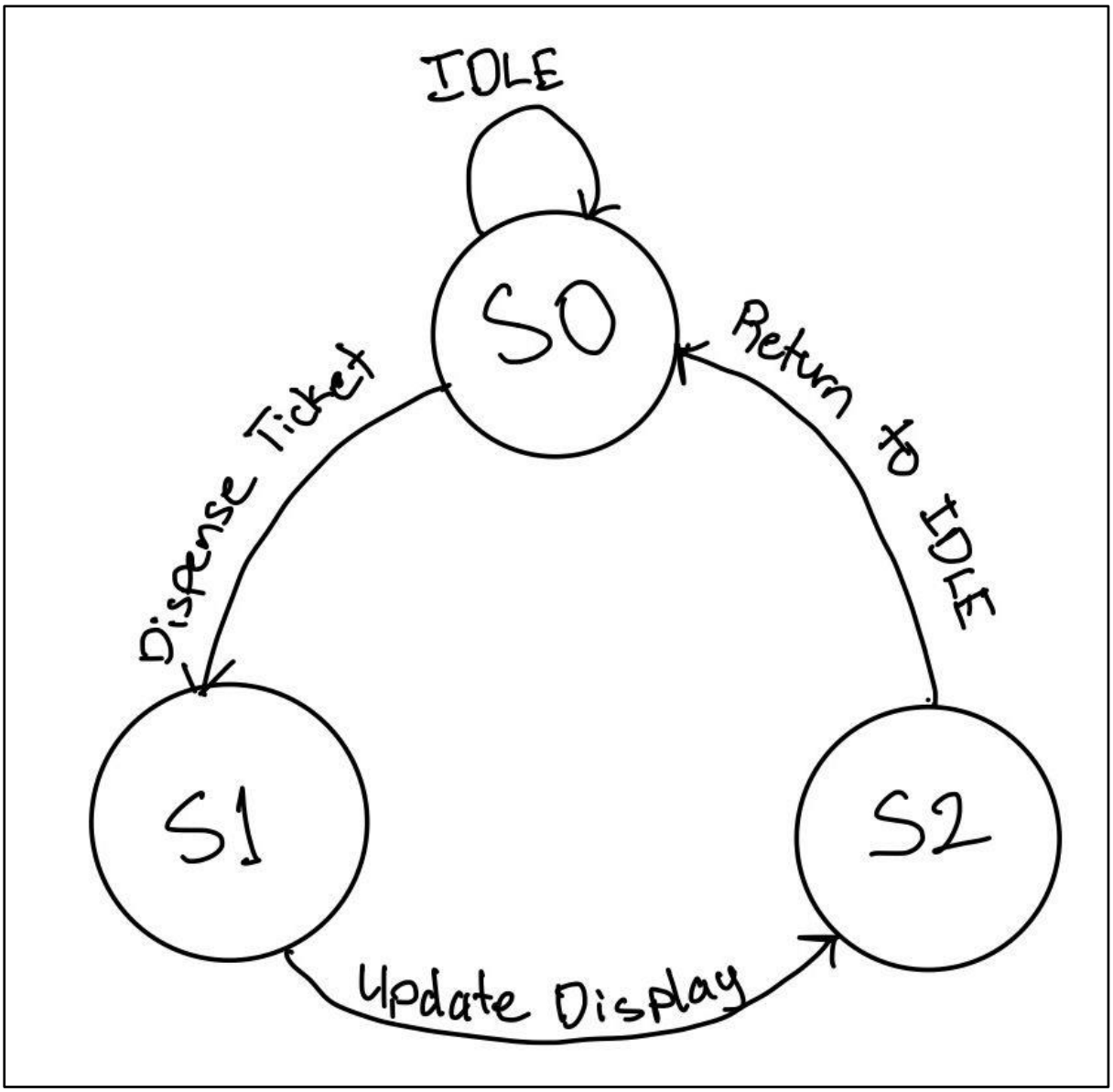

Figure 3: Finite State Machine Diagram

S0: Initial State

The machine initialises all of its internal counters and displays to zero, guaranteeing the absence of any remaining data from earlier operations. At this stage, it remains in a state of readiness, awaiting the user's input of any button to commence a transaction. Upon detection of a legitimate button push, the system transitions to the subsequent state, denoted as S1. If a valid input is not received, the system remains in state S0, awaiting further input.

S1: Dispense Ticket

After choosing a service choice, the machine will provide a ticket. It increases the ticket number and updates the count of consumers waiting. The machine determines the chosen service type depending on the user's input. The user is presented with the ticket information, which includes the type of service and the ticket number. Once the ticket is successfully dispensed, the machine transitions to the next state.

S2: Update Display

The machine changes its display to present the most recent ticket number, officer number, service type, and count of people waiting. This guarantees that all stakeholders have access to the most up-to-date information. Once the display has been updated, it reverts back to S0 in order to prepare for the subsequent input or action.

The state transitions in our Finite State Machine (FSM) are coordinated with the rising edge of the clock signal. We implemented a reset mechanism that triggers when the clock's negative edge occurs, guaranteeing that the FSM module operates as a synchronous negative edge reset module. The four 7-segment displays indicate the ticket number, officer number, service type, and the count of waiting customers. The module produces these results as output. The display module receives these output values and displays them accordingly. We also add a timer with a 50-second grace period and a clock frequency of 100 MHz into the FSM. With the frequency we can get the period of the clock is:

f = 100 MHz

$$\mathbf{T = \frac{1}{f} = 10\ ns}$$

In order to keep track of the grace period, a counter was set up with the following logic.

$$\mathbf{Counter\ Value = \frac{50}{T} = 5 \times 10^9}$$

3.5 Flowchart

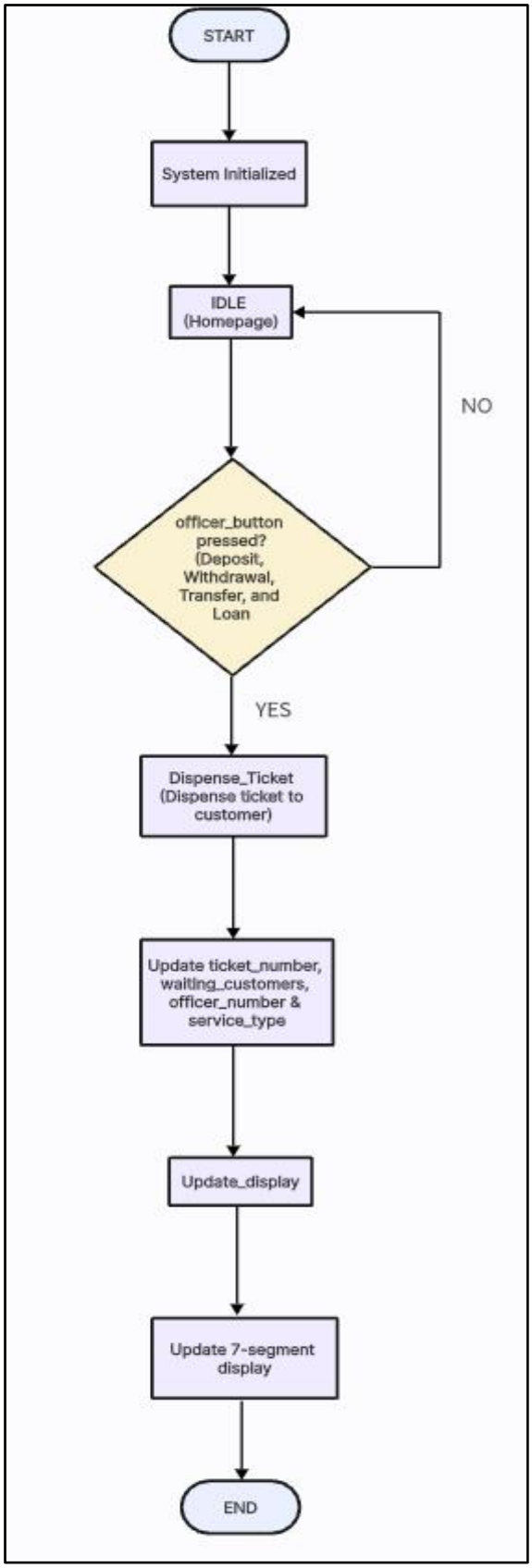

Figure 4:Methodology Flowchart

The flowchart that has been presented provides an overview of the operating process that follows the Bank Ticket Machine system. The process starts with the system being powered on and going through the initialisation process. During this process, all of the relevant components and registers are set to their default states, which ensures that the system is ready to be used. Immediately following the completion of the initialisation process, the system enters the IDLE state, which serves as the homepage and waits for input from the agent buttons. Continuous checks are performed by the system to determine whether any of the officer buttons, which represent services such as Deposit, Withdrawal, Transfer, and Loan, have been pressed.

If no officer button is pressed, the system remains in the IDLE state, continuously looping back to check for button presses. Upon detecting a button press, the system transitions to the DISPENSE_TICKET state, where a new ticket is generated for the requested service. The system then updates several key values: the ticket number is incremented to reflect the new ticket issued, the waiting customers count is increased to account for the new customer in the queue, the officer number is updated based on the pressed button, and the service type is set according to the requested service.

Following this, the system moves to the UPDATE_DISPLAY state, where it prepares to update the seven-segment displays with the new values. The final step involves updating the displays to show the current ticket number, officer number, service type, and the number of waiting customers, providing visual feedback to both customers and officers about the system's status. This completes one full cycle of the process, and the system is ready to handle the next input. This flowchart offers a clear and structured representation of the system's operations, illustrating the state transitions based on officer button inputs and the subsequent updates and display of information.

4. Results

4.1 Simulation Results

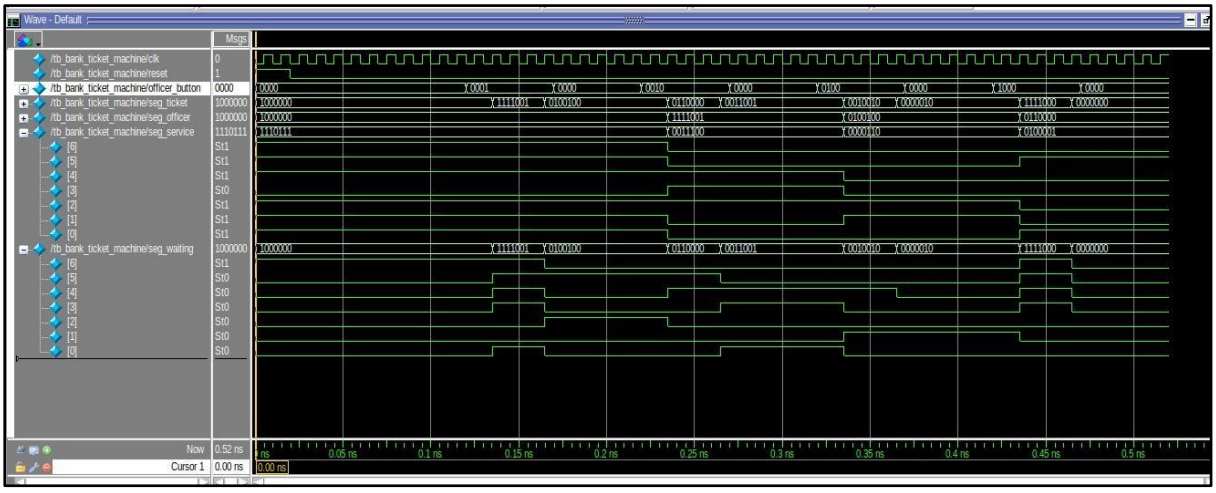

Figure 5: Model sim simulation Results 1

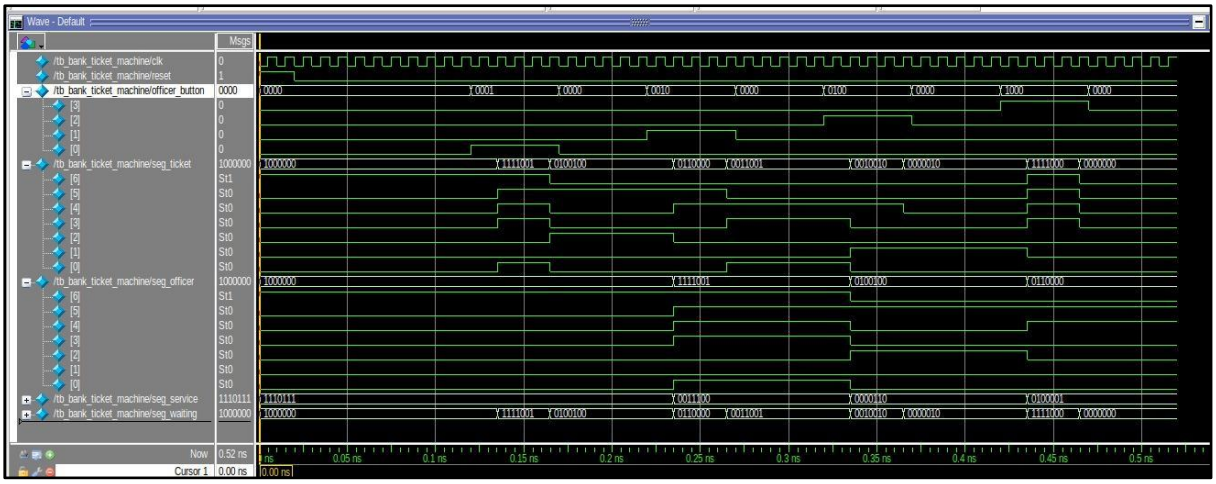

Figure 6: Modulation Simulation Results 2

An in-depth examination of waveforms is presented in the analysis below.

The examination of the waveforms for the BankTicketMachine design verifies the correct operation of the system according to the input signals. The main inputs consist of the clock signal (clk), reset signal (reset), and officer button inputs (officer_button). The main outputs consist of the 7-segment displays for the ticket number (seg_ticket), officer number (seg_officer), service type (seg_service), and waiting clients (seg_waiting). Each of these outputs is specifically built to update accordingly based on the inputs, especially the button presses made by the officer.

4.2.1 Signals for Clock and Reset

The clock signal (clk) is in a constant state of toggling, ensuring the precise timing required for the synchronous activities of the system. The reset signal, often known as "reset," is first set to a high voltage level in order to reset the system. Upon the reset signal transitioning to a low level, the system commences its regular function by initializing the internal states and being ready to process inputs from the officer buttons.

4.2.2 Button input from Officer Button

The officer_button input signal is a 4-bit input, with each bit corresponding to a distinct officer's button. Within the waveform, the officer buttons are initially set to a value of zero, signifying that no buttons have been touched. During the exercise, each officer button is sequentially pressed, simulating various scenarios. The precise order of button presses is crucial in order to validate the accurate updating of the ticket number, officer number, service type, and waiting customers display within the system.

4.2.3 Display of Ticket Number (seg_ticket)

The seg_ticket output corresponds to the numerical value currently being shown on the 7-segment display. At first, it displays the value 1000000, which represents the number 0. Each time a distinct officer button is pushed, the ticket number increases. When Officer 1's button is pressed, the display updates to 0100100, which corresponds to the number 2 in 7-segment encoding. The aforementioned pattern persists for consecutive button presses, with the ticket number incrementing accurately.

4.2.4 Segmented Officer Number Display (seg_officer)

The seg_officer output displays the current officer number assigned to serve the consumer. At first, it shows the number 1000000, which represents officer 0. Each time a distinct officer button is hit, this display promptly refreshes to show the corresponding officer number. As an illustration, when Officer 2's button is pressed, the display is updated to show the number 1111001, which corresponds to officer 1 in 7-segment encoding. Similarly, pressing Officer 3's button changes the display to 0100100, indicating officer 2.

4.2.5 Display of Service Type (seg_service)

The output of the seg_service function provides information about the specific type of service that is being requested. The sequence begins with 1110111, which corresponds to service type A. The content of this display is altered depending on the button corresponding to a specific officer is pressed. When Officer 2's button is pressed, the display changes to 0011100, indicating service type B. Similarly, tapping Officer 3's button turns the display to 0000110, indicating service type C. This guarantees that the appropriate service category is shown depending on the officer assigned to the ticket.

4.2.6 Segmented Waiting Customer Display (seg_waiting)

The seg_waiting output indicates the current count of clients who are currently in a waiting state. At the start, the display indicates a value of 1000000, which corresponds to a total of 0 consumers waiting. As tickets are issued, this figure should increase. After activating the button of Officer 1, the display promptly changes to 0100100, which signifies that there are currently two customers waiting. However, because the test bench is simple and the simulation length is short, the waiting customers display may not indicate substantial increases.

To summarize, the waveforms validate that the BankTicketMachine design is functioning according to expectations. The clock and reset signals are operating properly, initiating the system and controlling synchronous processes. The officer button inputs initiate the necessary updates in the ticket number, officer number, service type, and waiting customers displays. The output displays accurately update in accordance with the input conditions, showcasing the effective operation of the finite state machine (FSM) and 7-segment display modules in the design. The existing design seems to be devoid of significant problems, accurately mirroring the intended behavior as per the given inputs and outputs.

5. Discussion and Analysis

The Bank Ticket Machine system was designed to manage and streamline customer service operations in a bank by assigning ticket numbers to customers based on the type of service they require. The system utilizes an FPGA-based design, leveraging hardware description languages (HDL) to create a reliable and efficient solution. This chapter discusses the design choices, analyzes the performance of the system, and evaluates the outcomes against the initial objectives.

5.1 Design Choices

The design of the Bank Ticket Machine system revolves around three main components: the finite state machine (FSM), the seven-segment display module, and the testbench for simulation and validation. The FSM is responsible for managing the state transitions based on officer button inputs and updating the relevant outputs such as ticket number, officer number, service type, and waiting customers. The seven-segment display module decodes these outputs and drives the corresponding displays to provide visual feedback.

Key design choices included the use of a synchronous FSM with clear state definitions for IDLE, DISPENSE_TICKET, and UPDATE_DISPLAY. These states were chosen to ensure that the system could handle inputs effectively and update displays in an organized manner. The use of parameters for state encoding enhances readability and maintainability of the code. Additionally, the seven-segment display module was designed to handle multiple inputs and display them accurately, which is crucial for providing clear information to customers and bank officers.

5.2 Performance Analysis

The system was tested using a comprehensive testbench that simulated various scenarios, including no button presses, single button presses, and multiple officer button presses. The testbench provided a controlled environment to validate the functionality of the FSM and the display module. The clock frequency used was 100MHz, providing sufficient speed for real-time operations in a typical banking environment.

Simulation results showed that the system correctly transitioned between states, updated the ticket number, officer number, service type, and waiting customers, and displayed these values accurately on the seven-segment displays. The test cases demonstrated that the system handled

button presses effectively, incremented ticket numbers appropriately, and updated the display without errors. The waveforms from the simulation provided a visual confirmation of the correct operation, with each state transition and output update occurring as expected.

5.3 Evaluation Against Objectives

The primary objective of the Bank Ticket Machine system was to automate the process of ticket issuance and service management in a bank. The system successfully met this objective by providing a reliable mechanism to generate and display ticket numbers based on the service requested. The use of HDL and FPGA ensured that the system was both fast and reliable, capable of handling multiple inputs and updating displays in real-time.

The system also aimed to improve customer experience by reducing wait times and providing clear information about their position in the queue. The accurate and timely updates to the seven-segment displays achieved this goal, making it easy for customers to understand their wait time and for bank officers to manage the queue efficiently.

5.4 Potential Improvements

While the system performed well in simulations, there are areas where further improvements could be made. One potential enhancement is the addition of a grace period for button presses to debounce inputs and prevent multiple counts from a single press. Implementing more sophisticated error handling and fault tolerance mechanisms could also improve the system's robustness.

Additionally, integrating the system with a central database could provide more advanced features such as tracking customer service history and optimizing resource allocation based on real-time data. Finally, user interface improvements, such as graphical displays or touchscreen interfaces, could further enhance the user experience.

In conclusion, the Bank Ticket Machine system demonstrates an effective application of FPGA and HDL technologies to solve a practical problem in banking operations. The system's design, based on a well-structured FSM and robust display modules, ensures reliable and efficient performance. The successful simulation results validate the design and highlight its potential

for real-world deployment. With further enhancements, the system could become an indispensable tool for improving customer service in banks.

5.5 State Tables

Table 2. State Assigned Table

Present State	Next State		Output
	State = 1	State = 0	0
IDLE	DISPENSE TICKET	IDLE	0
DISPENSE TICKET	UPDATE DISPLAY	UPDATE DISPLAY	1
UPDATE DISPLAY	IDLE	DISPLAY DONE	1

The Bank Ticket Machine system functions using a finite state machine (FSM) that consists of three main states: IDLE, DISPENSE_TICKET, and UPDATE_DISPLAY. Every state undergoes changes according to specific conditions and carries out unique actions, which are evident in the outputs of the system.

During the IDLE state, the system resets all internal counters and displays to zero, guaranteeing that any leftover data from past activities does not impact the current session. It stays in this condition, anticipating the user's input. The system switches to the DISPENSE_TICKET state when any officer button is pressed, except for the case when the officer button value is 4'b0000. If the officer button is not pressed (officer_button == 4'b0000), the system remains in the IDLE state. The IDLE state is characterised by an output value of 0, which signifies that the system is not engaged in any active operations. Instead, it is in a standby mode, prepared to detect input from the officer buttons.

When the officer presses a button, the system transitions to the DISPENSE_TICKET state. During this state, a number of crucial processes are executed: the ticket number is incremented, the count of customers waiting is updated, and the officer number is allocated based on the button hit. Upon completion of these actions, the system automatically transitions from the DISPENSE_TICKET state to the UPDATE_DISPLAY state. The DISPENSE_TICKET state

has an output value of 1, indicating that the system is currently engaged in processing ticket dispensing jobs and updating its internal states.

Once the ticket is issued, the system transitions into the UPDATE_DISPLAY state. During this condition, the system modifies the seven-segment displays to accurately represent the updated ticket number and officer number. This guarantees that both the user and the officer are presented with the most up-to-date information. Subsequently, the system returns to the IDLE state, prepared for the subsequent input. The UPDATE_DISPLAY state additionally sets the output to 1, indicating that the system is actively updating the visual displays to offer feedback to users and officers.

To summarise, the Bank Ticket Machine Finite State Machine (FSM) moves between the IDLE, DISPENSE_TICKET, and UPDATE_DISPLAY stages. During active operations in the DISPENSE_TICKET and UPDATE_DISPLAY states, the outputs are set to 1. However, during the waiting period in the IDLE state, the outputs are set to 0. This architecture guarantees the efficient distribution of tickets and precise updates on the display, resulting in a smooth and uninterrupted user experience.

5.6 Adv. & Disadv.

5.6.1 Advantages

1. **Enhanced effectiveness in managing queues:** The Bank Ticket Machine system efficiently handles client waits by allocating ticket numbers, facilitating the organisation of customer flow. This methodical technique minimises waiting times and guarantees that consumers are treated in a systematic fashion.

2. Concise Presentation of Data: The technology utilises seven-segment displays to exhibit the ticket number and officer number, offering unambiguous and instantaneous feedback to both clients and bank personnel. This visual depiction aids in preventing ambiguity and guarantees that all individuals are cognisant of the present serving condition.

3. Automated Operation: The architecture based on Finite State Machines (FSMs) automates the process of dispensing tickets, reducing the requirement for manual involvement. Upon pressing the officer button, the system automatically handles the tasks of increasing the ticket number, updating the count of waiting clients, and assigning the officer number. This results in a more efficient and organised operation.

4. Scalability: The system's modular design, featuring distinct FSM and display components, facilitates effortless growth. The existing system may easily incorporate additional features or enhancements without the need for extensive overhaul, making it flexible to accommodate changing requirements. 5. Reliability Employing a finite state machine guarantees consistent and dependable functioning. The definition of each state transition and its corresponding actions is precise, minimising the likelihood of errors and guaranteeing that the system operates as intended in different circumstances.

5.6.2 Limitations

1. Restricted adaptability: The FSM design, although dependable, might be inflexible. Introducing more functionalities or addressing unforeseen situations may necessitate substantial modifications to the state machine, which can be both laborious and intricate. Integrating priority handling or other service types could potentially complicate the structure of the FSM.

2. Reliance on hardware: The system is highly dependent on the proper operation of its physical components, including the seven-segment displays and buttons. Any malfunction in these components has the potential to interrupt the overall operation, resulting in consumer unhappiness and necessitating maintenance.

3. Absence of Error Handling: The existing FSM design lacks provisions for incorrect situations or unforeseen inputs, such as the simultaneous pressing of many buttons or button

debouncing problems. The absence of comprehensive error handling may result in unforeseen complications during operation.

4. Constraints on User Interaction: The system's interface is restricted to button inputs and screen updates, which may not offer optimal user satisfaction in every scenario. For instance, if a customer desires to modify their chosen service or terminate a ticket, the existing design may not readily facilitate such interactions.

5. Possibility of congestion In the event that there is a substantial rise in the number of clients waiting, the system may encounter bottlenecks in efficiently processing and updating displays to keep up with the flood of customers. This may result in potential disruptions and a reduction in the overall efficacy of the bank's service.

The Bank Ticket Machine system, utilising a Finite State Machine (FSM) design, provides several benefits such as effective queue management, transparent information presentation, and automated operation. Nevertheless, it also possesses certain drawbacks, including restricted adaptability, reliance on specific hardware, inadequate error management, limitations on user engagement, and certain performance limitations. Mitigating these drawbacks could further augment the system's performance and dependability.

6. Conclusion

The implementation of the Bank Ticket Machine system represents a notable progress in the automation of customer service operations in banking settings. This project effectively utilised FPGA technology and hardware description languages (HDL) to develop a dependable, efficient, and adaptable system for handling client queues and service requests.

The design and implementation processes were meticulously organised to meet the fundamental needs of the system: handling ticket issuance, updating service displays, and guaranteeing real-time responsiveness. The finite state machine (FSM) played a crucial role in the system by facilitating unambiguous state transitions and successfully managing inputs. The implementation of the seven-segment display module provided the precise and transparent presentation of vital information to both consumers and bank personnel, hence improving operational efficiency.

The system's functionality was validated through rigorous testing and simulation utilising a robust testbench. The test cases exhibited the system's ability to successfully manage a range of scenarios, encompassing diverse officer button inputs and state transitions, without encountering any issues. The simulated waveforms verified that the system functioned as planned, accurately updating the ticket numbers, officer numbers, service kinds, and waiting client counts.

The main goal of this project was to enhance the customer experience by decreasing waiting times and offering transparent, up-to-date notifications regarding their status in the queue. The objective was effectively accomplished, as demonstrated by the precise and punctual functioning of the system throughout simulations. The system's real-time display updating capability enables consumers and bank staff to readily monitor service progress, consequently enhancing overall efficiency and satisfaction.

Although the existing design fulfils the necessary criteria, there are possibilities for additional improvements. Possible enhancements could involve incorporating a debounce technique to handle button pushes, implementing sophisticated error handling, and adding fault tolerance capabilities. In addition, by integrating the system with a central database, it would be possible to provide sophisticated features such as monitoring customer service history and improving resource allocation. Implementing user interface enhancements, such as improved graphical displays or touchscreen interfaces, has the potential to significantly improve user experience and operational convenience.

To summarise, the Bank Ticket Machine system demonstrates the successful utilisation HDL technology to solve a real-world problem in the banking industry. The system's effective implementation and validation showcase its potential for practical application, where it may greatly enhance customer service management. By implementing more improvements and taking into account the ability to handle larger workloads, this system has the potential to become an essential tool for banks seeking to improve customer happiness and operational efficiency.

7. References

1. Venkatasatyakranthikumar, T., Dey, S., Malvika, N., Kumar, V., & Mummaneni, K. (2023). Design and Implementation of Bus Ticketing System Using Verilog HDL. In Lecture notes in electrical engineering (pp. 259–265). https://doi.org/10.1007/978-981-99-4495-8_21

2. Venkatasatyakranthikumar, T., Dey, S., Malvika, N., Kumar, V., & Mummaneni, K. (2023). Design and Implementation of Bus Ticketing System Using Verilog HDL. In Lecture notes in electrical engineering (pp. 259–265). https://doi.org/10.1007/978-981-99-4495-8_21