Lebanese American University

COE322 – Logic Design Lab

Final project – Logic- Controlled Board

Ali Skeineh 202204451

Jana Monzer 202301360

Raneem Reda 202302524

13/05/2025

Table of Contents

Table of Figures 3
Table of Tables 4
Abstract 5
Introduction 6
Equipment Used 7
Analysis 8
A. Overview of the Design 8
B. Breakdown of the Design 9
Stage 1: Boot State 9
Stage 2: Locked State (1-1 Mapping) 11
Stage 2.1: Sequence Detector State 12
Stage 3: Number of ON Switches Counter 14
Stage 4: Mapping (LEDs Multiplexers) 19
C. 555 Timer 23
D. Capless trick (Sequence 3) 24
Power Consumption Analysis 25
Financial Study 26
Problems Faced During the Project 27
Key design points that present advantages over alternative designs 28
Conclusion 29
References 30

Table of Figures

Figure 0-1- General map of the design 8
Figure 0-2- Logic circuit of the boot state 9
Figure 0-3- Simulation of the boot state 10
Figure 0-4- Logic circuit of the locked state 11
Figure 0-5- Logic circuit of the sequence detector 13
Figure 0-6- Simulation of the sequence detector 13
Figure 0-7- k- map for C1 15
Figure 0-8- k- map for C0 15
Figure 0-9- k- map for i1 16
Figure 0-10- Logic circuit of the counter 17
Figure 0-11- Simulation of the counter circuit 17
Figure 0-12- example from the counter simulation 17
Figure 0-13- Delay case 1 18
Figure 0-14- Delay case 2 18
Figure 0-15- Logic circuit of mapping 20
Figure 0-16- Simulation of mapping 20
Figure 0-17- Example from mapping simulation 21
Figure 0-18- 555 timer chip 23

Table of Tables

Table 0-1- State table of the first stage 9
Table 0-2- priority encoder truth table 12
Table 0-3- Truth table of the counter 14
Table 0-4- Truth coding of encoding the counter into 2 bits 16
Table 0-5- Truth table for mapping 19

Abstract

This project involves the development of a logic-controlled interactive board, utilizing digital logic principles and a finite state machine (FSM) for its operation. The system includes four switches and four corresponding colored LEDs, with their behavior dynamically determined by the most recently deactivated switch. A unique feature is introduced in Sequence 3, where a cap removal trick enables a deceptive “capless” mode, adding an element of user misdirection. The entire design and sub-circuits were tested through simulation using Altera Quartus II. Overall, the project showcases key concepts in sequential logic, I/O management, and timing-based state transitions.

Introduction

The Logic-Controlled Board is a user-interactive circuit composed of four toggle switches and four LEDs in red, green, blue, and yellow. The association between each switch and LED shifts across four sequences, depending on which switch was most recently turned off. This report explores the system's conceptual design, manual FSM planning and online simulation on Altera Quartus II.

Equipment Used

- Different integrated chips: 74LS08 (AND), 74LS32 (OR), 74LS86 (XOR) and 74LS04 (NOT), 74153 (4:1 multiplexers).
- Flip-flops: 74LS74 (D flip- flops).
- 2 resistors (1 kΩ and 10 kΩ).
- 2 capacitors (68 μF and 68 nF).
- Breadboard
- Switches/ LEDs
- 555 Timer chip
- Software: Altera Quartus II.

Analysis

A. Overview of the Design

After gaining a clear understanding of the project's concept and functionality, we developed the following map (state diagram) to guide our design process and support deeper analysis throughout the implementation phase.

The key- points are:

- Boot is the initial stage of the design, proceeding to the next stage is determined by the behavior of switch 2.
- When switch 2 is ON, our next stage is the locked state which is a simple 1-1 mapping. Otherwise, the next stage is the sequence detector which determines the behavior of the circuit depending on the sequence we are in.
- Our whole design is controlled by a clock, specifically the 555 timer (treset). This 555 timer has a period of 4s, meaning that its high state lasts for 4 continuous pulses. Once this timer switches to low state, the system resets.

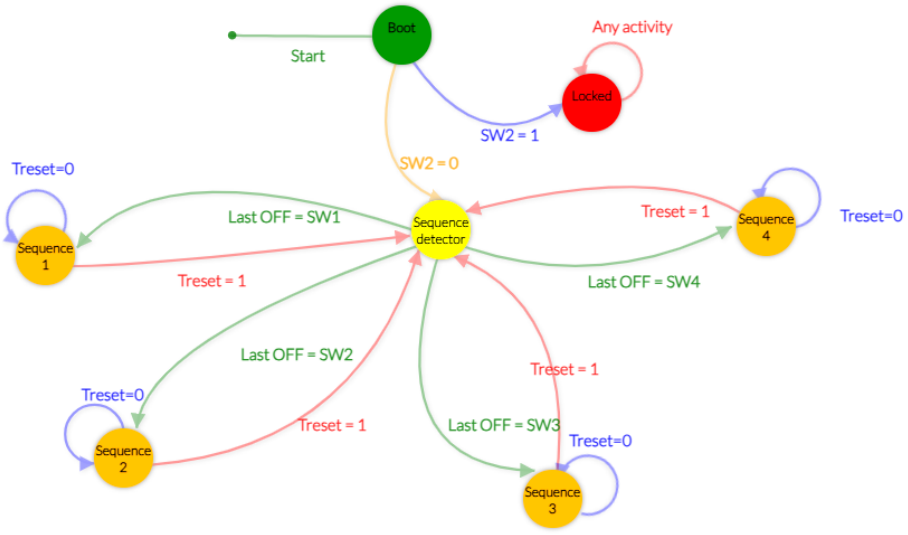


Figure 0-1- General map of the design

B. Breakdown of the Design

Stage 1: Boot State

Upon powering on the system, the project enters the boot state, represented by the binary code 000. In this initial state, the value of switch 2 determines the next state transition. If switch 2 is ON (1), the system transitions to the locked state (001). If switch 2 is OFF (0), it transitions to the sequence detector state (010).

To design this behavior, we constructed a state table that maps the current state (Boot) and the input (Switch 2) to the next state.

Table 0-1- State table of the first stage

Current state			Input	Next state		
S2	S1	S0	Switch 2	S2+	S1+	S0+
0	0	0	0	0	1	0
0	0	0	1	0	0	1

Based on this table, we derived the logic equations for the next state bits as follows:

S1+ = S1'S0'SW2'

S0+ = S1'S0'SW2

The resulting circuit was implemented using two D flip-flops in Quartus as provided below.

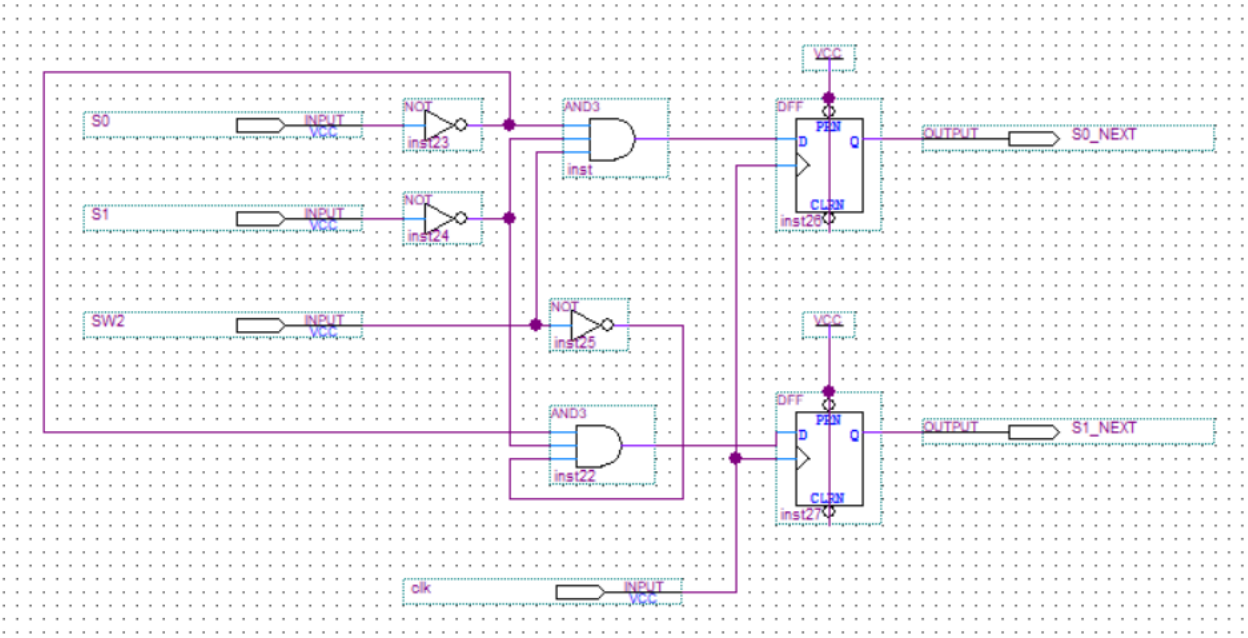


Figure 0-2- Logic circuit of the boot state

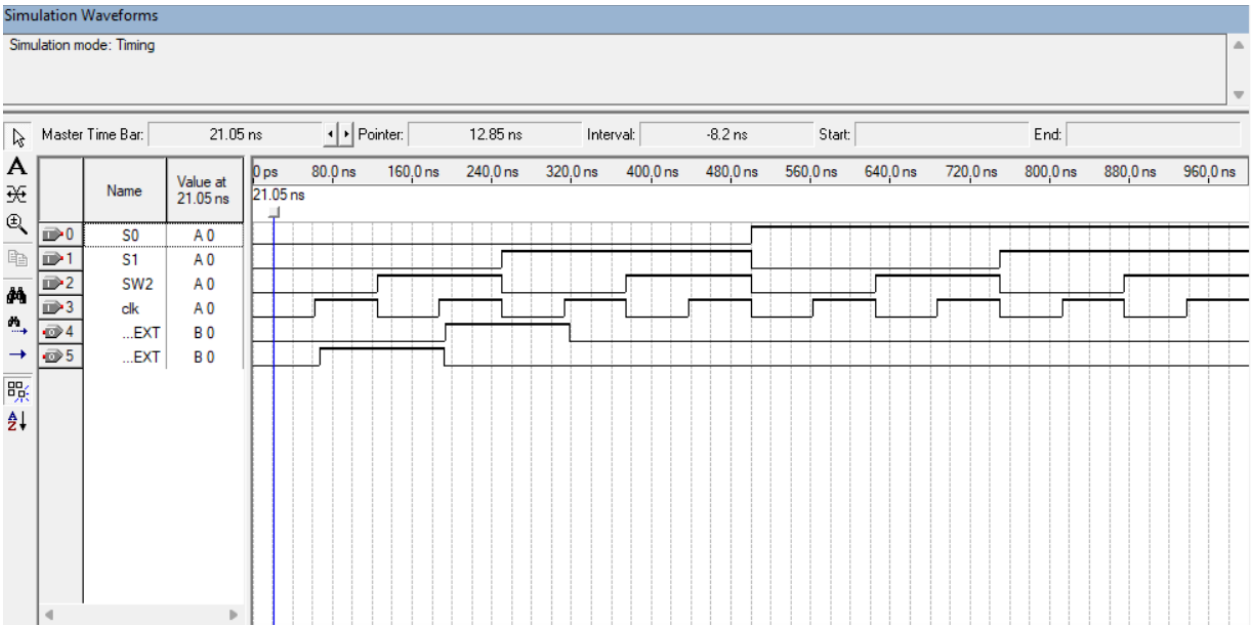


Figure 0-3- Simulation of the boot state

After analyzing the output waveforms, we confirmed that the circuit functions as expected. It is important to note that, for this part of the simulation, we focused exclusively on the condition where both S0 and S1 are 0, indicating that the system is initially in the Boot state.

Stage 2: Locked State (1-1 Mapping)

This state is reached only under the condition that switch 2 is ON. In this stage, each switch is directly connected to its corresponding LED (facing it).

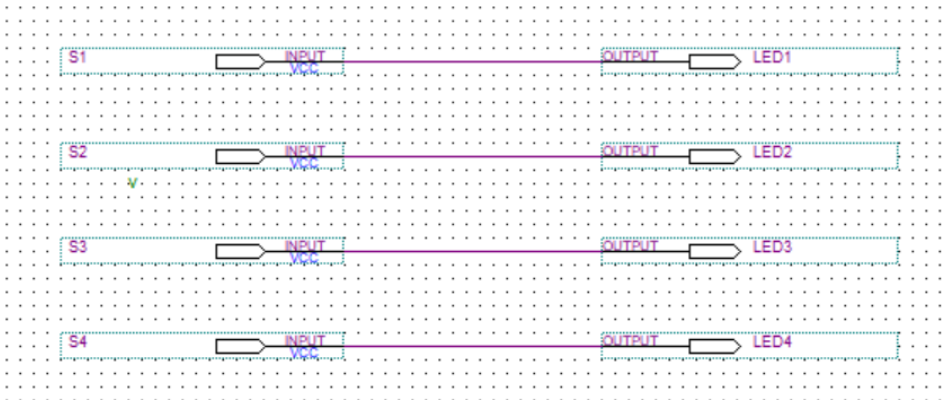


Figure 0-4- Logic circuit of the locked state

Stage 2.1: Sequence Detector State

This stage is reached only under the condition that switch 2 is OFF. In this stage, we need to detect which sequence we are in by identifying which switch was most recently turned off. This corresponds to detecting a falling edge on the switch signal when the switch transitions from HIGH (ON) to LOW (OFF). To achieve this, each switch input was connected to a D flip-flop to store its previous state. We then used an AND gate between the previous state (Q) and the inverted current state (¬D) to detect a falling edge. To ensure that only one falling edge is recognized at a time, we further AND gated each falling edge signal with a condition that all switches are currently off. The resulting signals were then inputted into a priority encoder, which encodes the active falling edge to determine the corresponding sequence number in 2 bits.

Now, since we did not use the built in chip of the priority encoder in quartus, we derived its truth table and then the equations.

Table 0-2- priority encoder truth table

F3	F2	F1	F0	Y1	Y0
0	0	0	0	X	X
1	0	0	0	1	1
0	1	0	0	1	0
0	0	1	0	0	1
0	0	0	1	0	0

We need to note that it is impossible to have a case where we have no falling edges, which is why we considered it do not care. Thus, the sequences are resembled as follows: 00 if F0=1 (switch 1 is last off), 01 if F1=1 (switch 2 if last off) and so on.

Now, using k- maps, we can derive the simplified equations of Y1 and Y0, which are:

Y1= F1'F0'

Y0=F2'F0'

The implementation of the sequence detector circuit on Quartus II is provided below.

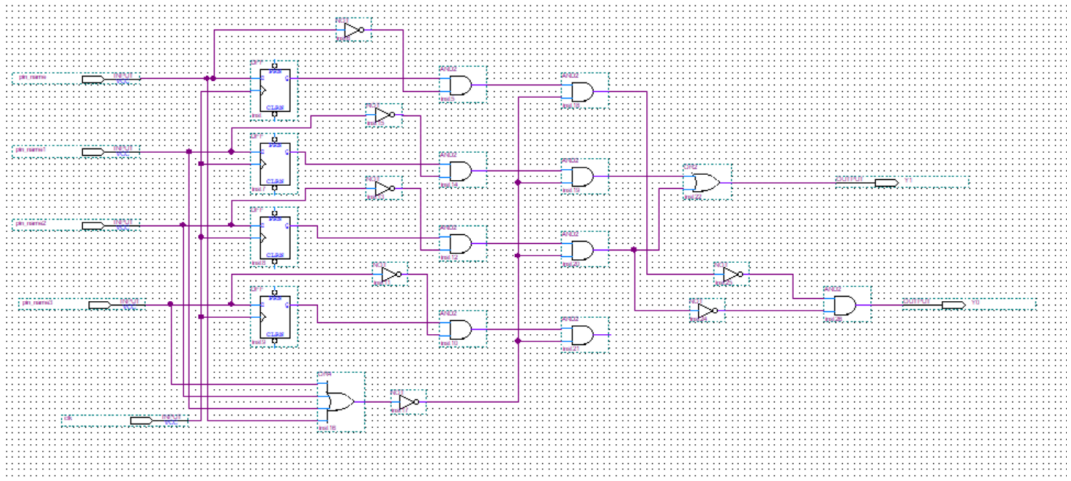


Figure 0-5- Logic circuit of the sequence detector

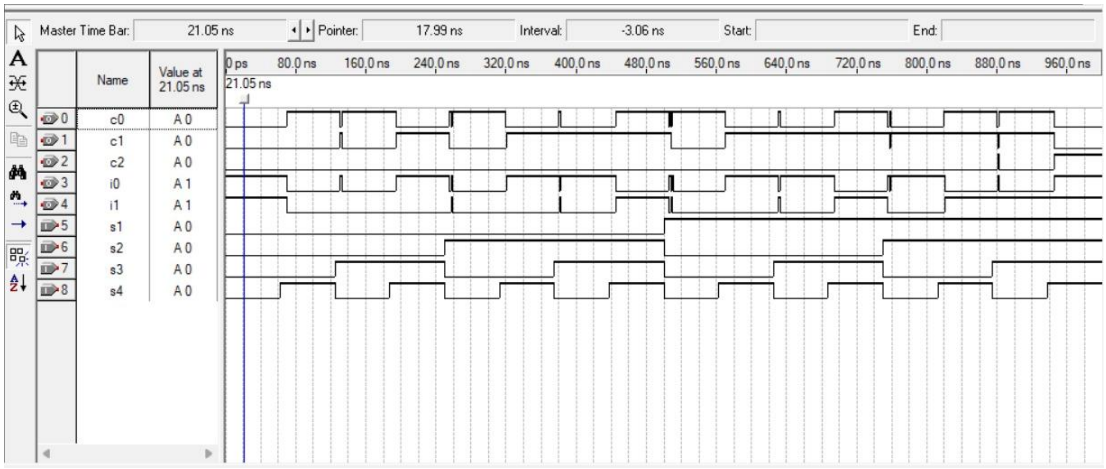


Figure 0-6- Simulation of the sequence detector

After observing the output waveforms, we notice that the output signals are inaccurate. Notably, the general concept of the sequence detector circuit and its functionality is there, but it is still missing sub-ideas we could not detect.

Stage 3: Number of ON Switches Counter

The purpose of this stage is to count the number of ON switches once the system starts operating, this will further help us in the implementation of the final stage.

This counter takes the 4 switches as inputs, and outputs the number of ON switches encoded in 3 bits. Considering all the cases, we get the following truth table.

Table 0-3- Truth table of the counter

S1	S2	S3	S4	C2	C1	C0
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	0
0	1	1	1	0	1	1
1	0	0	0	0	0	1
1	0	0	1	0	1	0
1	0	1	0	0	1	0
1	0	1	1	0	1	1
1	1	0	0	0	1	0
1	1	0	1	0	1	1
1	1	1	0	0	1	1
1	1	1	1	1	0	0

Now, we are ready to derive the equations of the outputs and simplify them using K-maps using K-map solver software.

- For C_2: there is no need for a K-map

  $C2 = S_1S_2S_3S_4$

- For C_1:

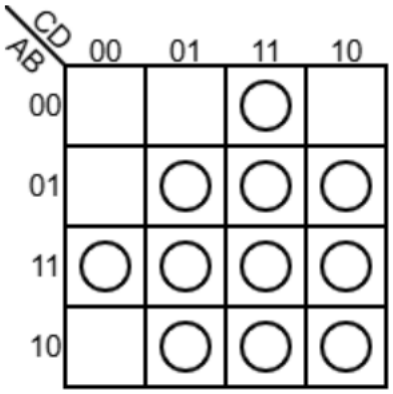


Figure 0-7- k- map for C1

Then, $C_1 = S_1'S_3S_4+S_1'S_2S_4+S_2S_3S_4'+S_1S_2'S_4+S_1S_2'S_3+S_1S_2S_3'$

- For C0:

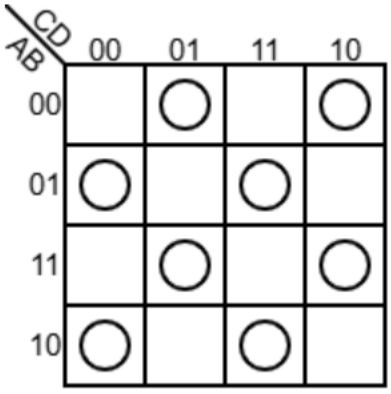


Figure 0-8- k- map for C0

Then, $C0 = S1 \oplus S2 \oplus S3 \oplus S4$

Now, in the above counter truth table, we considered the case that all switches are OFF, which is the first case in the truth table. However, in practice, this case does not exist since this stage comes after the sequence detector, where it is impossible to have none of the switches ON. So, we encode the 3-bit output of the counter into 2 bits according to the following truth table.

Table 0-4- Truth coding of encoding the counter into 2 bits

C2	C1	C0	i1	i0
0	0	0	X	X
0	0	1	0	0
0	1	0	0	1
0	1	1	1	0
1	0	0	1	1
1	0	1	X	X
1	1	1	X	X

The equations of i0 and i1 can be derived using k- maps.

- For i1:

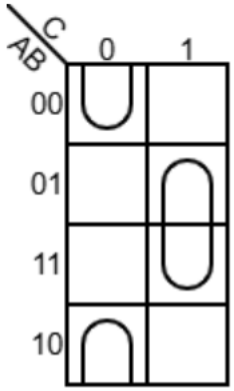


Figure 0-9- k- map for i1

Then, i1= (C1⊕C0)

Similarly, we get i0=C0’

Now, we are ready to build the counter circuit on Quartus and simulate it.

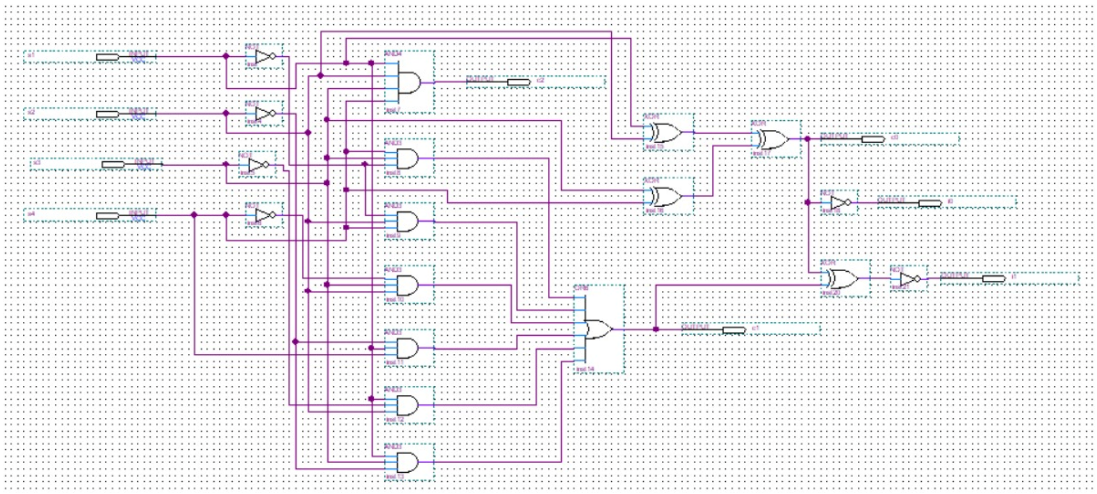


Figure 0-10- Logic circuit of the counter

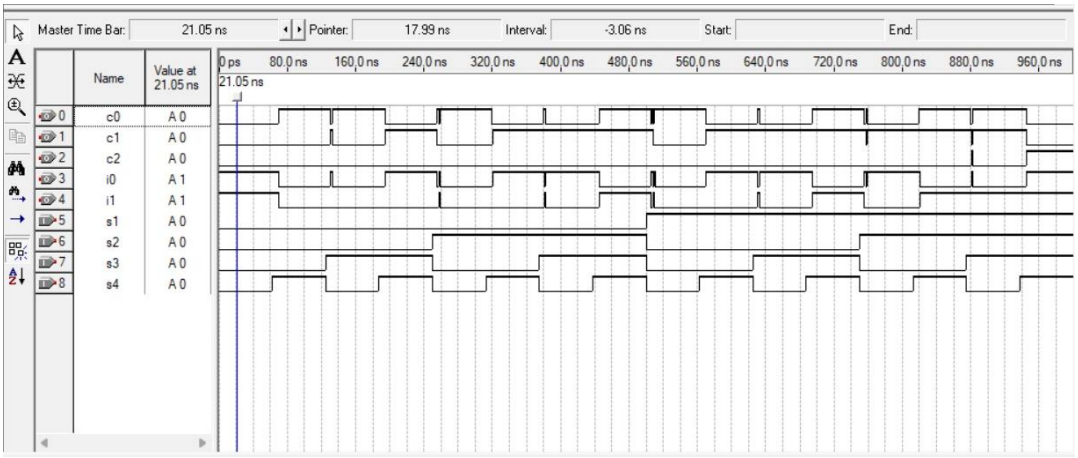


Figure 0-11- Simulation of the counter circuit

After observing our output waveforms, we made sure that our outputs are accurate. This is illustrated in the example below.

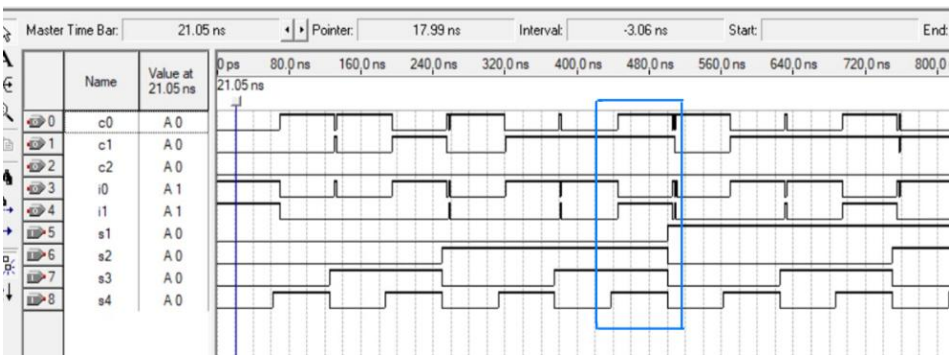


Figure 0-12- Example from the counter simulation

In the case of output marked below, when three switches are high (ON), we have i1=1 and i0=0, and knowing that i1 is the most significant bit and i0 is the least, this output represents 2 in decimal, which is correct and correctly defines the number of On switches after neglecting the case where all switches are OFF (in this case, when the counter is 0, 1 switch is ON, counter 1, two switches are ON and so on).

Now, to calculate the delay in the output, we will observe the output waveform for multiple delay cases and determine their average.

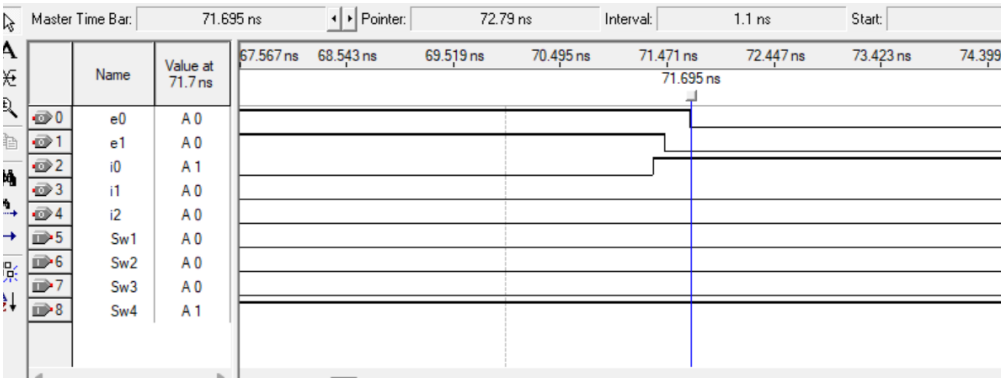


Figure 0-13- Delay case 1

In this case, delay = 71.695 - 71.36 = 0.335ns.

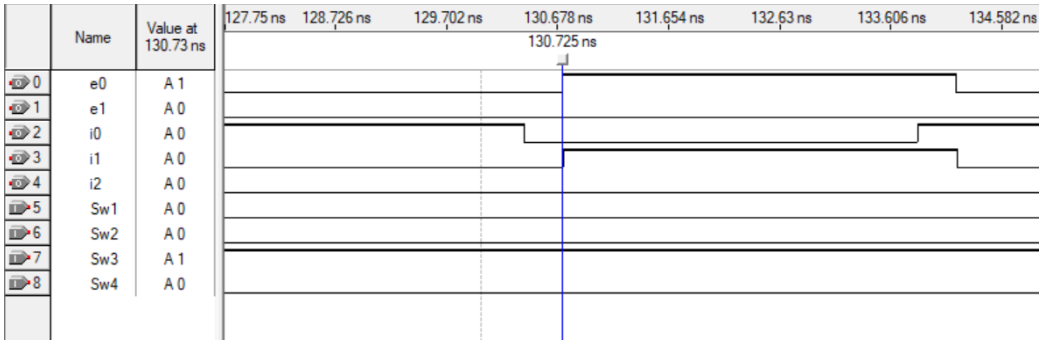


Figure 0-14- Delay case 2

In this case, delay = 130.725 - 130.39 = 0.335ns.

We notice that the delays in different stages of the output are equal, then the overall delay is 0.335 ns.

Stage 4: Mapping (LEDs Multiplexers)

In this stage, we implemented a mapping mechanism to control the behavior of the four LEDs based on two key factors: the current sequence detected and the number of active switches. The sequence detector provides a 2-bit output (Y1, Y0) that identifies the current sequence, while the switch counter, which was implemented earlier, outputs another 2-bit value (I1, I0) indicating how many switches are turned ON.

To determine the LED outputs, we constructed a truth table using the 4-bit input formed by combining the sequence (Y1Y0) and counter (I1I0) values.

Table 0-5- Truth table for mapping

Y1	Y0	I1	I0	L1	L2	L3	L4
0	0	0	0	1	0	0	0
0	0	0	1	1	1	0	0
0	0	1	0	1	1	1	0
0	0	1	1	1	1	1	1
0	1	0	0	0	1	0	0
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	1	1	1
1	0	0	0	0	0	1	0
1	0	0	1	0	0	1	1
1	0	1	0	1	0	1	1
1	0	1	1	1	1	1	1
1	1	0	0	0	0	0	1
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	1
1	1	1	1	1	1	1	1

Based on this truth table, we derived individual logic functions for each LED.

Seq1: L1=1; L2=I1+I0; L3=I1; L4=I1.I0

Seq2: L1=I1.I0; L2=1; L3=I1+I0; L4=I1

Seq3: L1=I1; L2=I1.I0; L3=1; L4=I1+I0

Seq4: L1=I1.I0; L2=I1; L3=I1+I0; L4=1

The final circuit was implemented using four 4:1 multiplexers (one for each LED). The select lines of each multiplexer are connected to the sequence bits (Y1, Y0), while the data inputs are defined by the logic functions derived from the truth table using the counter bits (I1, I0). This setup ensures that each LED displays the correct output based on the current sequence and switch configuration.

This circuit is implemented on Quartus as shown below.

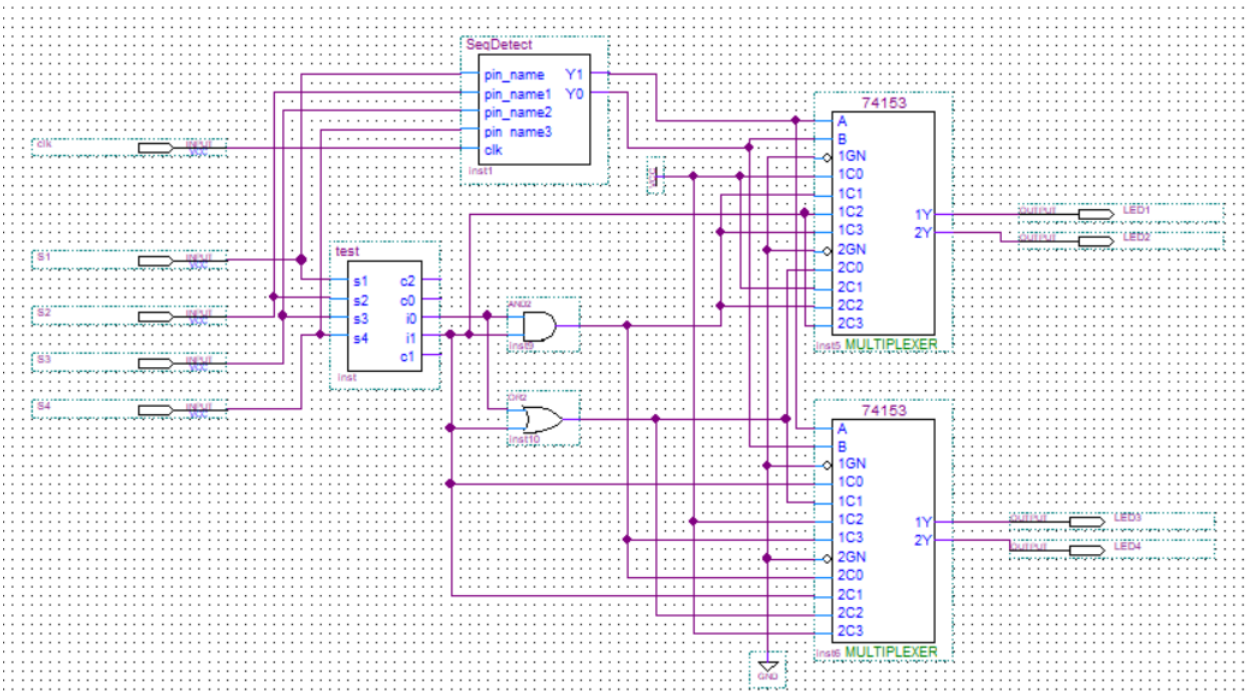


Figure 0-15- Logic circuit of mapping

This circuit combines the counter, sequence detector, and the mapping together.

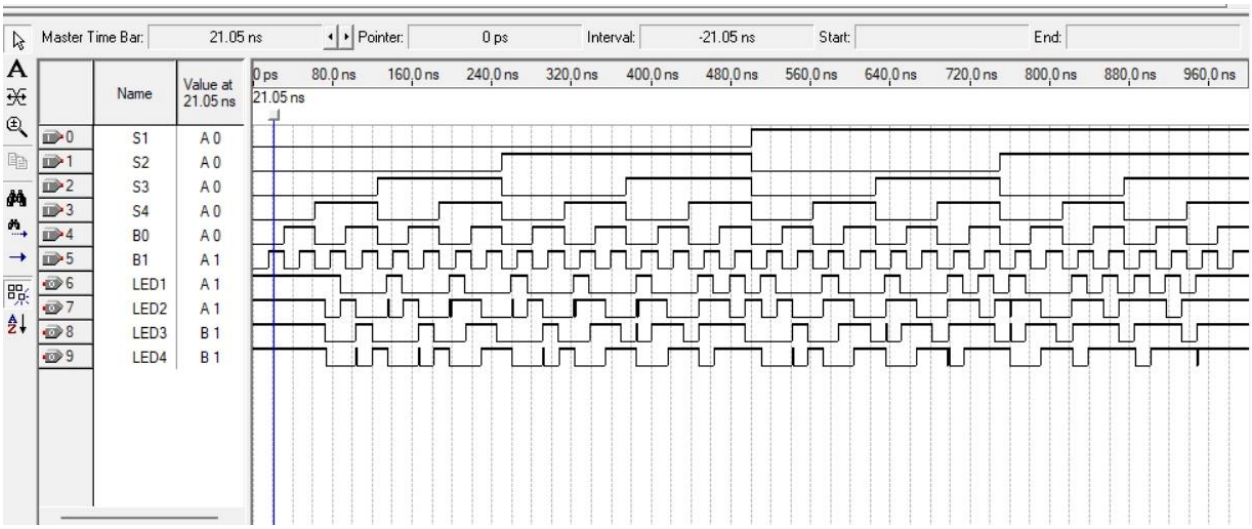


Figure 0-16- Simulation of mapping

After observing our output waveforms, we made sure that our outputs are accurate.

This is illustrated in the example below.

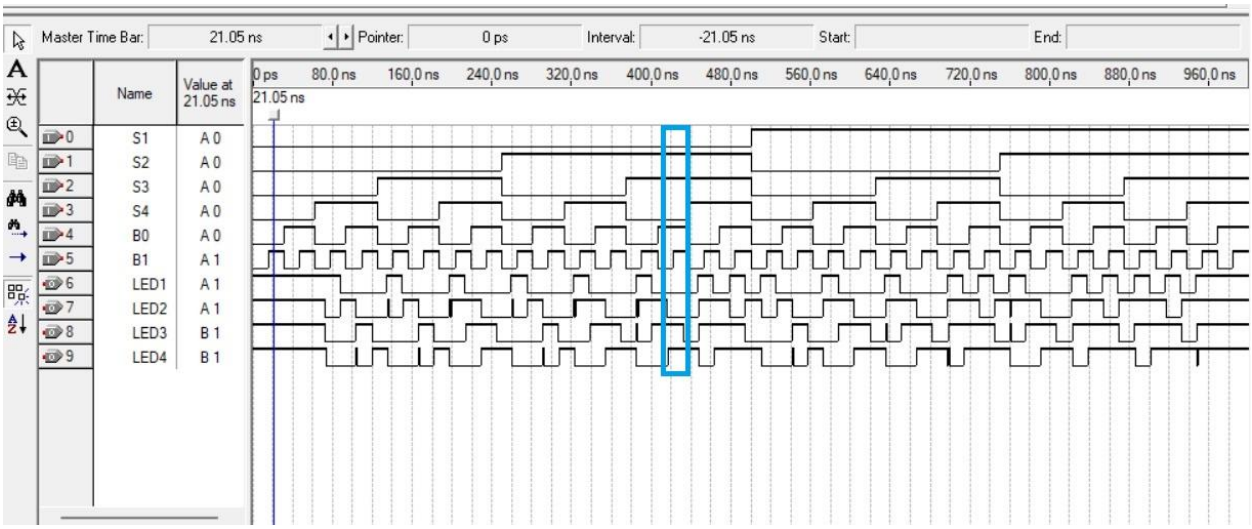
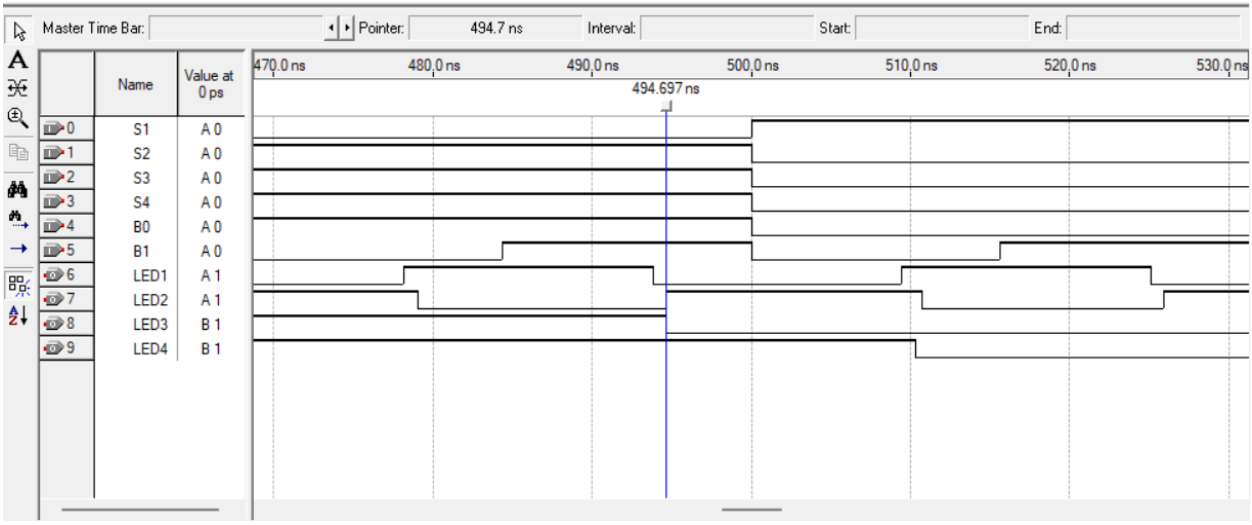


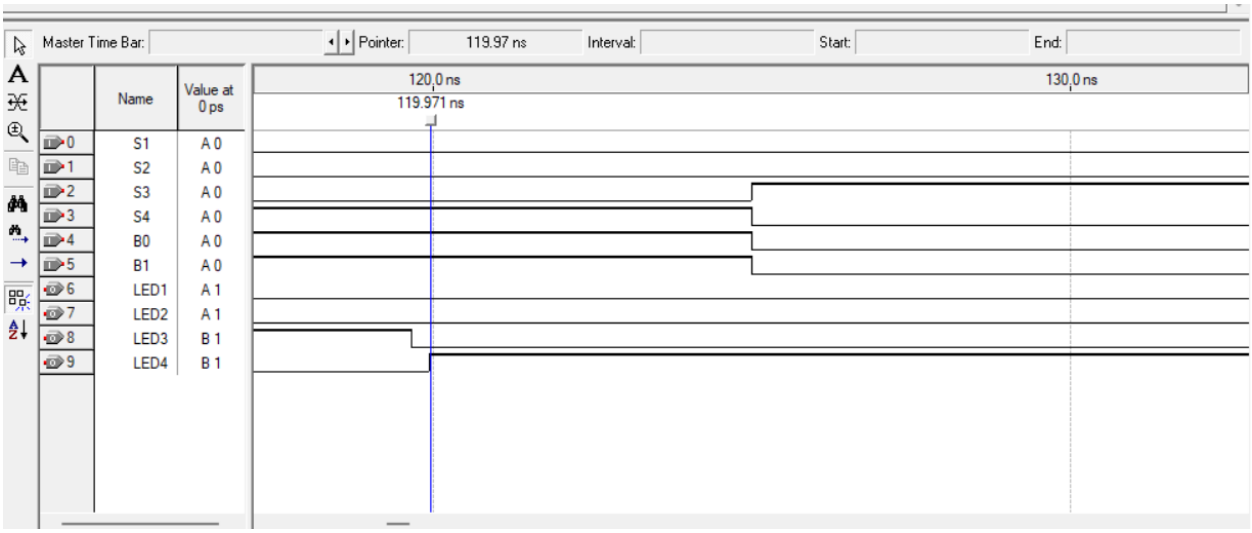
Figure 0-17- Example from mapping simulation

In the output case shown above, two switches are ON, and the sequence detector indicates sequence 4 (this corresponds to B1 = 1 and B0 = 1). As a result, LEDs 3 and 4 are ON, which aligns with our truth table and confirms the intended functionality of the mapping logic.

Now, to calculate the delay in the output, we will observe the output waveform for multiple delay cases and determine their average.



In this case, delay = 494.697 - 493.894 = 0.803ns.



In this case, delay = 119.971 - 119.687 = 0.284ns.

Thus, the average delay is 0.5435ns.

C. 555 Timer

The 555 timer is a versatile integrated circuit commonly used in three operating modes: astable, monostable, and bistable. In this project, we utilized the timer in both astable and monostable configurations to serve different purposes.

In the astable mode, the 555 timer was configured to generate a continuous square wave signal, effectively functioning as a clock. This clock was designed to produce a 4-second low pulse (transition from high to low) without requiring any external triggering. While such clocks are typically provided automatically in most systems, we constructed our clock manually to demonstrate its operation and behavior.

To achieve the desired timing, we selected specific component values based on the following formula for the timer's frequency:

$$f = \frac{1 \cdot 44}{(R_1 + 2R_2) \times C} \tag{1}$$

We are given that the period is 4s, so the frequency is 0.25 Hz.

By rearranging the equation, we get

(R1+2R2).C= 5.76

So, we choose the following values of the resistors:

R1= 10kΩ and R2= 15kΩ, this gives us that C= 14.4µF.

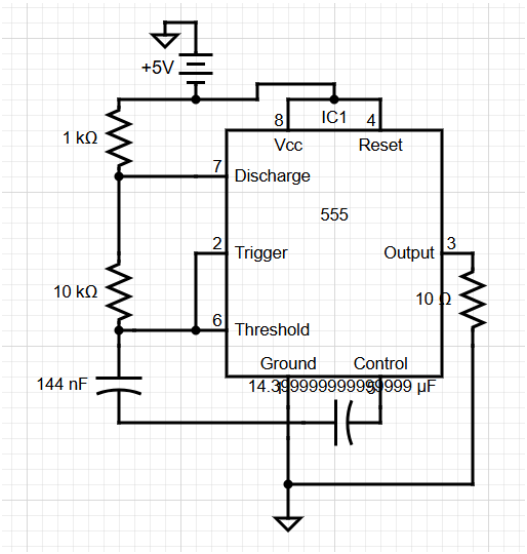


Figure 0-18- 555 timer chip

D. Capless trick (Sequence 3)

Special Trick #3 is activated under the condition that at least two LEDs are ON during Sequence 3. Once triggered, the system initiates a capless mode by turning OFF one of the active LEDs. At this point, the corresponding switch can be toggled ON and OFF freely without lighting up its LED (if the cap is removed).

However, this capless state is time-sensitive. A 4-second timer, implemented using our existing 555-timer-based reset circuit, begins as soon as the cap is removed. If the switch remains OFF for more than 4 seconds, the system automatically resets, and the switch becomes inactive. To reactivate it, the cap must be placed back on.

This behavior is managed using the previously implemented switch counter and is only applicable within Sequence 3. Entering this special sub-state requires detecting the correct combination of active switches and sequence value. The trick can be repeated across multiple switches, allowing users to perform the capless operation on different parts of the board if each follows the same timing and reset rules.

This feature adds a layer of interactivity and control, demonstrating conditional logic and timed behavior within a predefined state.

Power Consumption Analysis

All the power consumption values are determined from the datasheet of each chip. Assuming the power supply voltage is +5V, then the power consumption of the components used are as follows:

- **Logic Components (TTL estimates):**

AND/OR/XOR/NOT gates: 10–20 mW per gate.

D Flip-Flop: 20 mW per flip-flop.

4:1 Multiplexer: 20–25 mW.

- **Input Switches and Output LEDs:**

Switches: Passive (negligible power); minor current through pull-up/down resistors (∼0.5 mW)

LEDs: We have that the forward voltage is approximately 2V, then the Current is ∼20 mA. Thus, Power per LED: ∼40 mW.

- **Passive Components**

Resistors (used with timer): Power ≈ (5V - 2V) × 20 mA = ∼60 mW per resistor.

Capacitors: Negligible power unless switching rapidly

Financial Study

Unfortunately, in our case, we were not able to practically implement the design, so we did not get any equipment. However, we will provide an estimate for the financial cost of the project.

It is known that the price of each chip ranges between 0.5$- 1$, and ideally, this project requires no less than 25 ICs, which results in a minimum cost ICs of 12.5-25.

The circuit needs to fit on multiple breadboards, no less than 6 breadboards, and the price of 1 breadboard is 1$, so their cost is minimum 6$.

Suppose that the cost of the resistors and capacitors used for the 555 timer is a maximum of 3$, that of the jumper wires is 7$ since we need a lot of them, and that of the switches and LEDs is about 10$.

Finally, the voltage source costs up to 7$, and we might have purchased other equipment like a digital multi-meter which costs a minimum of 15$.

Thus, adding all these costs up, we get that the project costs a minimum of 65$, and the cost could vary depending on the number of ICs and equipment used.

Problems Faced During the Project

• The design was complex and detailed, involving numerous cases and conditions, making the understanding process challenging.

• Despite our efforts, we were unable to fully grasp and apply some of the key concepts.

• Implementing the reverse operation of the system was particularly difficult; while we managed to handle the forward operation, reversing the process proved much more challenging.

• On the hardware side, we anticipated that the project would be too large for practical implementation, which significantly increased the likelihood of errors and wiring mistakes.

Key design points that present advantages over alternative designs

• **Modular & Scalable Architecture**: The system is divided into clear stages (Boot, Locked State, Sequence Detector, Counter, and Mapping), allowing for easier debugging, testing, and future enhancements compared to monolithic designs.

• **Priority-Based Sequence Detection**: A custom priority encoder ensures unambiguous detection of the last switch turned off, eliminating conflicts that could arise in simpler edge-detection implementations.

• **Dynamic LED Control via Multiplexers**: The use of 4:1 multiplexers for LED mapping enables flexible and efficient switching between sequences, reducing hardware complexity compared to fixed-logic designs.

• **Optimized Logic Minimization**: Karnaugh maps were used to derive minimized Boolean equations for the counter and sequence detector, reducing gate count and improving circuit efficiency.

• **Reliable Timing with 555 Timer**: A manually configured 555 timer provides precise 4-second reset intervals, ensuring stable state transitions without relying on external clock sources.

• **Thorough Simulation & Validation:** Extensive Quartus II simulations verified each module's functionality before hardware implementation, minimizing practical errors.

• **Clear Documentation & Structured Analysis**: Detailed truth tables, state diagrams, and timing analyses ensure reproducibility and ease of understanding, setting this design apart from less-documented alternatives.

Conclusion

In conclusion, this project demonstrates the practical application of fundamental digital logic principles, both combinational and sequential, in the design of a logic-controlled board. By logically controlling the switches and their corresponding LEDs, we created dynamic and engaging behavior. This was primarily achieved through effective sequence detection based on the most recently turned-off switch, highlighting the power of precise edge detection and state-based logic in interactive hardware design.

References

Gibson, E. (2025, January 6). *How many LEDs can a 5V battery power? Explore LED wiring limits and calculations.* PoweringAutos. https://poweringautos.com/how-many-leds-can-a-5v-battery-power/