Lebanese American University



ELE 303 – Electric Circuits Lab

Section 32

Dr. Ehab Shayya

Final Project

Lama Haidar-Ahmad 202205709

Joe Daou 202301424

Ali Ismail 202302900

Christopher Farah 202306349

Table of Contents

[Introduction 4](#_Toc197888446)

[Components and equipment used 5](#_Toc197888447)

[Analysis 6](#_Toc197888448)

[Paper design 8](#_Toc197888449)

[State Table 8](#_Toc197888450)

[Karnaugh Maps and Boolean Equations 8](#_Toc197888451)

[Schematic Diagram 11](#_Toc197888452)

[Quartus design and analysis 13](#_Toc197888453)

[Breadboard design and analysis 15](#_Toc197888454)

[Financial study, focusing on minimizing the cost of the design 17](#_Toc197888455)

[Delay calculation 18](#_Toc197888456)

[Power consumption analysis 19](#_Toc197888457)

[Problems faced during the project 20](#_Toc197888458)

[Key design points that present advantages over alternative designs 21](#_Toc197888459)

[Conclusion 22](#_Toc197888460)

[References 23](#_Toc197888461)

[Figure 1: state diagram 7](#_Toc197888481)

[Figure 2: 4-bit parallel register 11](#_Toc197888482)

[Figure 3: 3bit parallel register 12](#_Toc197888483)

[Figure 4:The quartus sketch of the circuit 14](#_Toc197888484)

[Figure 5: Circuit Diagram 15](#_Toc197888485)

[Table 1: List of Required Components and Quantities 17](#_Toc197888495)

# Introduction

The project develops a digital magic box system using basic principles of digital design. Rather than a traditional lock-based system, the digital magic box accepts binary inputs and generates fascinating outputs that provide a sense of digital magic. Using combinational logic components and sequential logic components, including logic gates and flip-flops, the design demonstrates that digital circuits are capable of delivering creative and interactive functionality. During the process, the project reinforces essential Boolean algebra and circuit analysis principles and demonstrates digital system creativity.

# Components and equipment used

* And IC 7408 10
* Or IC 7432 10
* NAND IC 7400 10
* XOR IC 7486 10
* Not IC 7404 10
* D flip flop IC 7474 10
* Timer IC LM555 4
* PCB
* IC chip chairs
* Breadboard

# Analysis

The MAGIC BOX is an applied project for digital logic design, which includes combinational and sequential circuits, finite state machines (FSMs), and user interfaces. We look into the functional nature, how it processes the input, and the underlying logic that determines the output states of the entire system. The board has 4 toggle switches and 4 LED indicators, and each is assigned a color. At first, each switch is directly linked to a lamp of the same color. But the system adjusts the mapping on the fly, according to what the last switch that was turned off in the on-off pattern was, and modifies the corresponding LED in the sequence.

The system supports four possible sequences based on which switch is turned off last:

* **Switch 1** → Sequence 1: 1 → 2 → 3 → 4
* **Switch 2** → Sequence 2: 2 → 3 → 4 → 1
* **Switch 3** → Sequence 3: 3 → 4 → 1 → 2 (includes Special Trick #3)
* **Switch 4** → Sequence 4: 4 → 3 → 2 → 1

The correct sequence is activated only after all switches are turned off and a timeout (4 seconds) occurs(A **4-second timer** is used to reset the system and trigger sequence changes.). A 7-segment display is used to show the active sequence for practice and debugging purposes.

The system’s behavior is controlled by a finite state machine (FSM), switching between states in response to input and internal computation:

Boot State: The starting point of the system

Locked State: Each switch has no control over its LED (no LED turns on).

Sequence Detector State: Examines the last switch switched off to detect the next active sequence.

Sequence States (1–4): Determines each sequence's LED behavior.

Special Trick State: Turns off a switch temporarily a switch when its cap is removed in Sequence 3.

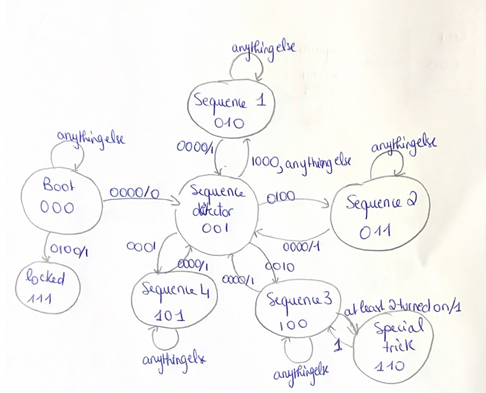


Figure 1: state diagram

Starting at the Boot state (000) for an input of 0100, and after 4 seconds have passed, we will go to locked state (111) where all LEDs are turned off and can’t be turned on, and we need to restart the system and return to normal operation with any other input, or for input 0000 and after less than 4 seconds we will be in the system detector (001)

Now in the sequence detector state:

If the input is 1000, we will go to the sequence 1 state (010). Once in this state, if we get 0000 and after 4 seconds have passed, we will go back to the sequence detector; any other input, we will stay in the sequence 1 state.

If the input is 0100, we will go to the sequence 2 state (011). Once in this state, if we get 0000 and after 4 seconds have passed, we will go back to the sequence detector; any other input, we will stay in the sequence 2 state.

If the input is 0010, we will go to the sequence 3 state (100). Once in this state, if we get 0000 and after 4 seconds have passed, we will go back to the sequence detector; any other inputs having at least 2 turned on and after 4s, we will go to the special trick state (110) where after 4s we go back to sequence 3, or else we will stay in sequence 3.

If the input is 0001, we will go to the sequence4 state (101). Once in this state, if we get 0000 and after 4 seconds have passed, we will go back to the sequence detector; any other input, we will stay in the sequence 4 state.

# Paper design

## State Table



It is given with given documents

## Karnaugh Maps and Boolean Equations

**For the k-maps, we used a website and we put it in the references.**

NSeq2(Seq2,Seq1,Seq0,Treset,SW1,SW2,SW3,SW4) = Sum(20, 33, 34, 49, 50, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 171, 171, 172, 173, 174,175, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255)

= Seq1'Seq0'TresetSW1'SW2SW3'SW4' + Seq1'Seq0SW1'SW2'SW3'SW4 + Seq1'Seq0SW1'SW2'SW3SW4' + Seq2SW4 + Seq2SW2 + Seq2Seq1 + Seq2Treset'SW3' + Seq2Seq0'SW3 + Seq2TresetSW1

NSeq1(Seq2,Seq1,Seq0,Treset,SW1,SW2,SW3,SW4) = Sum(20, 32, 35, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 147, 149, 150, 151, 153, 154, 155, 156, 157, 158, 159, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255)

= Seq2'TresetSW1'SW2SW3'SW4' + Seq2'Seq0SW3SW4 + Seq2'Seq0SW2 + Seq2'Seq0SW1 + Seq2'Seq1'Seq0SW3'SW4' + Seq1Treset' + Seq2'Seq1SW4 + Seq2'Seq1SW3 + Seq2'Seq1SW1 + Seq2Seq1'Seq0'TresetSW3SW4 + Seq2Seq1'Seq0'TresetSW2SW4 + Seq2Seq1'Seq0'TresetSW2SW3 + Seq2Seq1'Seq0'TresetSW1SW4 + Seq2Seq1'Seq0'TresetSW1SW3 + Seq2Seq1'Seq0'TresetSW1SW2 + Seq2Seq1Seq0

NSeq0(Seq2,Seq1,Seq0,Treset,SW1,SW2,SW3,SW4) = Sum(0, 20, 33, 36, 49, 52, 80, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 144, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255)

= Seq2'Seq1'Seq0'Treset'SW1'SW2'SW3'SW4' + Seq2'Seq1'TresetSW1'SW2SW3'SW4' + Seq0SW1'SW2'SW3'SW4 + Seq0SW1'SW2SW3'SW4' + Seq2'Seq1TresetSW1'SW2'SW3'SW4' + Seq1Seq0 + Seq2Seq1'TresetSW1'SW2'SW3'SW4' + Seq2Seq0

L1(Seq2,Seq1,Seq0,Treset,SW1,SW2,SW3,SW4) = Sum(71, 72, 73, 74, 75,76, 77, 78, 79, 88, 89, 90, 91, 92, 93, 94, 95, 97, 99, 101, 103, 105, 107, 109, 111, 113, 115, 117, 119, 121, 123, 125, 127, 130, 131, 134, 135, 138, 139, 142, 143, 146, 147, 150, 151, 154, 155, 158,159,161, 161, 163, 165, 167, 169, 171, 173, 175, 177, 179, 181, 183, 185, 187, 189, 191, 194, 195, 198, 199, 202, 203, 206, 207, 210, 211, 214, 215, 218, 219, 222, 223)

= Seq2'Seq1Seq0'SW1 + Seq2'Seq1Seq0SW4 + Seq2Seq0'SW3 + Seq2Seq1'Seq0SW4 + Seq2'Seq1Treset'SW2SW3SW4

L2(Seq2,Seq1,Seq0,Treset,SW1,SW2,SW3,SW4) = Sum(68, 69, 70, 76, 77, 78, 79, 84, 85, 86, 87, 94, 95, 92, 93, 104, 105, 106, 107, 108, 109, 110, 111, 120, 121, 122, 123, 124, 125, 126, 127, 127, 129, 131, 133, 135, 137, 139, 141, 143, 145, 147, 149, 151, 153, 155, 157, 159, 162, 167, 170, 171, 174, 175, 178, 179, 182, 183, 186, 187, 190, 191, 193, 195, 197, 199, 201, 203, 205, 207, 209, 211, 213, 215, 217, 219, 221, 223)

= Seq2'Seq1Seq0'SW2SW4' + Seq2'Seq1Seq0SW1 + Seq2Seq0'SW4 + Seq2Seq1'Seq0SW2'SW3SW4' + Seq2Seq1'SW2SW3SW4 + Seq2Seq1'Seq0SW1SW3 + Seq2Seq1'Seq0TresetSW3 + Seq2'Seq1Seq0'SW2SW3' + Seq1Seq0'SW1SW2SW4 + Seq2'Seq1Seq0'TresetSW2

L3(Seq2,Seq1,Seq0,Treset,SW1,SW2,SW3,SW4) = Sum(66, 67, 70, 71, 74, 75, 78, 79, 82, 83, 86, 87, 90, 91, 94, 95, 100, 101, 102, 103, 108, 109, 110, 111, 116, 117, 118, 119, 124, 125, 126, 127, 136, 137, 138, 139, 140, 141, 142, 143, 152, 153, 154, 155, 156, 157, 158, 159, 164, 165, 166, 167, 172, 173, 174, 175, 180, 181, 182, 183, 188, 189, 190, 191, 200, 201, 202, 203, 204, 205, 206, 207, 216, 217, 218, 219, 220, 221, 222, 223)

= Seq2'Seq1Seq0'SW3 + Seq2'Seq1Seq0SW2 + Seq2Seq0'SW1 + Seq2Seq1'Seq0SW2

L4(Seq2,Seq1,Seq0,Treset,SW1,SW2,SW3,SW4) = Sum(65, 67, 69, 71, 73, 75, 77, 79, 81, 83, 85, 87, 89, 91, 93, 95, 98, 99, 102, 103, 106, 107, 110, 111, 114, 115, 118, 119, 122, 123, 126, 127, 132, 134, 135, 140, 141, 142, 143, 148, 149, 150, 151, 156, 157, 158, 159, 168, 169, 170, 171, 172, 173, 174, 175, 184, 185, 186, 187, 188, 189, 190, 191, 196, 197, 198, 199, 204, 205, 206, 207, 212, 213, 214, 215, 220, 221, 222, 223)

= Seq2'Seq1Seq0'SW4 + Seq2'Seq1Seq0SW3 + Seq2Seq0'SW2SW4' + Seq2Seq0'SW2SW3 + Seq2Seq0'TresetSW2 + Seq2Seq1'Seq0SW1 + Seq2Seq1'SW1SW2 + Seq1Seq0'SW2SW4

a(Seq2,Seq1,Seq0,Treset,SW1,SW2,SW3,SW4) = Sum(96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223)

= Seq2'Seq1Seq0 + Seq2Seq0'

b(Seq2,Seq1,Seq0,Treset,SW1,SW2,SW3,SW4) = Sum(64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191)

= Seq2'Seq1 + Seq2Seq1'

c(Seq2,Seq1,Seq0,Treset,SW1,SW2,SW3,SW4) = Sum(64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223)

= Seq1Seq0' + Seq2Seq1'

d(Seq2,Seq1,Seq0,Treset,SW1,SW2,SW3,SW4) = Sum(96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223)

= Seq2'Seq1Seq0 + Seq2Seq0'

e(Seq2,Seq1,Seq0,Treset,SW1,SW2,SW3,SW4) = Sum(96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127)

= Seq2'Seq1Seq0

f(Seq2,Seq1,Seq0,Treset,SW1,SW2,SW3,SW4) = Sum(160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223)

= Seq2Seq1'Seq0 + Seq2Seq1Seq0'

g(Seq2,Seq1,Seq0,Treset,SW1,SW2,SW3,SW4) = Sum(96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223)

= Seq2'Seq1Seq0 + Seq2Seq1' + Seq2Seq0'

## Schematic Diagram

* 4-bit parallel register:

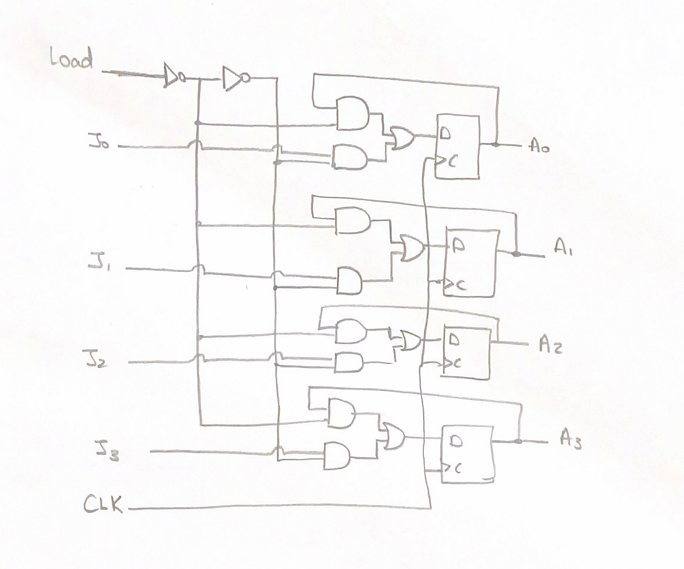


Figure 2: 4-bit parallel register

* 3-bit parallel register

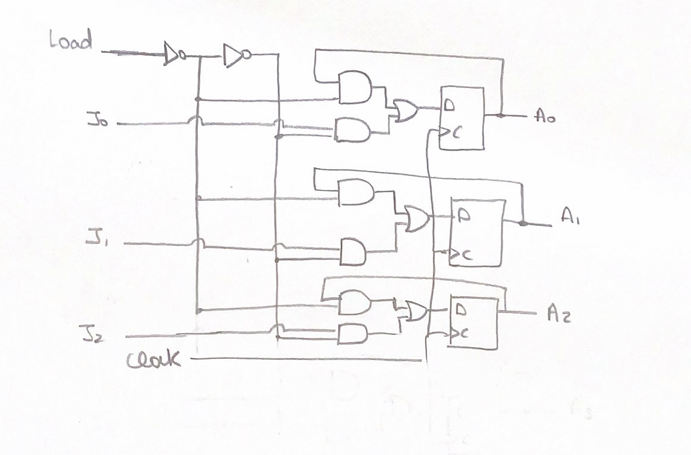


Figure 3: 3bit parallel register

* 555timer

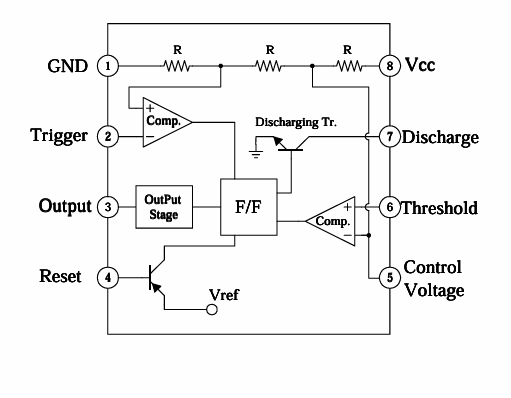


Figure 4: LM555

# Quartus design and analysis

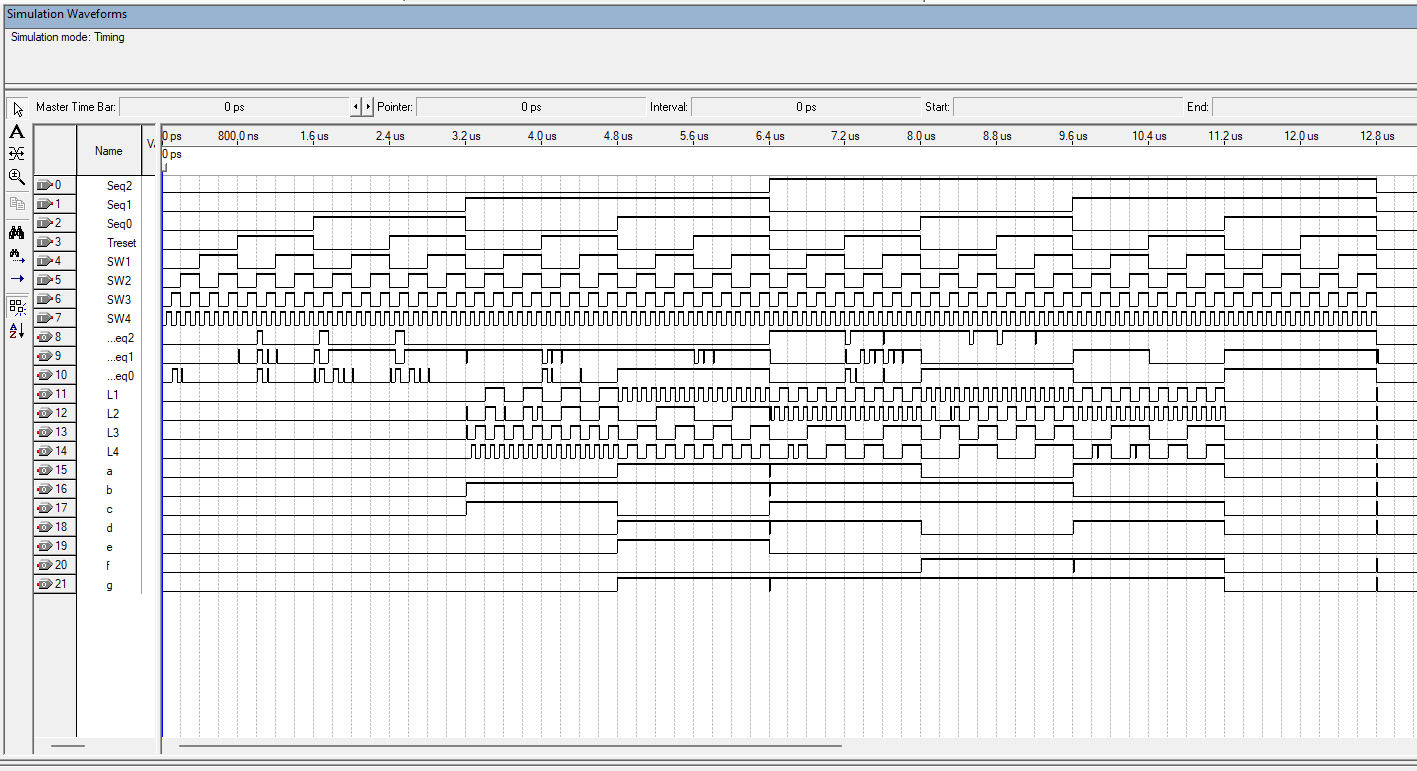


Figure 5:The Quartus Waveform Results

# Breadboard design and analysis

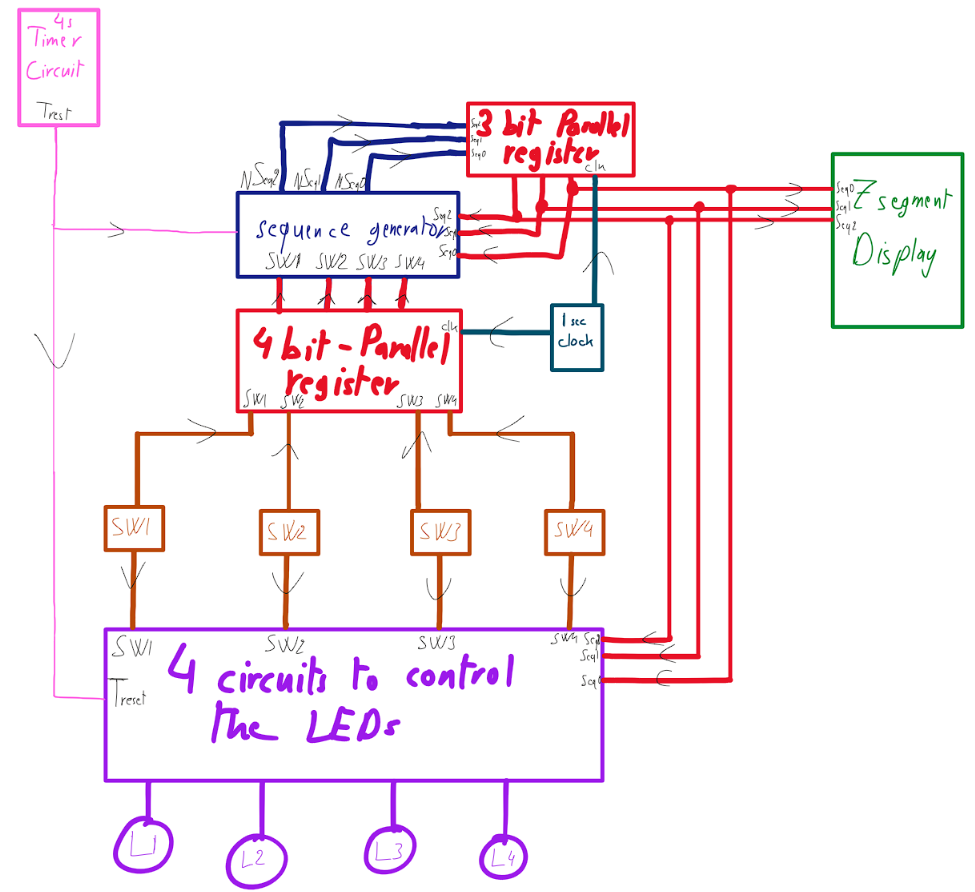


Figure 6: Circuit Diagram

* **4s Timer Circuit:** Produces a signal logic high (Treset) after 4 seconds timer and it resets, also it resets to count again when all switches are OFF. Initiates a sequence or reset of a system depending on which switch was last turned off. FSM timing and special tricks, such as those of Sequence 3 (removal of cap).
* **Sequence Generator:** has Switch states (SW1–SW4), the 3-bit sequence selector (Seq0, Seq1, Seq2), and Treset as inputs which is used to generate the next sequence is being used as output. Defined by which switch is turned off last and the last sequence used in the system and creates the correct sequence for the system to function next. Drives the main FSM logic for all operational modes (Sequences 1–4).
* **3-bit Parallel Register:** The sequence signals generated from the Sequence Generator are the input of the 3-bit Parallel Register and it saves that value each one second. Holds the value of the old sequence to be used in the system when it’s necessary.
* **7-Segment Display:** Shows sequence number (1 to 4).Enabling users to see and debug what is currently running. Used only while working with practice mode, potentially removed for performance mode.
* **1-second Clock:** Generates a continuous timing signal to specify when the 3-bit and 4-bit register is updated. Refreshes the sequence output on a scheduled regular interval and maintain order of the system.
* **4-bit Parallel Register:** Switch (SW1–SW4) as inputs. Serves as a temporary storage of the four switches' ON/OFF state and it saves each 1 second. Acts as a buffer for raw user input and for the logic control of the LEDs.
* **Switches (SW1–SW4):** Primary input users. Switches are the main input to control the LED circuit. These are manipulated by users to alter states of LEDs and indirectly cause sequence changes.
* **4 Circuits to Control LEDs:** Treset, sequence states, and switch status as inputs and drive LEDs L1–L4 as outputs. Central logic unit that determines which next LED is illuminated based on the current sequence and switch states. Carries out the true behavior and "Special Trick" of the system by executing sequence logic and cap tricks rules.
* **LEDs (L1–L4) Role:** The visual outputs of the system. Define the current status of all switches controlled by dynamic logic.

# Financial study, focusing on minimizing the cost of the design

Table 1: List of Required Components and Quantities

|  |  |  |
| --- | --- | --- |
| Item | Quantity | Prices |
| Soldering Iron | 1 | owned |
| Soldering Iron Wire | 1 | owned |
| Perforated Circuit Board / Breadboard | 2 | 3$ |
| Voltage Source | 1 | owned |
| Wires | 150 | owned |
| 555 Timer | 4 | 1$ |
| Resistors and Capacitors | 10 | owned |
| And | 10 | 3$ |
| Or | 10 | 4.5$ |
| XOR | 10 | 5$ |
| Not | 10 | 3$ |
| D flip-flop | 10 | 3.5$ |
| Toggle Switches | 4 | owned |
| LEDs(red, blue, yellow, green) | 4 | owned |
| 7-Segment Display | 1 | owned |

# Delay calculation

The input in the waveform ends at 12.8 µs while the last output ends at 12.812 µs, then the delay is 0.012 µs

# Power consumption analysis

* Voltage(V) is the supply voltage used
* Current (i) is the total current drawn by all components combined (measured in Amperes or milliamperes)

Table 2: Power Consumption Analysis

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Components** | **Quantity** | **Current(each)** | **Total Current** | **Power in Watts** |
| LEDs | 4 | 15 mA | 60 mA | 0.300 W |
| 555 Timer | 1 | 10 mA | 10 mA | 0.050 W |
| 74LS08 (AND) | 10 | 4 mA | 40 mA | 0.200 W |
| 74LS32 (OR) | 10 | 4 mA | 40 mA | 0.200 W |
| 74LS00 (NAND) | 10 | 4 mA | 40 mA | 0.200 W |
| 74LS86 (XOR) | 10 | 4 mA | 40 mA | 0.200 W |
| 74LS04 (NOT) | 10 | 4 mA | 40 mA | 0.200 W |
| 74LS74 (D Flip-Flop) | 10 | 4 mA | 40 mA | 0.200 W |
| 74LS138 (Decoder) | 5 | 4 mA | 20 mA | 0.100 W |
| 7-Segment Display | 1 | 70 mA | 70 mA | 0.350 W |

Therefore, on average, this circuit has a power consumption of 5V × (total current) = 5V × 0.4 A = 2 W.

# Problems faced during the project

* Use of a finite state machine (FSM) that properly respects sequences and transitions for all 4 switch states needs to be designed and tested carefully.
* Tracking numerous sequences and specific behaviors (such as the capless trick) introduced extra states and state transitions, which made logic bugs harder to debug.
* Correct 4-second accurate reset and capless switch operations were challenging to accomplish, especially when synchronized with FSM state changes.
* The 555-timer circuit required adjustment of values for resistors and capacitors for reproducible timing.
* The dynamic disabling and enabling of switches based on cap status needed extra logic and state transitions, which were hard to simulate and verify.
* Enabling switches to remain flappable even when capless (but without actually engaging the LED) introduced edge-case bugs.
* The minimization of output functions by Karnaugh map or Boolean algebra used to be extremely time-consuming and error-prone.
* Some simplified expressions did not behave as expected during Quartus simulation initially and required redesign.
* Hardware Implementation Difficulties Breadboard wiring tended to develop loose contacts or shorts, which resulted in impaired performance of LEDs and switch responses.
* Soldering of components neatly, particularly with numerous wires and ICs, was tricky and needed to be planned carefully.
* There was a risk of heat damage to ICs or switches while soldering. Shortages of certain ICs or 7-segment displays occasionally required substitutes or redesign.
* Erroneous or inappropriate values for resistors affected current flow through LEDs, altering brightness. Performance of circuits that are simulated using Quartus didn't always match real hardware performance due to physical phenomena (bounce, noise, delay).
* Switch flipping or capless behavior was harder to reproduce with digital design tools.
* Packaging and Assembly: Keeping it modular and neat with a clean enclosure for the final project proved challenging with layout and wire organization.

# Key design points that present advantages over alternative designs

Outstanding Design Features Which Offer Advantages Over Alternative Designs

The design of the logic-controlled board incorporates some thoughtful choices that yield functional or practical advantages over simpler or less modular implementations. They include:

1. Modular System Architecture

* Advantage: Dividing parts (sequence generator, register banks, control circuits) into modular blocks eases debugging, simulation, and hardware construction.
* Alternative: A monolithic architecture would be more difficult to test and respond to changes.

2. Application of Finite State Machine (FSM) to Control Logic

* Benefit: Using FSM allows structured control of complex sequences and behaviors (like the capless trick).
* Alternative: Hardwired logic would be inflexible and far harder to extend or change.

3. Timer-Based Dynamic Behavior

* Benefit: Utilization of 4-second and 1-second timers adds interactive and timed behavior, making such things as automatic sequence detection and capless trick possible.
* Alternative: Static control or manual reset would decrease interactivity and user engagement.

4. Practice vs Performance Mode

* Advantage: The removable 7-segment display allows the user to switch between a learning/debugging mode and a performance mode to contribute to the illusion.
* Alternative: Having the sequence displayed constantly would dispel the illusion for the audience.

5. Cap Detection Feature

* Benefit: Provides an interactive "trick" that involves users and illustrates conditional logic handling.
* Alternative: Straight switch-to-LED mapping doesn't carry this level of complexity and innovation.

# Conclusion

The Logic-Controlled Board project successfully demonstrated the direct application of digital logic design principles, using combinational and sequential circuits with FSMs to create a flexible and interactive system. Depending on the input, it runs through sequences that turn several lamps on and off. With features such as the capless switch trick and reset behavior, the project highlighted the considerable amount of flexibility and complexity possible in digital logic.

We gained much practical experience in circuit analysis, FSM design, logic simplification, and simulation of hardware through Quartus, as well as the physical implementation of circuits via breadboarding and soldering. In doing so, we encountered and dealt with issues of timing, correctness of logic, and hardware stability, which further enhanced our debugging and problem-solving skills.

Finally, the project served to bridge the gap between theoretical concepts and actual digital systems, reinforcing our knowledge of how logic circuits govern daily technologies such as traffic lights, vending machines, and interactive games. It has prepared us for more complex design assignments and laid stress on the importance of accuracy, planning, and a high degree of teamwork in the manufacture of digital systems

# References

1. Altera Quartus II User Manual.
2. 74LS74 Datasheet.
3. 5161AS Datasheet.
4. DM74LS32 Datasheet.
5. DM74LS138 Datasheet.
6. DM7408 Datasheet.
7. DM7486 Datasheet.
8. LM555 Datasheet.
9. M54HCH04 Datasheet.
10. MM74HC00 Datasheet.