Lebanese American University

Logic Design Lab – COE 322 – Final Project – Logic-Controlled Board

Karim Sakr 202303146

Edward Mouawad 202300736

Charbel Chaaya 202303120

Rawad Saliba 202302492

12/5/2025

Table of Contents

Table of Contents 1

Table of Figures 3

No table of figures entries found. ..... **Error! Bookmark not defined.**

Table of tables 4

No table of figures entries found. ..... **Error! Bookmark not defined.**

Introduction 5

Components and equipment used 6

Circuit analysis 7

- Clock 9
- The 4s counter 10
- Boot stage 12
- Enable stage 13
- Sequence Selector 14
- Led Driving circuit 16
- Special trick circuit 22
- 7 segment display (Controller board) 24

Delay, power consumption and financial study 25

- Controller board 25
- Clock circuit 25
- 4s counter circuit 26
- Boot state detection circuit 27
- Enable condition detection circuit 27
- LED driving circuit 28

Sequence selector circuit 28
Special trick logic circuit 29
Problems, advantages of the design 31
Problems faced 31
key advantages over other alternatives 31
Conclusion 32

Table of Figures

Figure 1 - Circuit diagram of the different blocks. 7
Figure 2 - Clock circuit. 9
Figure 3 - 4 second counter. 10
Figure 4 - 4 seconds counter on Quartus. 11
Figure 5 - Simulation of the circuit. 11
Figure 6 - Boot stage circuit. 12
Figure 7 - Enable stage circuit. 13
Figure 8 - Sequence Selector diagram. 16
Figure 9 - Simulation of the Sequence Selector. 16
Figure 10 - LED driving circuit. 18
Figure 11 - LED driving circuit on Quartus (LED1). 19
Figure 12 - LED driving circuit on Quartus (LED2). 19
Figure 13 - LED driving circuit on Quartus (LED3). 20
Figure 14 - LED driving circuit on Quartus (LED4). 20
Figure 15 - Simulation of the LED driving circuit. 21
Figure 16 - Special trick circuit implementation on breadboard. 22
Figure 17 - Special trick implementation on Quartus II. 23
Figure 18 - Simulation of the circuit. 23
Figure 19 - 7 segment display. 24

Table of tables

Table 1 - Truth table of the Sequence Selector 15
Table 2 - LED driving based on the sequence..... 17
Table 3 - Power consumption and delay of the controller board. 25
Table 4 - Delay and power consumption of the clock. 26
Table 5 - Delay and power consumption of the 4 second counter circuit. 26
Table 6 - Delay and power consumption of the Boot state detection circuit. 27
Table 7 - Delay and power consumption of the enable detection circuit. 27
Table 8 - Delay and power consumption of the LED driving circuit. 28
Table 9 - Delay and power consumption of the sequence selector circuit..... 28
Table 10 - Delay and power consumption of the special trick logic circuit. 29

Introduction

In this project, techniques and skills acquired during this semester were applied to provide a practical implementation of theoretical concepts of logic design. A logic-controlled board was designed to light four LEDs based on different sequences and cases. This report will cover the basics of sequential and combinational logic blocks to create a functioning timed circuit. We will use memory elements (D flip-flops and shift registers) to track states and enable specific sequences in the circuit. Our design will combine both sequential and combinational logical circuits to ensure smooth functioning. First, we will explain the system behavior covering inputs, outputs and finite state machines (FSMs), along with the circuit simulation and Quartus implementation. Then, we will focus on the financial study, delay and power consumption analysis. Finally, problems faced, improvements and alternatives will be discussed.

Components and equipment used

- 8 Quad 2-input OR gate 7432
- 14 Quad 2-input AND gate 7408
- 5 Hex inverter 7404
- 5 D-flip flop chip 7474
- 1 clock 555 timer
- 1 XOR gate chip 7486
- 1 Dual 4-input AND gate 7421
- 1 seven segment display
- 1 BCD to 7-segment decoder CD4511
- 1 Dual 4-input OR gate CD4072
- 2 shift-registers 74195
- 1 3-to-8-line decoder 74138
- Breadboard and connecting wires
- Quartus II

Circuit analysis

Our design consists of different blocks built apart then re-assembled to constitute our final project. The different blocks are the clock, the 4 second counter, the boot circuit, the enable circuit, the LED driving circuit, the sequence selector and the special trick circuit.

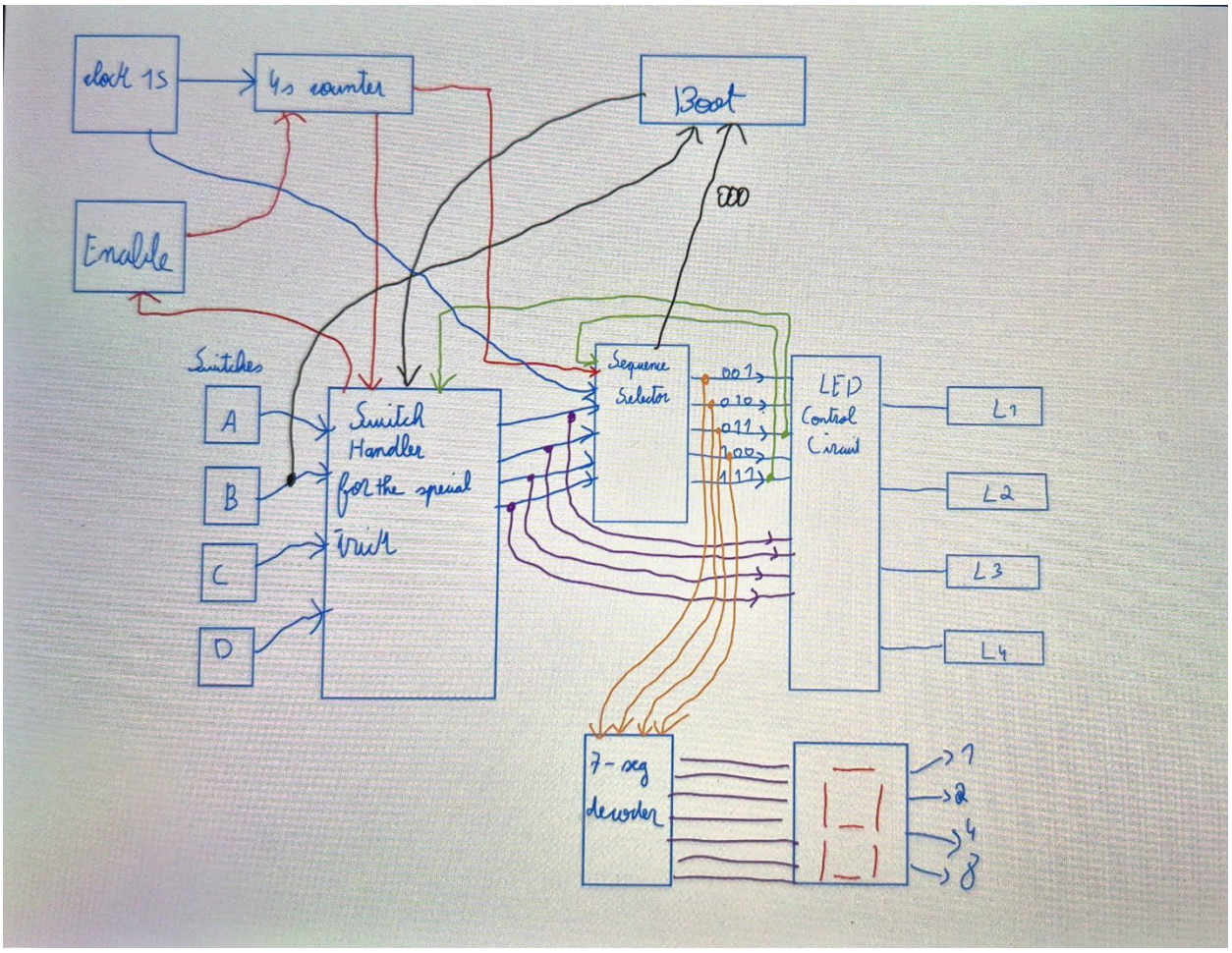

Figure 1 - Circuit diagram of the different blocks.

The finite state machine begins in the BOOT state, which is represented by the code 000. This initial state is responsible for system startup and determines whether the logic-controlled board should enter normal operating mode or the locked configuration. If Switch 2 is detected to be ON at startup, the system transitions directly to the LOCKED state, coded as 111. In this state, each switch is directly connected to its corresponding LED, eliminating any dynamic behavior or sequencing. However, if Switch 2 is OFF during initialization, the system

moves into the SEQUENCE SELECTOR state, labeled as 101. From this point, the system monitors which switch is last to be turned off for a duration of four seconds in order to determine the appropriate sequence state to enter. If Switch 1 is was last, the FSM transitions to SEQUENCE 1, identified by the state 001. In this mode, the switches follow a straightforward mapping where Switch 1 controls LED 1, Switch 2 controls LED 2, and so on until Switch 4 controls LED 4. If Switch 2 is held, the system shifts to SEQUENCE 2, represented as 010, where the switches are remapped in a rotated order so that Switch 1 controls LED 2, Switch 2 controls LED 3, Switch 3 controls LED 4, and Switch 4 controls LED 1. Similarly, activating Switch 3 brings the FSM to SEQUENCE 3 with state code 011. This sequence follows the LED pattern 3, 4, 1, 2, assigning control of LED 3 to Switch 1, LED 4 to Switch 2, LED 1 to Switch 3, and LED 2 to Switch 4. SEQUENCE 3 includes an additional functionality referred to as the SPECIAL TRICK. If two or more switches are ON for four continuous seconds while in SEQUENCE 3, the system transitions into the SPECIAL state, labeled 110. Although the LED mapping remains identical to SEQUENCE 3, any switch that remains ON for more than four seconds becomes temporarily disabled and will no longer control its corresponding LED. The switch must then be turned OFF for a continuous four seconds in order to regain functionality and return to SEQUENCE 3. Lastly, if Switch 4 is ON for four seconds, the FSM transitions to SEQUENCE 4, identified as state 100. In this configuration, the mapping is reversed: Switch 1 controls LED 4, Switch 2 controls LED 3, Switch 3 controls LED 2, and Switch 4 controls LED 1. In all four sequence states, the system is designed to return to the SEQUENCE SELECTOR if all switches are OFF simultaneously for four seconds.

Our design was based on this truth table, considering the states, switches and Treset as inputs. It is found on the excel sheet.

Clock

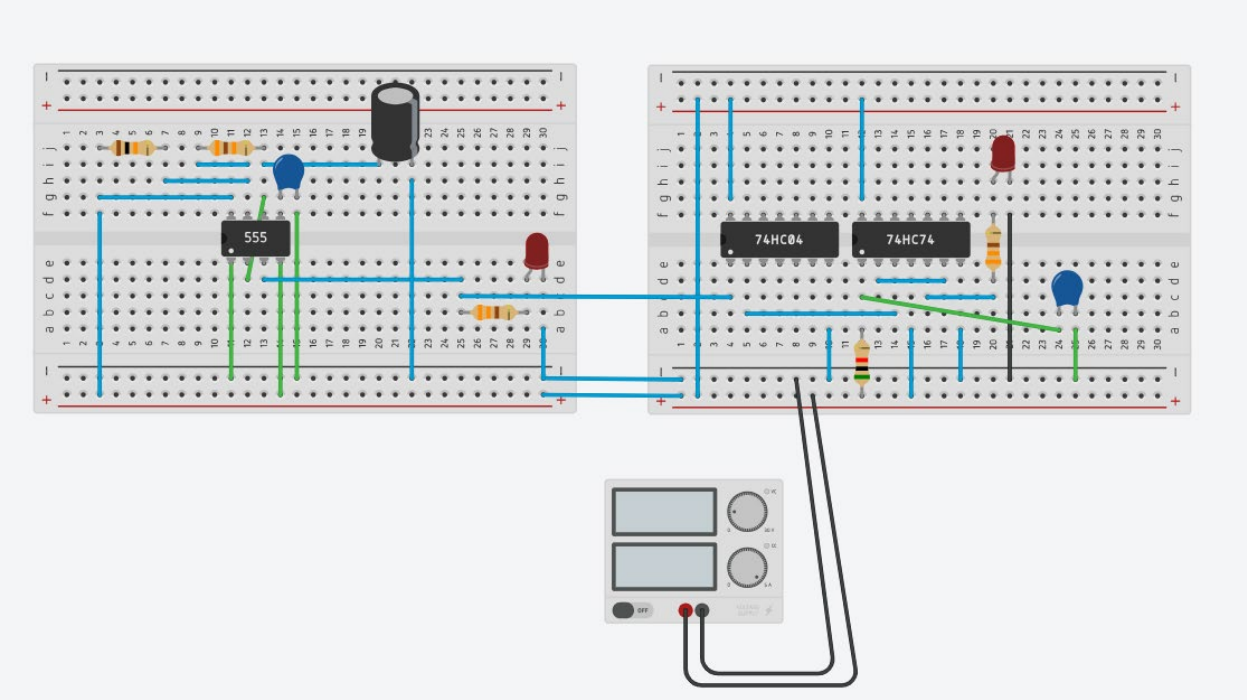

Figure 2 - Clock circuit.

First, we designed a 1 second clock, in other words, it will be HIGH for 0.5s and LOW for the other 0.5s. The clock circuit of the logic-controlled board was designed using a 555-timer configured in astable mode to generate a continuous square wave signal. A 10µF electrolytic capacitor, along with two resistors of 10kΩ and 31kΩ, was connected to the threshold and discharge pins of the timer, setting the charge and discharge times that define the clock frequency. However, the raw output of the 555-timer does not produce a perfectly symmetrical waveform, which can affect the performance of sequential logic circuits. To address this, the output was routed through a 74HC04 hex inverter. This component was used to clean the waveform by sharpening the rising and falling edges, ensuring well-defined digital transitions. The inverted signal was then fed into a single 74HC74 D-type flip-flop, which divided the clock frequency by two and produced a square wave with an accurate 50% duty cycle. This adjustment was critical to synchronize the timing across the system, providing a consistent and reliable clock signal for the operation of the finite state machine. The entire assembly was implemented on a breadboard, with careful attention to wiring, grounding, and power supply stability to ensure dependable performance.

The 4s counter

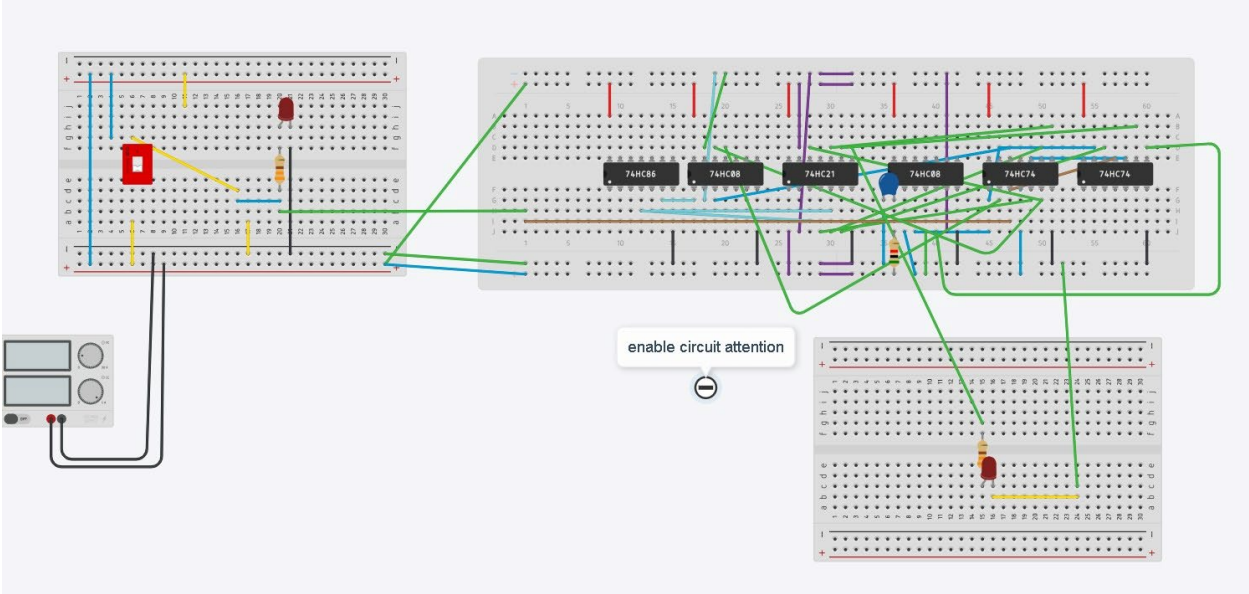

Figure 3 - 4 second counter.

The timing subsystem responsible for generating the 4-second low and 1-second high pulse was implemented using a 3-bit binary counter composed of three D-type flip-flops spread across two 74HC74 ICs. These flip-flops are clocked by the stable square wave previously produced and divided using the 555 timer and a D flip-flop. The counter progresses through a binary sequence, with the Q and $\bar{Q}$ outputs from each flip-flop fed into a combinational logic network formed using 74HC08 AND gates, 74HC86 XOR gates, and a 74HC21 dual 4-input AND gate. This logic was designed to evaluate specific counter states and generate a system output that remains LOW for approximately 4 seconds and then HIGH for 1 second, effectively creating a timing pulse suitable for the FSM reset mechanism. A ceramic capacitor and resistor were also placed at the input stage, forming a simple RC delay circuit that interfaces with an AND gate. This ensures that any toggling of the enable line resets the counter cleanly, allowing the system to initialize into a known state upon startup or manual reset.

We simulated this block on Altera Quartus II as well.

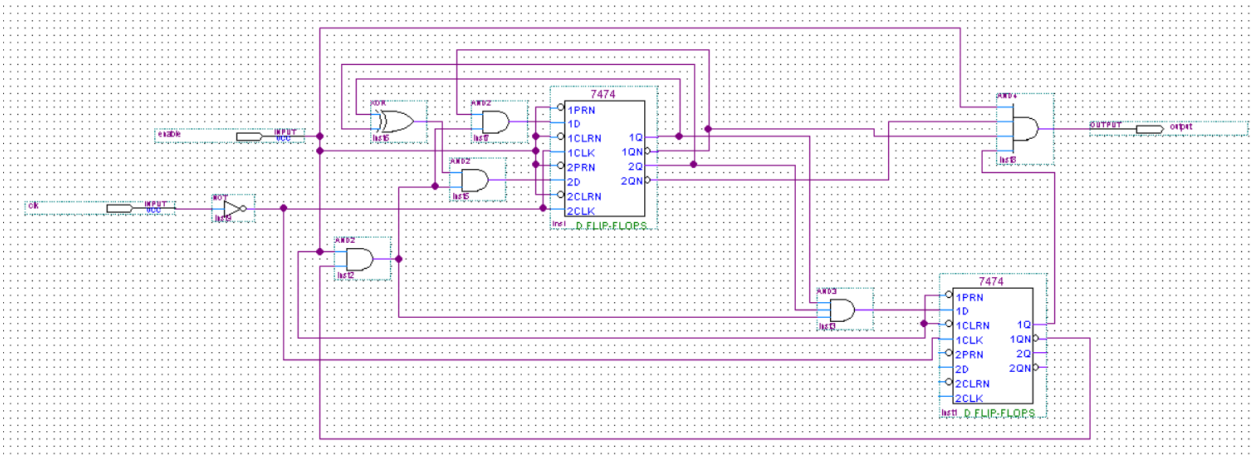

Figure 4 - 4 seconds counter on Quartus.

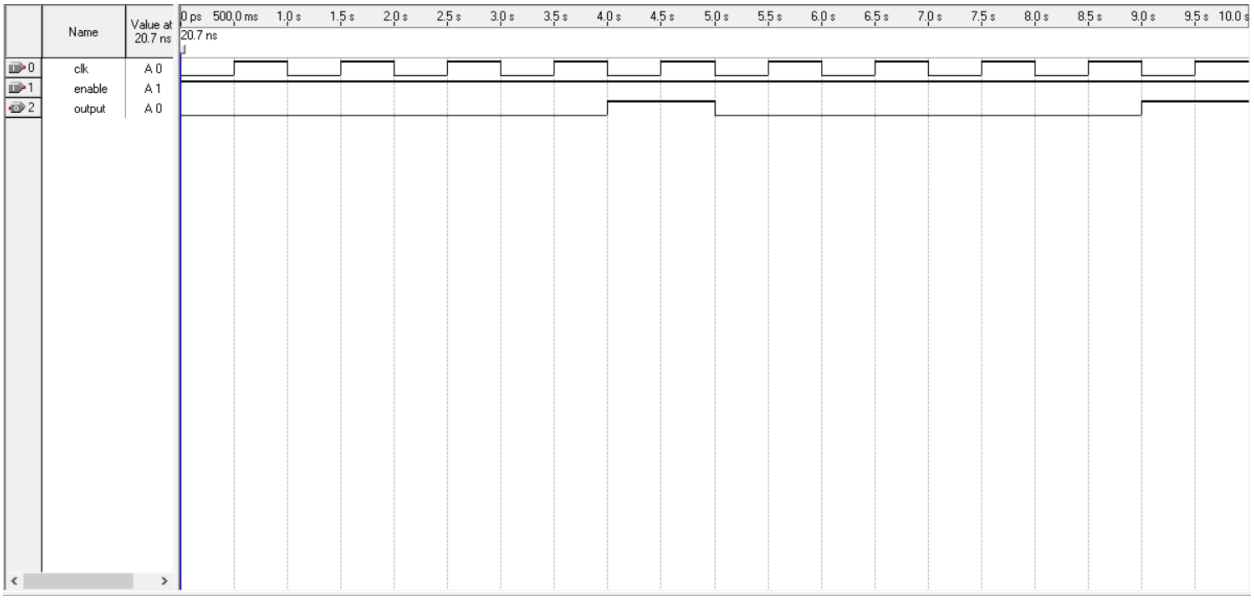

Figure 5 - Simulation of the circuit.

As we can see in the simulation, the input clock signal is of duration one second as the one designed previously. The 4 second counter counts correctly 4 seconds LOW signal and then outputs for 1 second as HIGH. This part will be used in different blocks for memory operations.

Boot stage

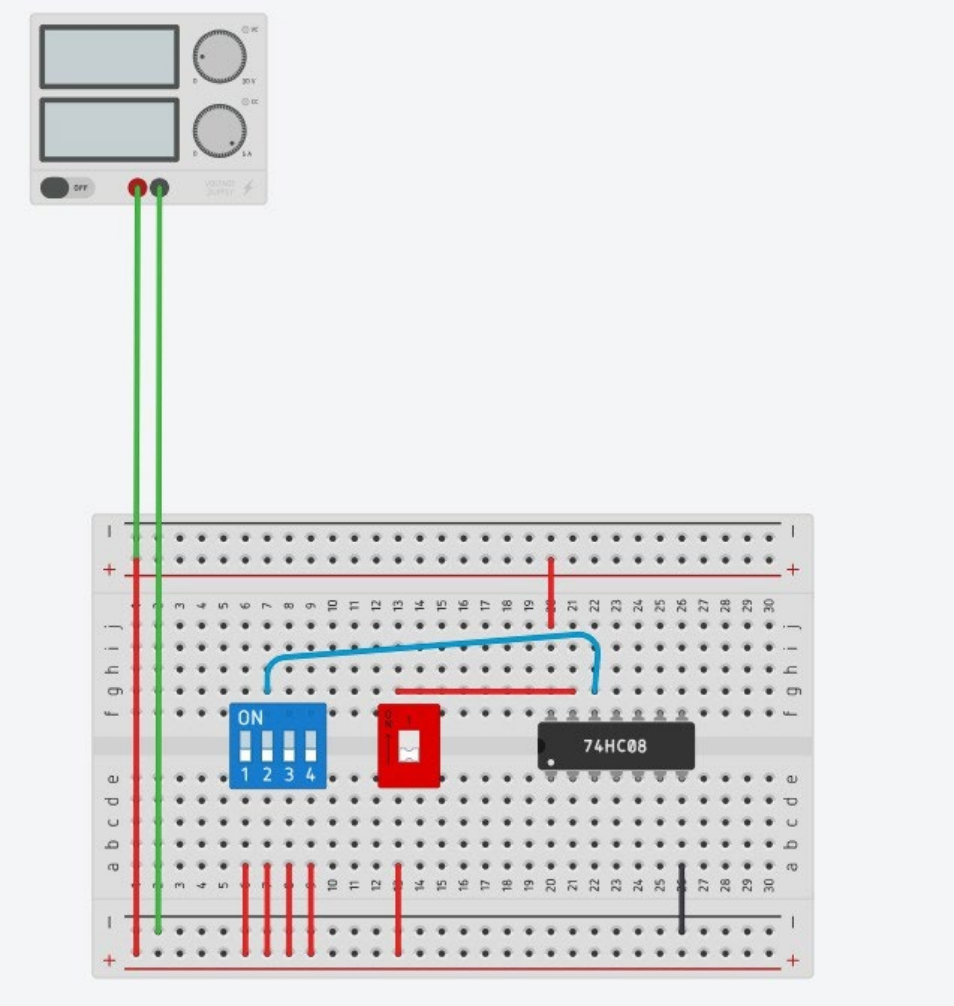

Figure 6 - Boot stage circuit.

The BOOT state was implemented using a basic combinational logic circuit that detects the startup condition of the system. A DIP switch was used to simulate the status of the four user switches at power-up, with a particular focus on Switch 2. The signal corresponding to the BOOT state, which is encoded as 000, is combined using an AND gate with the signal from Switch 2. The 74HC08 quad AND gate was used to perform this logic operation. If the output of this gate is HIGH, meaning that the system is in the BOOT state and Switch 2 is ON, the logic sends a signal that triggers the transition to the LOCKED state. Otherwise, if Switch 2 is OFF, the system proceeds to the SEQUENCE SELECTOR state. This simple logic effectively

differentiates between the two possible initial operating modes and ensures that the correct state is selected immediately after powering on the circuit.

Enable stage

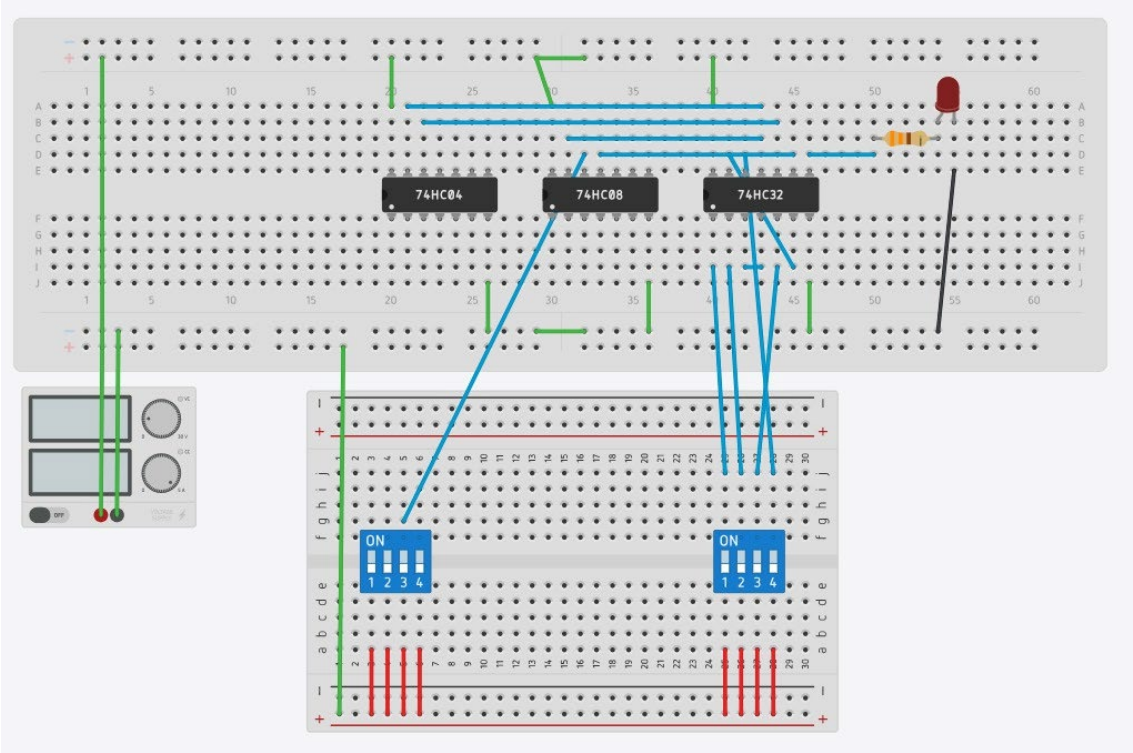

Figure 7 - Enable stage circuit.

The enable circuit was implemented to determine whether the system is currently operating within the standard sequences, such as Sequence 1, Sequence 2, or Sequence 4, or within the advanced behaviors associated with Sequence 3 and the Special Trick. This distinction is essential, as certain transitions and features depend on knowing which group of states the system is currently in. The logic design begins by passing the signals from all four users' switches into a 74HC32 OR gate. The output of this OR gate is then inverted using the 74HC04 hex inverter, producing a signal that is HIGH only when all switches are OFF. This result is then combined with the signal from Switch 3 using a 74HC08 AND gate. The final output of this logic network serves as an enable signal, which becomes active when Switch 3 is ON while all other switches are OFF. This mechanism is used to detect transitions into or out of the special logic behavior associated with Sequence 3. It ensures the system can distinguish

between standard and advanced sequences, enabling accurate control of state transitions within the finite state machine.

Sequence Selector

We directly simulated the sequence selector on Quartus II due to some issues from the circuit simulator used for the rest of the parts, but first, let's explain how the circuit works. The Sequence Selector plays a central role in the logic-controlled board, acting as the decision-making stage that determines which operational sequence the system should enter based on the state of the input switches. To achieve this, the design incorporates two 74195 shift registers, several logic gates, and a 3-to-8 line decoder. The first shift register is responsible for storing the previous state of the switches. Its clock is enabled when at least one of the four switches is ON and the system is not in the Locked state or Sequence 1. This ensures that a valid switch activation is detected under the correct operating conditions. The output from this shift register captures the switch configuration, which is then fed into a combinational logic circuit that interprets the input pattern. Based on the truth table derived and shown in the documentation, the circuit translates the active switch into a corresponding 3-bit code (S2, S1, S0), each representing a specific operational sequence. These three signals are then loaded into a second shift register, which is clocked only when all switches are OFF for a continuous period, representing the system's reset condition. This mechanism ensures that the system will only commit to a new sequence after confirming that the user has fully disengaged all inputs, thus preventing accidental transitions. The output of the second shift register is connected to a 74138 3-to-8 line decoder, which activates one of eight output lines corresponding to the possible sequences. These outputs are active LOW and are inverted to drive the following modules as needed. This staged structure, combining state memory, logic decoding, and controlled transitions, allows the Sequence Selector to manage the system's dynamic behavior in a robust and deterministic manner.

The design was implemented based on the below truth table.

Table 1 - Truth table of the Sequence Selector

SW1	SW2	SW3	SW4	S2	S1	S0
0	**0**	**0**	**0**	0	0	1
0	**0**	**0**	**1**	1	0	0
0	**0**	**1**	**0**	0	1	1
0	**0**	**1**	**1**	X	X	X
0	**1**	**0**	**0**	0	1	0
0	**1**	**0**	**1**	X	X	X
0	**1**	**1**	**0**	X	X	X
0	**1**	**1**	**1**	X	X	X
1	**0**	**0**	**0**	0	0	1
1	**0**	**0**	**1**	X	X	X
1	**0**	**1**	**0**	X	X	X
1	**0**	**1**	**1**	X	X	X
1	**1**	**0**	**0**	X	X	X
1	**1**	**0**	**1**	X	X	X
1	**1**	**1**	**0**	X	X	X
1	**1**	**1**	**1**	1	1	1

After simplification using Karnaugh maps, the obtained formulas for each of S2, S1 and S0 are:

- S2 = SW4
- S1 = SW4' . SW2' + SW3
- S0 = SW2 + SW3

Then we simulated these functions of Quartus as described previously.

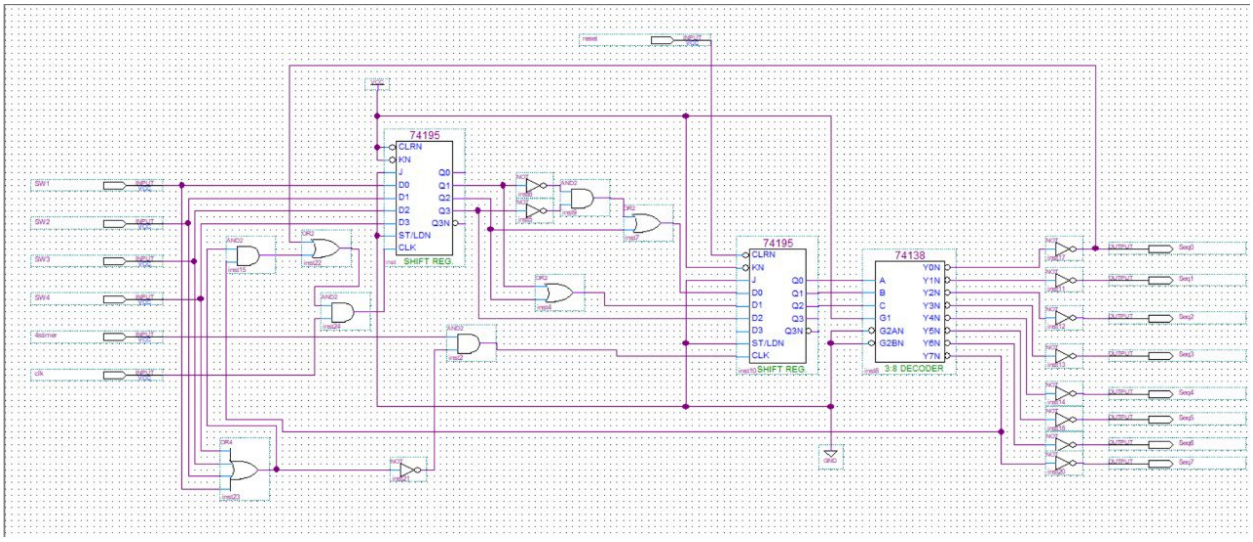

Figure 8 - Sequence Selector diagram.

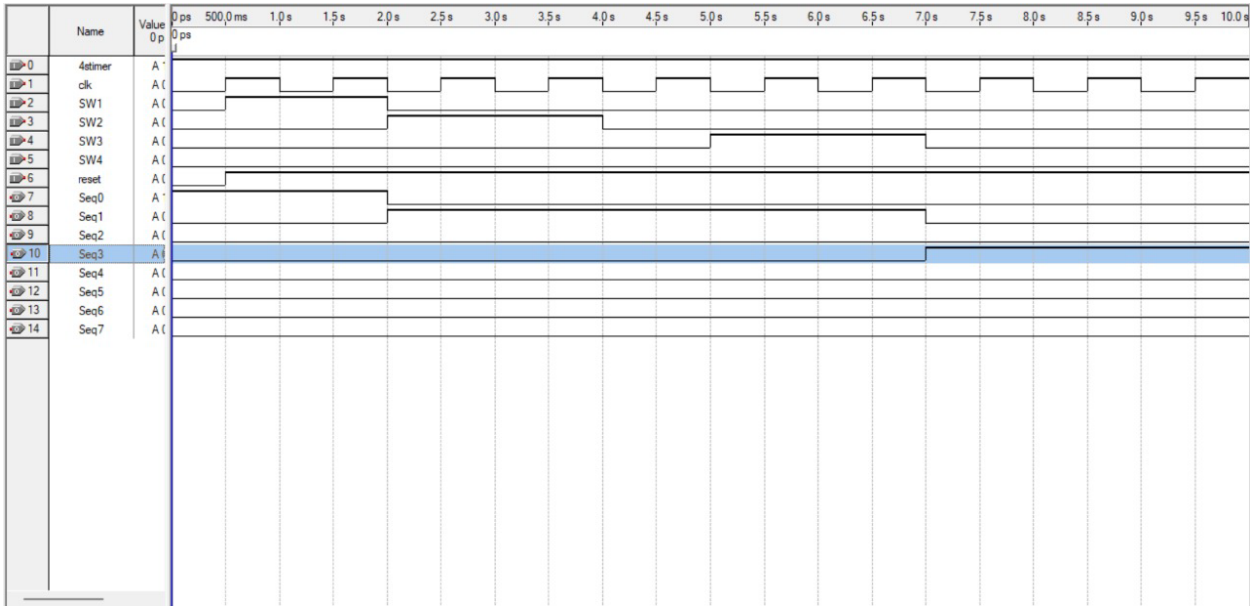

Figure 9 - Simulation of the Sequence Selector.

We can observe that for each switch that is ON its corresponding sequence will become the output coming out of the decoder (Seq0, Seq1, Seq2, Seq3…).

Led Driving circuit

For this part, we designed a circuit that will determine which LED should turn ON for which switch, at which sequence (based on the outputs of the switches, and of the sequence selector, explained previously).

The following table describes the logic used to control each LED based on the active sequence and the associated switch input. Each LED can be triggered by a specific switch depending on the current sequence. In the Locked state (Sequence 7), each LED is also linked to a fixed switch for direct control.

Table 2 - LED driving based on the sequence.

LED	Sequence 1 (Seq1)	Sequence 2 (Seq2)	Sequence 3 (Seq3)	Sequence 4 (Seq4)	Locked State (Seq7)
LED1	Switch 1	Switch 4	Switch 3	Switch 2	Switch 1
LED2	Switch 2	Switch 1	Switch 4	Switch 3	Switch 2
LED3	Switch 3	Switch 2	Switch 1	Switch 2	Switch 3
LED4	Switch 4	Switch 3	Switch 2	Switch 1	Switch 4

Following this logic, we understand that each LED output is the result of the OR of all the possible combinations for each LED, leading to this circuit on breadboard.

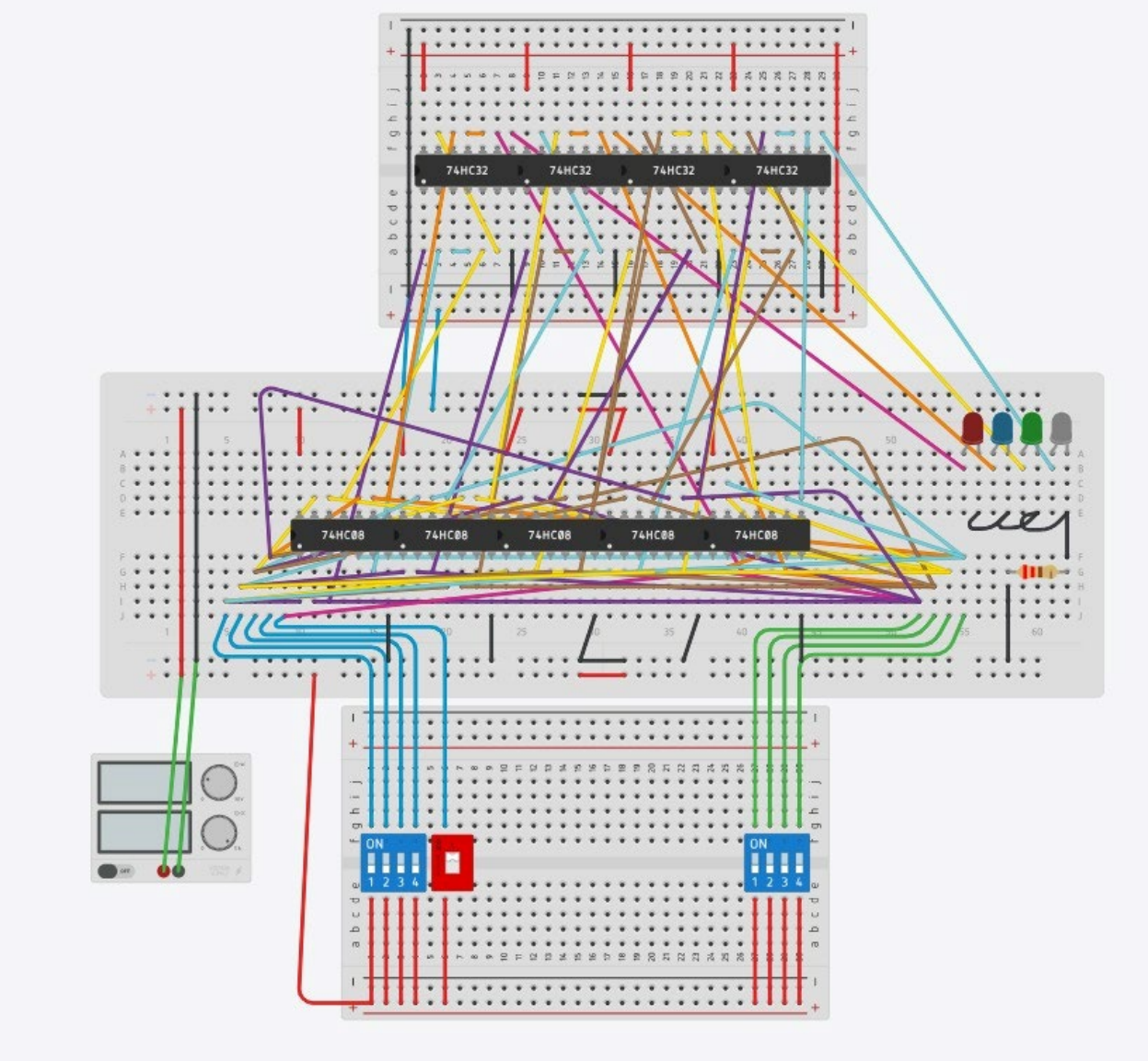

Figure 10 - LED driving circuit.

For clarity, we implemented it as well on Quartus II.

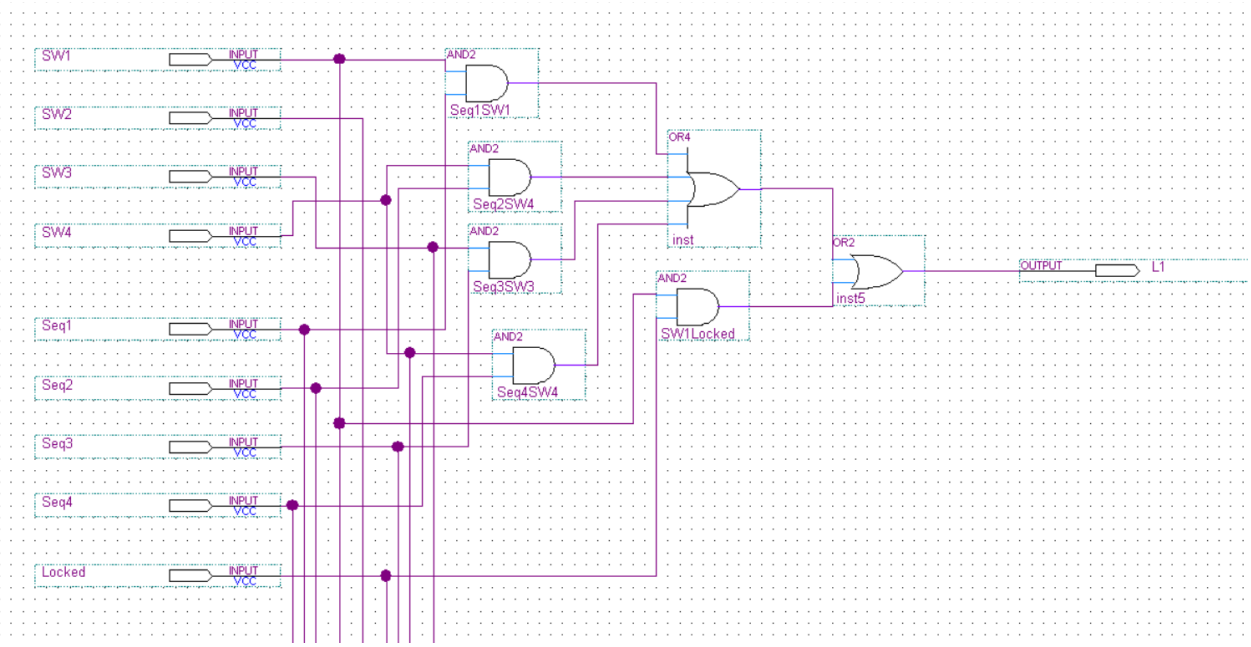

Figure 11 - LED driving circuit on Quartus (LED1).

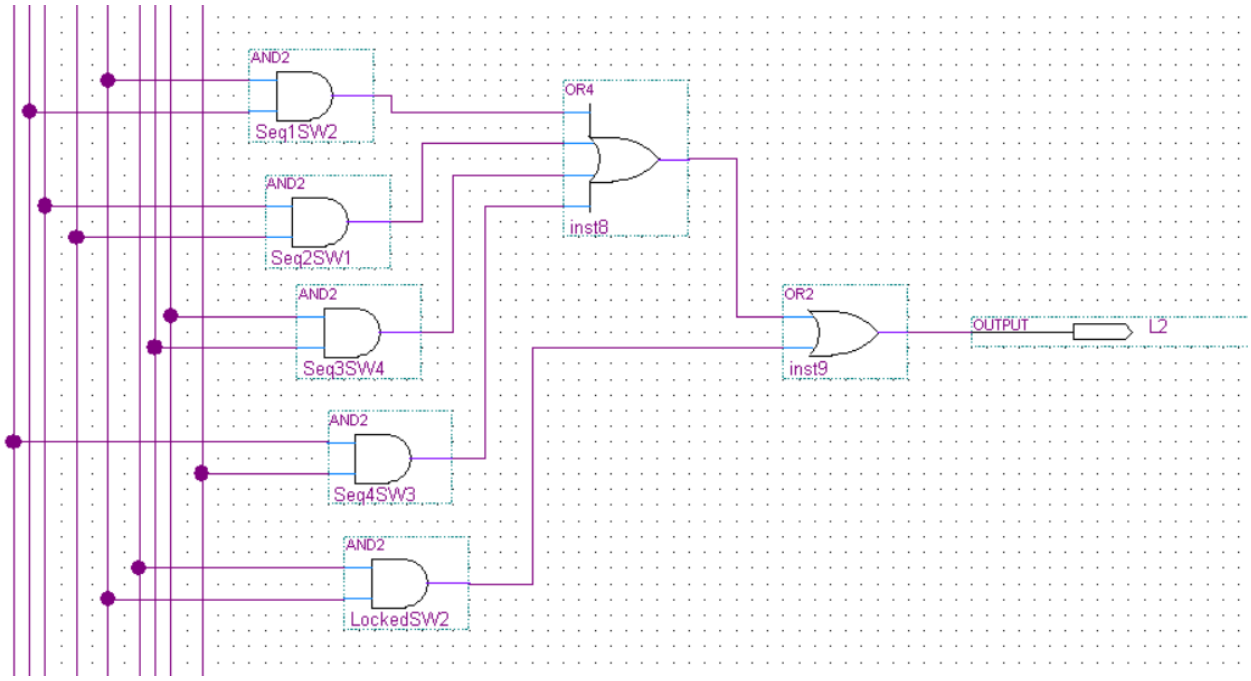

Figure 12 - LED driving circuit on Quartus (LED2).

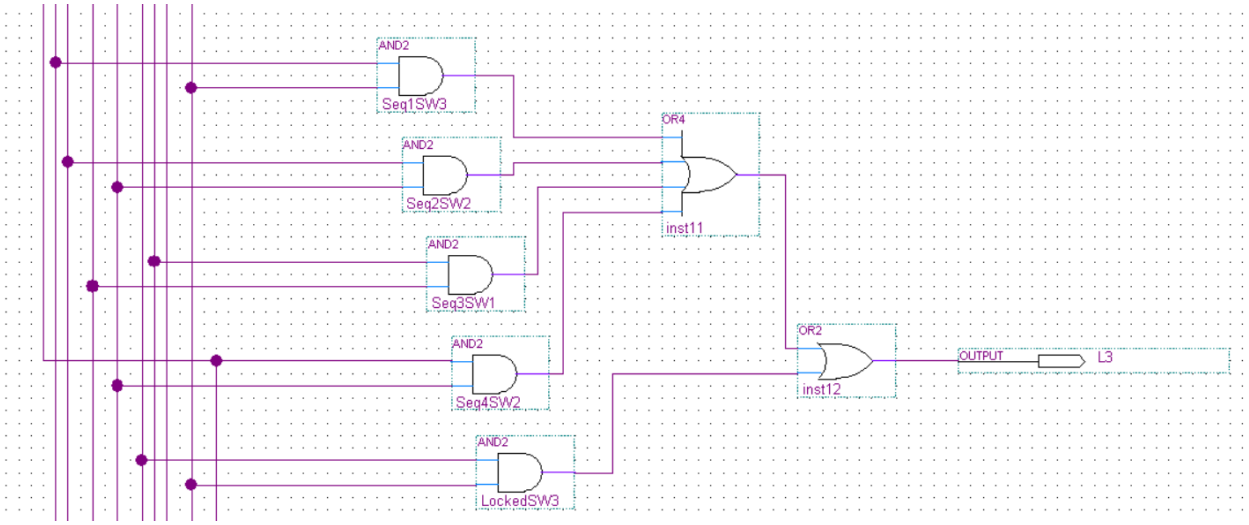

Figure 13 - LED driving circuit on Quartus (LED3).

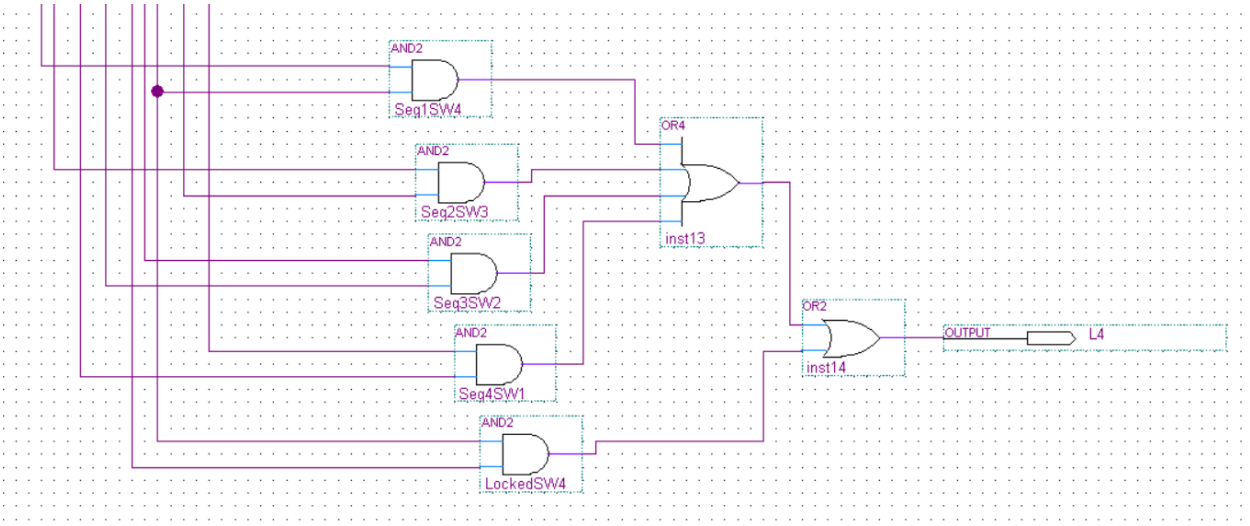

Figure 14 - LED driving circuit on Quartus (LED4).

Each of these screenshots corresponds to each LED as described in the table, and we can clearly observe on each AND gate which inputs are ANDed together (ex in the figure above: Seq1SW4 is the AND gate joining Seq1 and SW4 when ON.).

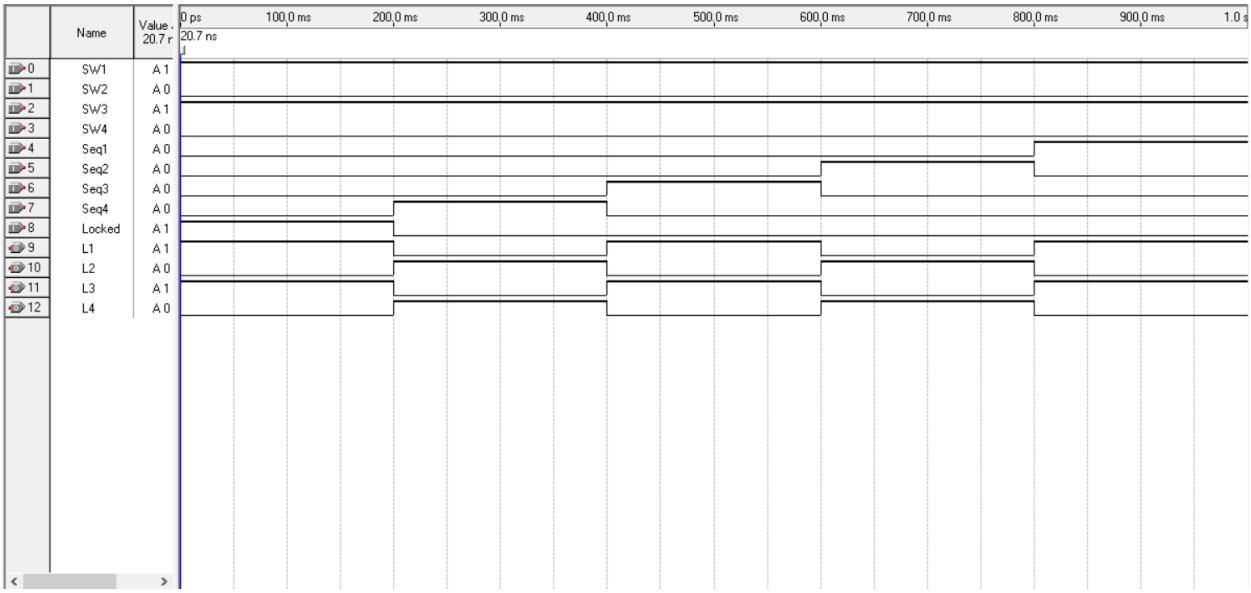

Figure 15 - Simulation of the LED driving circuit.

After simulation, for switches 1 and 3 being continuously ON and for each of the 5 states (Locked, Seq1 to 4), we can observe the corresponding output. We can see the following:

- Locked ON →SW1 lights L1, SW3 lights L3
- Seq4 ON → SW1 lights L4, SW3 lights L2
- Seq3 ON → SW1 lights L3, SW3 lights L1
- Seq2 ON → SW1 lights L2, SW3 lights L4
- Seq1 ON → SW1 lights L1, SW3 lights L3

This circuit alone is combinational.

Special trick circuit

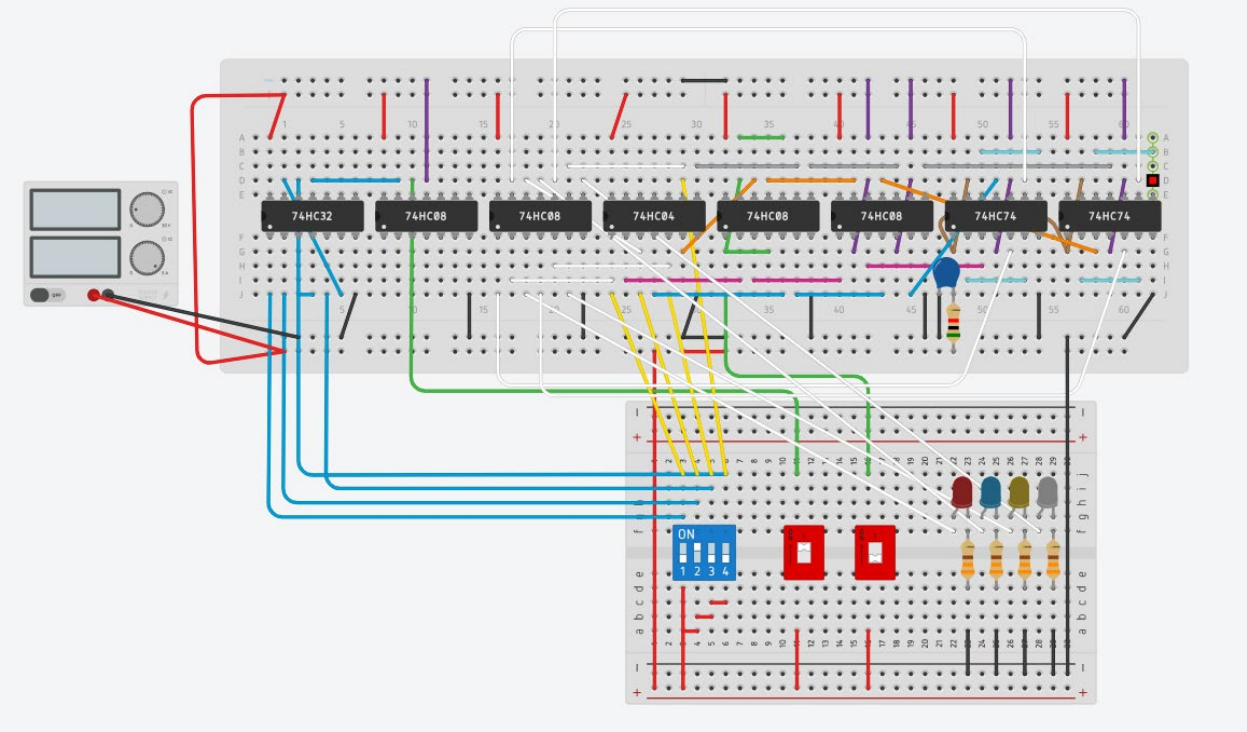

Figure 16 - Special trick circuit implementation on breadboard.

The Special Trick functionality, which operates exclusively within Sequence 3, was implemented using four D-type flip-flops to act as flags corresponding to each of the four user switches. Each flip-flop maintains the enabled or disabled status of its respective switch. When the output of a flip-flop is set to logic HIGH, it signifies that the associated switch is currently active and able to control the system. The disabling mechanism is triggered only when three specific conditions are met simultaneously: at least one switch is in the ON position, the system is operating in Sequence 3, and a rising edge is detected on the four-second clock signal. When these conditions are fulfilled, each switch input is individually inverted and then ANDed with the common condition logic. This ensures that only switches which are in the OFF state at the time of the rising edge receive the clock signal necessary to toggle their corresponding flip-flops. As a result, only the flags of the OFF switches are changed, effectively disabling them. To re-enable a previously disabled switch, it must again be in the OFF position when the next rising edge of the four-second clock occurs, which causes its respective flip-flop to toggle back to the active state. This design provides a reliable

and efficient way to implement temporary switch deactivation based on user behavior, enhancing the interactive complexity of Sequence 3 while preserving full control logic within the system.

We simulated it as well on Quartus II.

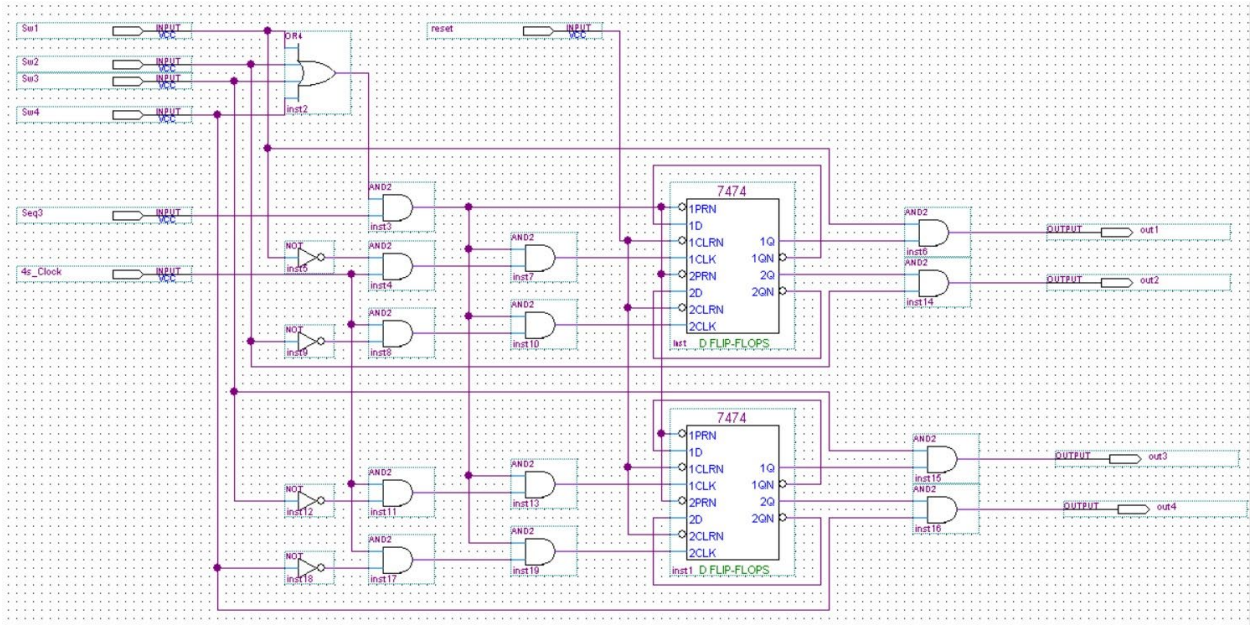

Figure 17 - Special trick implementation on Quartus II.

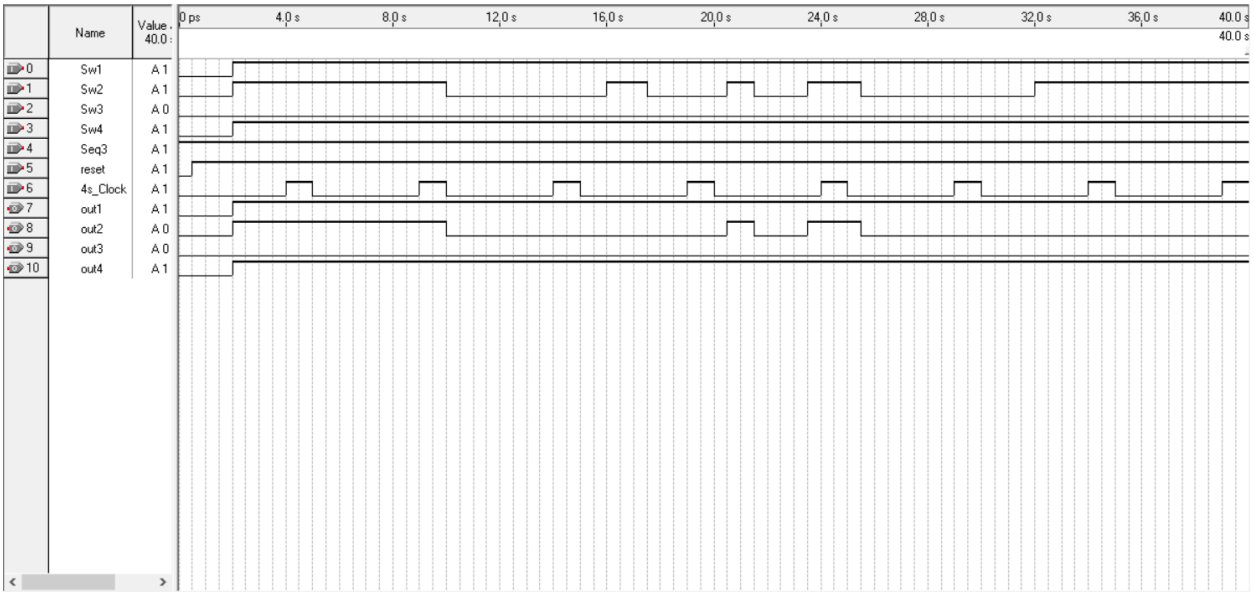

Figure 18 - Simulation of the circuit.

We can observe that we have switches 1, 2 and 4 ON. At 10s, switch 2 is OFF for more than 4 seconds, leading the next toggle of the switch to not affect the corresponding LED of switch 2. Then after being OFF for more than 4 seconds, it will reactivate normally.

7 segment display (Controller board)

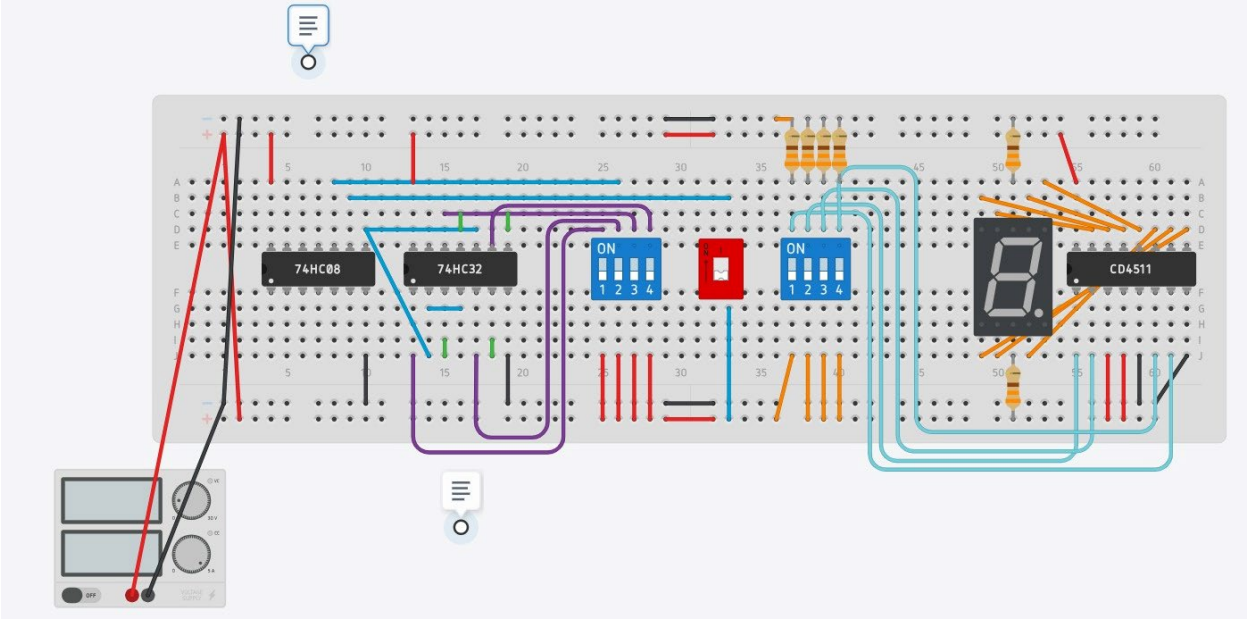

Figure 19 - 7 segment display.

We were asked to design a 7-segment display that will tell us in which sequence we are when in “practice mode”.

This circuit acts as the central interface between user input and the FSM logic, handling both the state display and state control mechanisms. On the left side of the board, the first DIP switch represents the four main inputs of the system — the user switches. These are first ANDed with a BOOT gate logic condition that is only true when the system is in State 0 (BOOT) and Switch 2 is ON. This AND result is then ORed with the original switch inputs before being fed into the input shift register of the sequence selector logic. This design allows the LOCKED state (State 7) to be reliably entered upon startup if the boot condition is detected.

On the right side of the board, a second set of DIP switches allows manual simulation of FSM states 2, 3, and 4, which are directly connected to a CD4511 BCD to 7-segment decoder. This decoder drives the display and converts 4-bit state codes into the corresponding segment combinations to light up the correct number on the 7-segment display. Pull-down resistors ensure clean logic levels on the inputs, and current-limiting resistors protect the 7-segment display.

Delay, power consumption and financial study

In this part, we will discuss the propagation delay and power consumption generated by each chip in our circuit, for each block.

Controller board

Table 3 - Power consumption and delay of the controller board.

IC Name	Function	Units Used	Power per Unit	Delay per Unit
74HC08	Quad AND Gate	1	1.5 mW	10 ns
74HC32	Quad OR Gate	1	1.5 mW	10 ns
CD4511	BCD to 7-Segment Decoder	1	3.0 mW	250 ns
Total			6.0 mW	~260 ns

Clock circuit

This section presents the power consumption and timing analysis for the Clock Circuit. The circuit includes a 555 timer generating a square wave, a 74HC04 inverter to clean the

signal, and a 74HC74 flip-flop to divide the frequency and ensure a 50% duty cycle. Only one gate from the 74HC04 and one flip-flop from the 74HC74 are used.

Table 4 - Delay and power consumption of the clock.

IC Name	Function	Used Units	Power Consumption (mW)	Propagation Delay (ns)
555 Timer	Clock Signal Generator	1	5.0	200
74HC04	Hex Inverter	1 gate (of 6)	1.2	8
74HC74	Dual D Flip-Flop	1 flip-flop (of 2)	1.0	12
Total			7.2 mW	220 ns

4s counter circuit

This circuit generates a 4-second pulse using XOR, AND, and D flip-flops to implement timing logic.

Table 5 - Delay and power consumption of the 4 second counter circuit.

IC Name	Function	Used Units	Power Consumption (mW)	Propagation Delay (ns)
74HC86	Quad XOR Gate	1	1.5	10
74HC08	Quad AND Gate	2	3.0	10

74HC21	Dual 4-input AND	1	1.5	18
74HC74	Dual D Flip-Flop	2	4.0	12
Total			10.0 mW	40 ns

Boot state detection circuit

Used to determine whether the system should start in the LOCKED state or proceed to sequence logic.

Table 6 - Delay and power consumption of the Boot state detection circuit.

IC Name	Function	Used Units	Power Consumption (mW)	Propagation Delay (ns)
74HC08	Quad AND Gate	1	1.5	10
Total			1.5 mW	10 ns

Enable condition detection circuit

Checks whether the system is in Sequence 3 group using NOR-like logic based on inverted switches.

Table 7 - Delay and power consumption of the enable detection circuit.

IC Name	Function	Used Units	Power Consumption (mW)	Propagation Delay (ns)
74HC04	Hex Inverter	1	1.2	8

74HC32	Quad OR Gate	1	1.5	10
74HC08	Quad AND Gate	1	1.5	10
Total			4.2 mW	28 ns

LED driving circuit

Implements switch-to-LED logic mapping for different sequences using multiple AND and OR gates.

Table 8 - Delay and power consumption of the LED driving circuit.

IC Name	Function	Used Units	Power Consumption (mW)	Propagation Delay (ns)
74HC08	Quad AND Gate	5	7.5	10
74HC32	Quad OR Gate	1	1.5	10
Total			9.0 mW	20 ns

Sequence selector circuit

Selects the active sequence using shift registers and decoder based on switch conditions.

Table 9 - Delay and power consumption of the sequence selector circuit.

IC Name	Function	Used Units	Power Consumption (mW)	Propagation Delay (ns)

74195	4-bit Shift Register	2	6.0	20
74HC138	3-to-8 Decoder	1	2.5	18
74HC04	Hex Inverter	2	2.4	8
74HC32	Quad OR Gate	2	3.0	10
74HC08	Quad AND Gate	1	1.5	10
Total			15.4 mW	66 ns

Special trick logic circuit

Enables or disables switches dynamically during Sequence 3 using D flip-flops and timing logic.

Table 10 - Delay and power consumption of the special trick logic circuit.

IC Name	Function	Used Units	Power Consumption (mW)	Propagation Delay (ns)
74HC74	Dual D Flip-Flop	2	4.0	12
74HC08	Quad AND Gate	4	6.0	10
74HC04	Hex Inverter	1	1.2	8
74HC32	Quad OR Gate	1	1.5	10
Total			12.7 mW	50 ns

This project is constituted of:

- 8 Quad 2-input OR gate 7432 (0.45$)
- 14 Quad 2-input AND gate 7408 (0.30$)
- 5 Hex inverter 7404 (0.30)
- 5 D-flip flop chip 7474 (0.50$)
- 1 clock 555 timer (0.25$)
- 1 XOR gate chip 7486 (0.50$)
- 1 Dual 4-input AND gate 7421 (0.50$)
- 1 seven segment display (reused from an older project)
- 1 BCD to 7-segment decoder CD4511 (0.50$)
- 1 Dual 4-input OR gate CD4072 (0.50)
- 2 shift-registers 74195 (1.50$)
- 1 3-to-8-line decoder 74138 (0.50$)

The final cost of our project elevates to 17.55$ without considering the cost of breadboard and power supplies (1.5 V batteries) that were already available from previous projects.

Problems, advantages of the design

Problems faced

- monitoring faults without proper debugging material

- time and wire management

- filling the system's truth table

- finding the solution to a dynamic system via memory element (sequential circuits) instead of filling a truth table for every single combination

- connecting the stand-alone parts

key advantages over other alternatives

- standalone functioning parts that still works even if the circuit is separated. It makes it a system built by joining subsystems taking a form of a huge complex interconnected network of nodes with a central decision element

- sequential behavior: Using memory elements and assumed unreachable conditions, the system uses flipflops to dynamically handle input combinations as local variables.

Conclusion

This project successfully demonstrated the practical implementation of key concepts in logic design through the construction of a fully functional logic-controlled board. By integrating both combinational and sequential circuits, we were able to develop a versatile system capable of dynamically responding to user input via switches, controlling LED outputs according to complex state-based logic, and incorporating time-sensitive behaviors such as the four-second pulse delay and the "special trick" sequence.

Throughout the design and implementation phases, we tackled challenges including timing synchronization, circuit modularization, and the development of an effective state machine using D flip-flops and shift registers. Each block—from the clock generator to the sequence selector and LED drivers—was carefully designed, tested, and validated using Quartus II simulations and physical breadboarding.

Our final system reflects a modular, scalable architecture where individual subsystems can function independently while contributing to the overall behavior of the FSM. The project not only solidified our understanding of digital logic design but also highlighted the importance of debugging, power efficiency, and system-level thinking. In future iterations, the board could be extended with more sequences, wireless inputs, or integration with microcontrollers for added flexibility and programmability.