**Lebanese American University**



***COE 322– Logic Design Lab***

***Project***

Jad Chamoun 202301352

Pietro Ziade 202305297

Raphael Khoury 202301561

Joanne Rizkallah 202000840

12/5/2025

Abstract

This project manual presents the design and implementation of a Logic-Controlled Board, an active switching system containing four switches and four colored lamps. The system reconfigures the lamp activation sequence based on the last switch turned off, integrating concepts from Boolean algebra, combinational and sequential logic, state machines, and circuit optimization.

The project also includes a feature where removing a switch cap temporarily disables its functionality. The system operates using a finite state machine (FSM) for dynamic behavior control, a reset timer, and a locking mechanism. The manual details the circuit design, hardware implementation, and software simulation using Quartus, along with guidelines for packaging, documentation, and grading criteria.

Table of Contents

[Abstract 1](#_Toc197923421)

[Introduction 5](#_Toc197923422)

[Equipment 6](#_Toc197923423)

[Analysis 7](#_Toc197923424)

[Inputs and Outputs 7](#_Toc197923425)

[State Diagram 7](#_Toc197923426)

[Truth Table 9](#_Toc197923427)

[Expressions 9](#_Toc197923428)

[4 second Reset Timer 15](#_Toc197923429)

[Fast Clock 17](#_Toc197923430)

[Paper Design 18](#_Toc197923431)

[Quartus Design and Analysis 23](#_Toc197923432)

[Breadboard Design and Analysis 30](#_Toc197923433)

[Financial Study 37](#_Toc197923434)

[Delay Calculation 38](#_Toc197923435)

[Power Consumption Analysis 39](#_Toc197923436)

[Problems Faced 40](#_Toc197923437)

[Strengths of Our Project 41](#_Toc197923438)

[Conclusion 42](#_Toc197923439)

Table of figures

[Figure 1: State Diagram 9](#_Toc197924557)

[Figure 2: k-map LED1 at BOOT state 11](#_Toc197924558)

[Figure 3: k-map LED2 at BOOT state 11](#_Toc197924559)

[Figure 4: k-map LED4 at BOOT state 11](#_Toc197924560)

[Figure 5: k-map LED3 at BOOT state 11](#_Toc197924561)

[Figure 6: k-map LED4 at Sequence 2 state 12](#_Toc197924562)

[Figure 7: k-map LED2 at Sequence 3 state 13](#_Toc197924563)

[Figure 8: k-map LED3 at Sequence 1 state 14](#_Toc197924564)

[Figure 9: k-map LED4 at Sequence 3 state 15](#_Toc197924565)

[Figure 10: Square wave of 555-timer 17](#_Toc197924566)

[Figure 11: 555-timer general circuit 17](#_Toc197924567)

[Figure 12: Hand-drawn diagram for A1 19](#_Toc197924568)

[Figure 13: Hand-drawn diagram for A2 20](#_Toc197924569)

[Figure 14: Hand-drawn diagram for A3 20](#_Toc197924570)

[Figure 15: Hand-drawn diagram for LED1 21](#_Toc197924571)

[Figure 16: Hand-drawn diagram for LED2 21](#_Toc197924572)

[Figure 17: Hand-drawn diagram for LED3 22](#_Toc197924573)

[Figure 18: Hand-drawn diagram for LED4 22](#_Toc197924574)

[Figure 19: Switch change detection diagram 23](#_Toc197924575)

[Figure 20: NE555 timer diagram for 23](#_Toc197924576)

[Figure 21: Final circuit diagram 24](#_Toc197924577)

[Figure 22: Quartus design for overall circuit 25](#_Toc197924578)

[Figure 23: Quartus block diagram for A1 26](#_Toc197924579)

[Figure 24: Quartus block diagram for A2 and A3 27](#_Toc197924580)

[Figure 25: Quartus block diagram for LEDs 28](#_Toc197924581)

[Figure 26: Quartus block diagram for 8:1 multiplexer using 2:1 multiplexers 28](#_Toc197924582)

[Figure 27: Quartus vector waveform showing the switching between sequences and the current state 29](#_Toc197924583)

[Figure 28: Quartus vector waveform showing the simulation of the overall circuit 30](#_Toc197924584)

[Figure 29: Quartus block diagram for switch change detection 30](#_Toc197924585)

[Figure 30: Mealy state diagram for switch change detection 31](#_Toc197924586)

[Figure 31: Breadboard design for A1 32](#_Toc197924587)

[Figure 32: Breadboard design for A2 33](#_Toc197924588)

[Figure 33: Breadboard design for A3 34](#_Toc197924589)

[Figure 34: Breadboard design for LED1 34](#_Toc197924590)

[Figure 35: Breadboard design for LED2 35](#_Toc197924591)

[Figure 36: Breadboard design for LED3 36](#_Toc197924592)

[Figure 37: Breadboard design for LED4 37](#_Toc197924593)

[Figure 38: Breadboard design for switch detection circuit 38](#_Toc197924594)

[Figure 39: Mealy state diagram for switch change detection circuit 38](#_Toc197924595)

[Figure 40: Delay in Quartus 41](#_Toc197924596)

Table of Tables

[Table 1: States Representation 9](#_Toc197924550)

[Table 2: Multiplexer inputs for LEDs 14](#_Toc197924551)

[Table 3: Multiplexer inputs for A2 and A3 bits of the next states 15](#_Toc197924552)

[Table 4: Summary of total number of IC chips for LEDs and states 15](#_Toc197924553)

[Table 5: Mealy state table for switch change detection circuit 30](#_Toc197924554)

[Table 6: Bill of materials showing materials and total costs 39](#_Toc197924555)

[Table 7: Power consumption summary 41](#_Toc197924556)

Introduction

This project realizes magic tricks using LEDs and switches. Our project utilizes basic gates, a 555 timer that acts as a control clock, and flip flops. The operation will switch between 7 states, depending on which switch was learned last: Boot, Lock, Sequence 1, Sequence 2, Sequence 3, Sequence 4, and special trick.

Equipment

**These were the used equipment in this project:**

1- Breadboard (6)

2- ICs’ Datasheet

3- Jumping Wires

4- Types of ICs:

74LS02, 74LS04, 74LS08, 74LS32, 74LS86, 74LS74, 74LS151, 555 Timer

5- Jumping wires

6- Capacitors: 10nF, 100

7- Resistors: 220, 100, 250

Analysis

The first step in the project was to come up with a truth table that represents the desired behavior of the circuit. To begin with, we defined the inputs and outputs. Next, we drew the state diagram. Once we confirmed the choices for the pinouts, we proceeded to fill in the truth table. Once the truth table was completed and verified multiple times, the expressions were derived and the circuits simulated on Quartus and Circuitos before final hardware implementation.

## Inputs and Outputs

Inputs:

* Switches 1->4
* Timer reset
* Current states

For the current states, since we have 7 states, 3 bits are enough to represent them all.

For the outputs:

* Next states (3 bits)
* LEDs (4 bits)

## State Diagram

For our project, we decided to simulate a specific order of events so that the truth table is simplified while maintaining the integrity of the magic trick. This decision came after attempting to make the circuit work in any possible sequence of tricks, which was quite complex. Since we are the magicians, we will be following the following diagram:

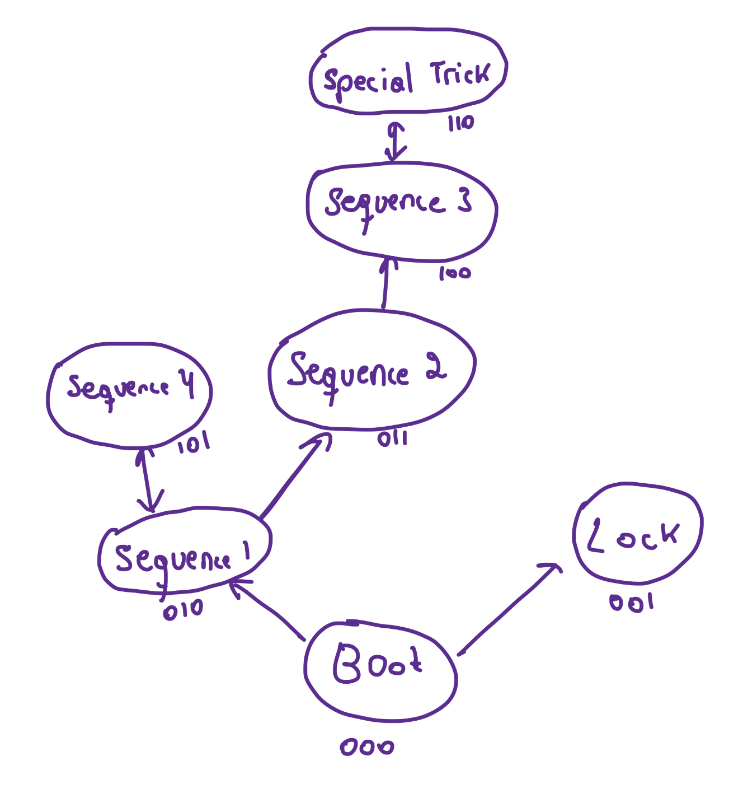


Figure 1: State Diagram

First, we will start at the boot state and show the viewer how the boot state works. After removing the battery, turning switch 2 ON only, and reinserting the battery, the state will transition to Lock. Next, we will unlock the device by restarting the system with switch 2 OFF.

To transition to sequence 1, we will turn ON the switch 1, wait for the 4 second timer to pass, and transition to sequence 1. During sequence 1, we will show how turning on switch 4 will turn on LED1, then switch 3 turn on LED2, then switch 2 turn on LED3, and switch 1 LED4. We will turn off the LEDs in that order, with switch 4 being the last switch to be ON. After waiting for 4 seconds, the state will transition to sequence 1.

To transition from sequence 1 to sequence 2, we turn off last Switch 2.

In sequence 2, the order will be:

* Switch 3 🡪LED2
* Switch 2 🡪 LED3
* Switch 1 🡪 LED4
* Switch 4 🡪 LED1

After turning them off in the same order, the state will transition to sequence 3. In sequence 3, the order will be:

* Switch 2 🡪 LED3
* Switch 3 🡪 LED4
* Switch 1 🡪 LED1
* Switch 4 🡪 LED2

To transition to the special trick, the switch 2 will be turned OFF last. Thus, the special trick will take place.

Since there are 7 states, the table below summarizes the representations (in 3 bits) of all states:

Table 1: States Representation

|  |  |
| --- | --- |
| **State** | **Representation** |
| Boot | 000 |
| Lock | 001 |
| Sequence 1 | 010 |
| Sequence 2 | 011 |
| Sequence 3 | 100 |
| Sequence 4 | 101 |
| Special Trick | 110 |

## Truth Table

The truth table was written in excel as one large table. For a neater report, we have uploaded the excel sheet on [OneDrive](https://lauedu74602-my.sharepoint.com/:f:/g/personal/joanne_rizkallah_lau_edu/EhQVtOKv3LBOppw4AFjOfBMBM2tG1nUMoeC07cQlOAZU4w?e=anEUWy). I1, I2, I3, I4 are switches 1,2,3,4, D1, D2, D3 are the current states, A1, A2, A3 are the next states, and O1, O2, O3, O4 are the LEDs. At the right of each state table are the switching sequences we will follow in the demo.

## Expressions

#### LEDs

For the LEDs, we utilized an 8:1 multiplexer for each LED that will direct the correct output to the corresponding LED, based on the input switches and the timer reset input (5 inputs). The select lines would be the current states. This makes sense and greatly simplifies our design.

To come up with the equations, we used multiple k-maps:

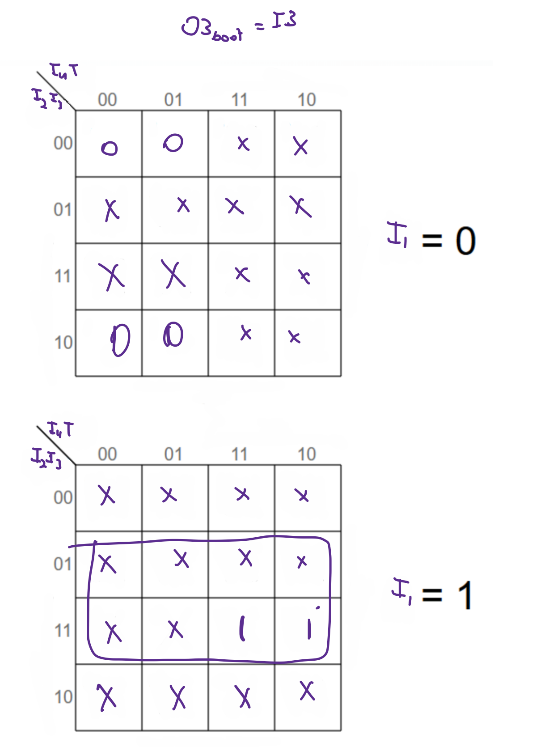
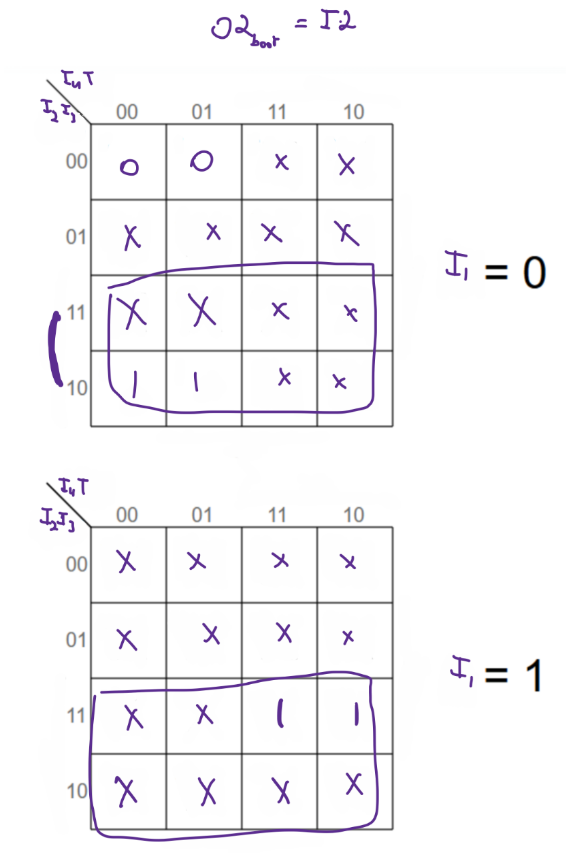
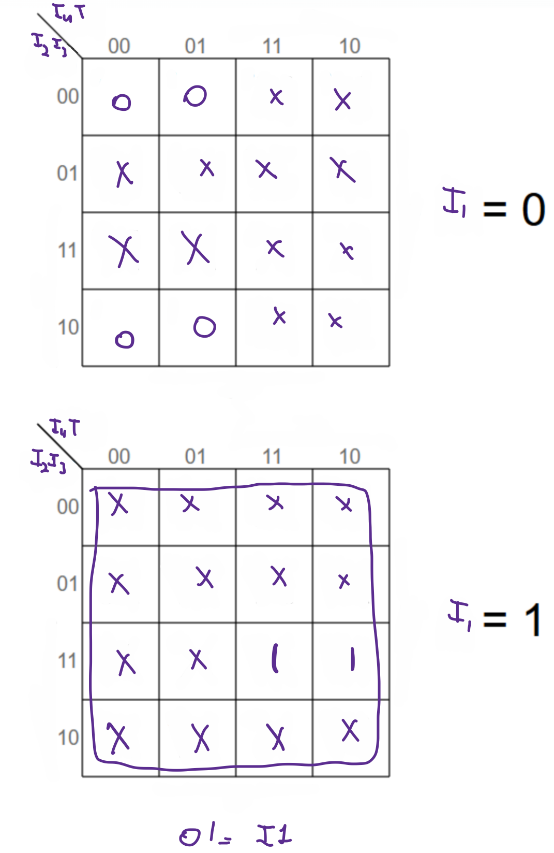
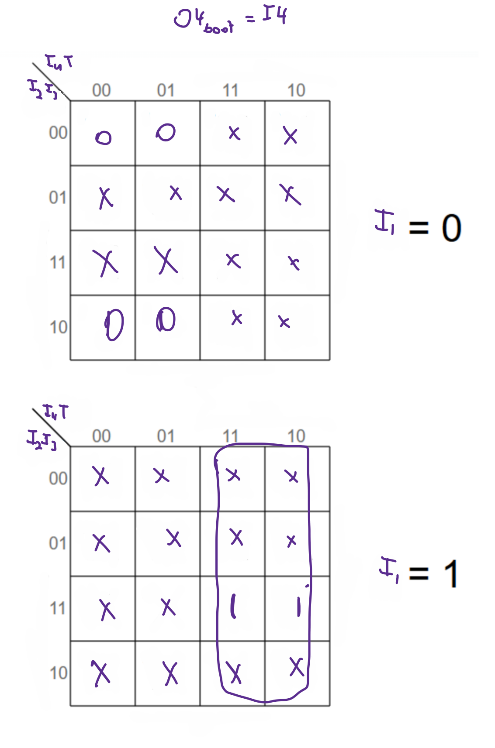
For the boot state (O1=LED1, O2=LED2, O3=LED3, O4=LED4, and T=Treset):

Figure 2: k-map LED1 at BOOT state

Figure 3: k-map LED2 at BOOT state

Figure 4: k-map LED4 at BOOT state

Figure 5: k-map LED3 at BOOT state



After following this method, we noticed that the LEDs are only function of the switches, which makes sense since the selects of the multiplexers used to manipulate the LEDs are the ones dependent on Treset. Thus, Treset will be integrated in the circuit of the multiplexers. For the rest of the multiplexer inputs, we used 4 input k-maps when needed that involve the switches only.

For the rest of LED1 (O1):

* Lock state: O1=I1 by simply observing the truth table
* Sequence 1: O1= I2 + I4 by simply observing the truth table
* Sequence 2: We drew a k-map to derive the equation O1= I4+I2.I3’

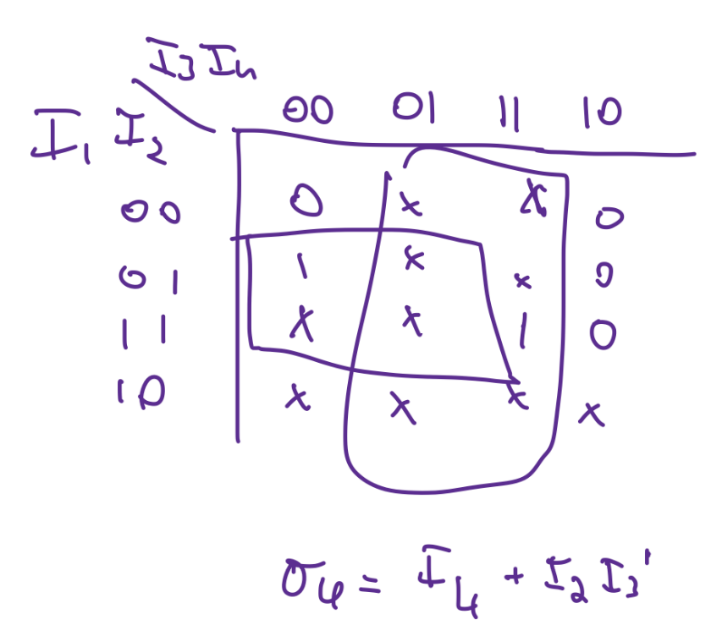


Figure 6: k-map LED4 at Sequence 2 state

* Sequence 3: O1=I1 through observation
* Sequence 4: O1 = I4 through observation
* Special trick: O1=0

For LED2 (O2):

* Boot: O2=I2
* Lock: O2 =I2
* Sequence 1: I3
* Sequence 2: I3
* Sequence 3: I4+I2’.I3 using k-maps:

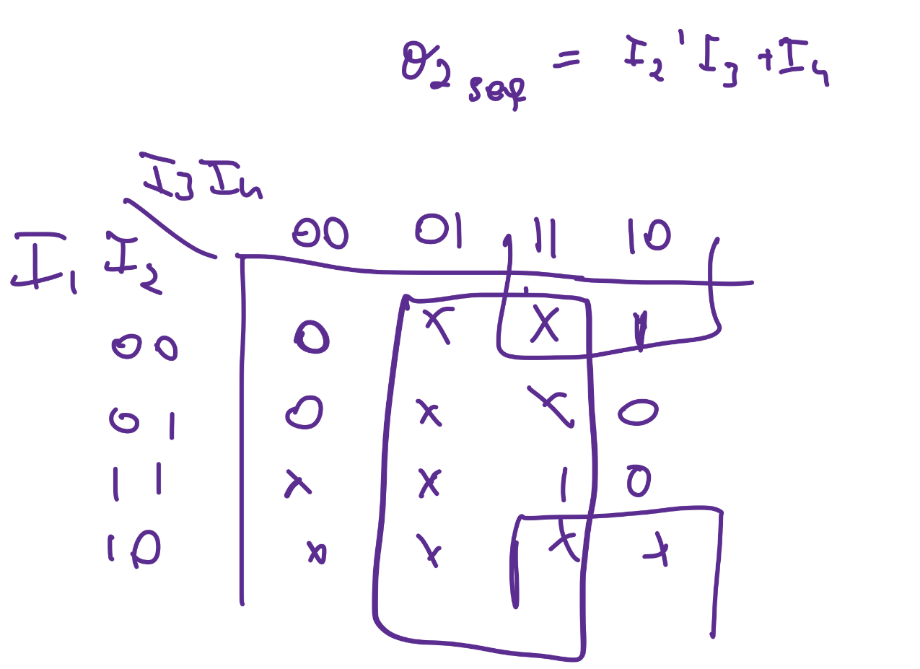


Figure 7: k-map LED2 at Sequence 3 state

* Sequence 4: I3
* Special Trick: 0

For LED3 (O3):

* Boot: I3
* Lock: I3
* Sequence 1: I2.I3 using k-maps:

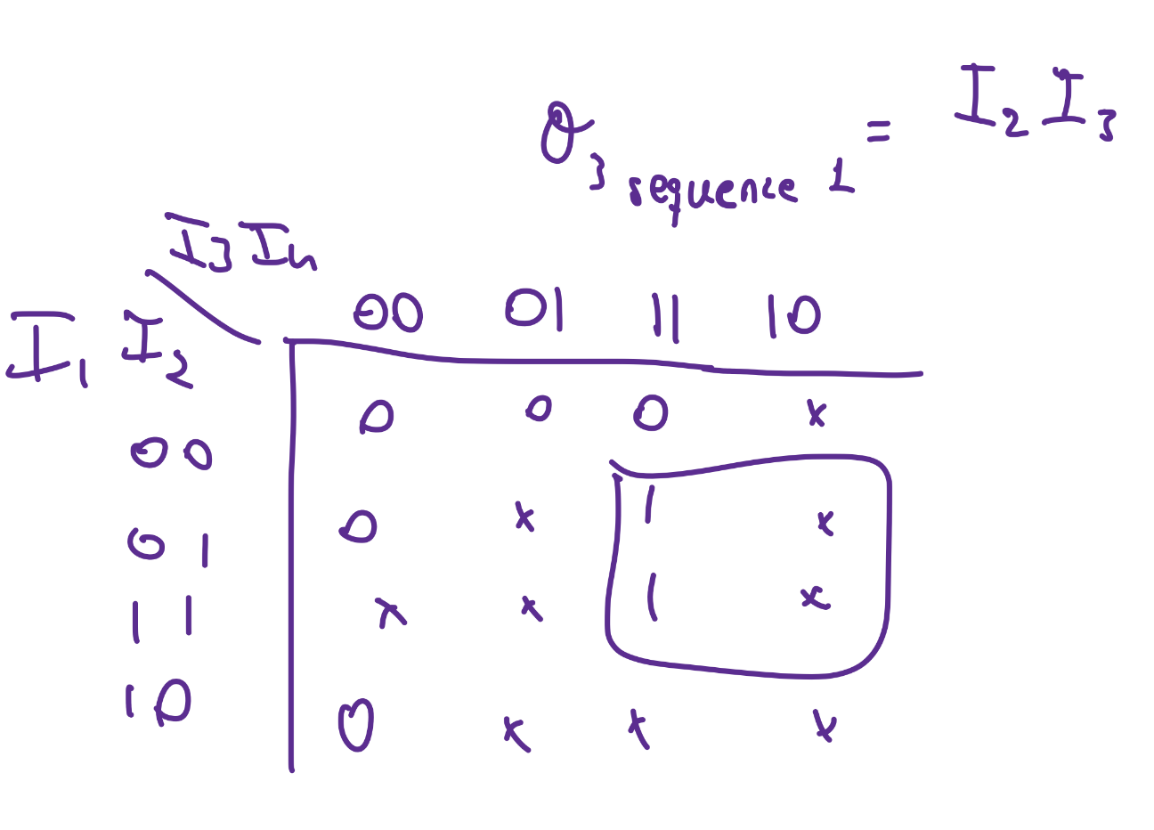


Figure 8: k-map LED3 at Sequence 1 state

* Sequence 2: I2.I3 (similar to sequence 2)
* Sequence 3: I2
* Sequence 4: I2
* Special trick: 1 or I2

For LED4:

* Boot: I4
* Lock: I4
* Sequence 1: I1
* Sequence 2: I1
* Sequence 3: I2.I3 using k-maps:

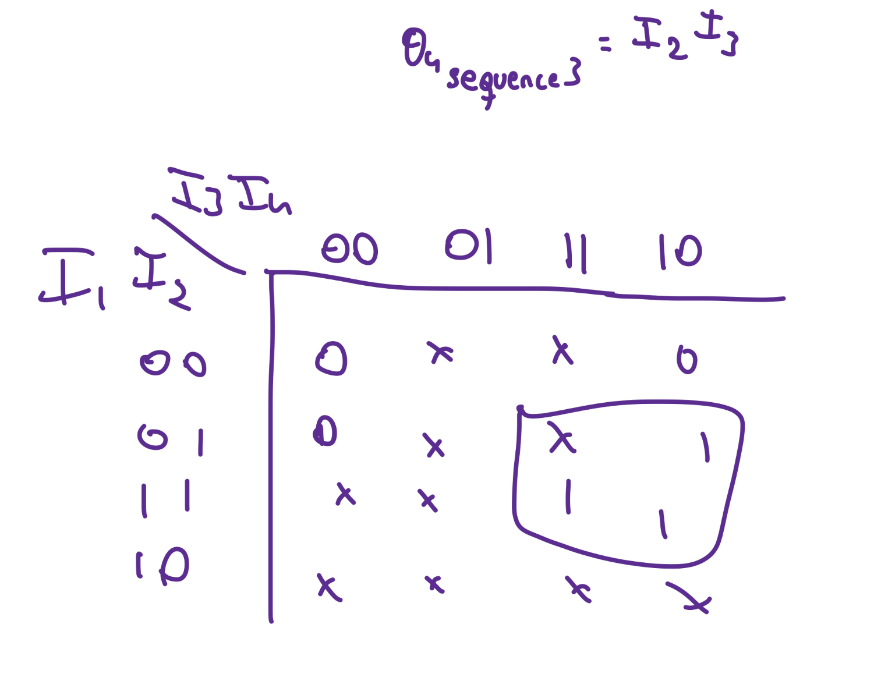


Figure 9: k-map LED4 at Sequence 3 state

* Sequence 4: I1
* Special trick: 0

As a result, the LEDs need a total of 8 gates: 3 AND gates, 3 OR gates, and 2 NOT gates. Concerning chips, we used 1 NOT chip (74LS04), 1 AND chip (74LS08), 1 OR chip (74LS32).

The table below summarizes the input to each multiplexer for each LED:

Table 2: Multiplexer inputs for LEDs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Multiplexer Input** | **LED1** | **LED2** | **LED3** | **LED4** |
| I0 (Boot) | SW1 | SW2 | SW3 | SW4 |
| I1 (Lock) | SW1 | SW2 | SW3 | SW4 |
| I2 (Seq 1) | SW2+SW4 | SW3 | SW2.SW3 | SW1 |
| I3 (Seq 2) | SW4+SW2.SW3’ | SW3 | SW2.SW3 | SW1 |
| I4 (Seq 3) | SW1 | SW4+SW2’.SW3 | SW2 | SW2.SW3 |
| I5 (Seq 4) | SW4 | SW3 | SW2 | SW1 |
| I6 (Special Trick) | 0 | 0 | 1 or SW2 | 0 |
| I7 (not used) | 0 | 0 | 0 | 0 |

#### Next States

For the states, we derived A1 using k-maps and Boolean algebra:

We further modified A1 to include NOR gates instead of 2 NOT gates and an OR gate.

For A2 and A3, we used an 8:1 multiplexer as well to simplify the expressions, with the select lines being the current states.

The table below summarizes the inputs to the 8:1 multiplexers:

Table 3: Multiplexer inputs for A2 and A3 bits of the next states

|  |  |  |
| --- | --- | --- |
| **Multiplexer Inputs** | **A2** | **A3** |
| I0 | SW1+SW2’ | SW2.SW4’ |
| I1 | 0 | 1 |
| I2 | SW4’+T’+SW3 | T.(SW3’.SW4+SW2.SW4’) |
| I3 | SW3’+T’+SW1 | SW3’+SW2+T’ |
| I4 | SW2.SW3’.T | 0 |
| I5 | SW1.SW2’.T | SW2+SW1’+T’ |
| I6 | T’+SW3 | 0 |
| I7 | 0 | 0 |

Each of these equations was obtained using a 5-input k-map similar to the part before for the LEDs.

As a result, A2 and A3 states require: 9 OR gates, 6 AND gates, and 5 NOT gates (some of which are used from before), and 2 D-flip flops.

A1 state requires: 2 4-input AND gate, 2 2-input AND gate, and 4 OR gates.

The total number of IC chips for the states would be summarized as:

Table 4: Summary of total number of IC chips for LEDs and states

|  |  |  |
| --- | --- | --- |
| **Gate** | **Total number of gates** | **Total number of IC chips** |
| AND | 3+6+2 | 3 |
| OR | 9+4+3 | 4 |
| NOT | 5 | 1 |
| 8:1 MUX | 6 | 6 |
| 4-input AND | 2 | 1 |

The 555-timer was used to transition to different states and plugged into the clock of the 3 D-flip flops used for the next states A1, A2, and A3. In the circuit that detects a switch being toggled, a fast clock of 0.3 seconds ON and 0.15s OFF was used.

## 4 second Reset Timer

In general, the 555 timer has 3 modes of operation: monostable, bistable, and astable. An astable circuit has no stable state – hence the name “astable”. The output continually switches states between high and low, producing a square wave. This is the type of output desired for our project to activate the D flip flops of the states.

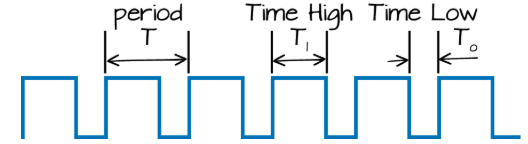


Figure 10: Square wave of 555-timer

The desired behavior is a high time of T1=4 seconds and a low time T0=4 seconds, with a reset pin that is activated once a switch is toggled. First, we looked at the circuit itself:

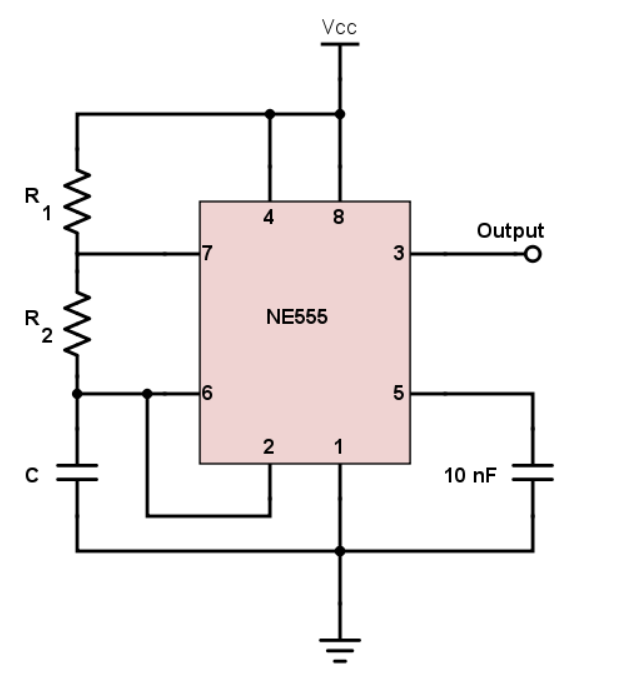


Figure 11: 555-timer general circuit

Through research, we were able to come up with the equations for T1 and T0

Using , we are able to satisfy the above conditions.

## Fast Clock

For the fast clock, the same equations were used to calculate the resistors and capacitors:

# Paper Design

The state table was designed using excel sheet and is located inside the [OneDrive folder](https://lauedu74602-my.sharepoint.com/:f:/g/personal/joanne_rizkallah_lau_edu/EhQVtOKv3LBOppw4AFjOfBMBM2tG1nUMoeC07cQlOAZU4w?e=anEUWy).

For A1, we kept the circuit as a simple combinational circuit. Note that for all drawings in this section the switches are labelled as I1, I2, I3, I4, the current states as D1, D2, D3, and the timer reset as T-reset.

A diagram of a circuit

Description automatically generated

Figure 12: Hand-drawn diagram for A1

For A2 and A3, we used an 8:1 multiplexer for each with the current states D1, D2, and D3 as select lines.

A diagram of a circuit board

Description automatically generated

Figure 13: Hand-drawn diagram for A2

A diagram of a circuit board

Description automatically generated

Figure 14: Hand-drawn diagram for A3

The same methodology was used for the LEDs, however, inputs were now a function of the switches I1, I2, I3, I4 only.

A drawing of a circuit board

Description automatically generated

Figure 15: Hand-drawn diagram for LED1

A diagram of a circuit board

Description automatically generated

Figure 16: Hand-drawn diagram for LED2

A drawing of a circuit board

Description automatically generated

Figure 17: Hand-drawn diagram for LED3

A diagram of a circuit board

Description automatically generated

Figure 18: Hand-drawn diagram for LED4

A drawing of a diagram

Description automatically generated

Figure 19: Switch change detection diagram

For the 555-timer used to simulate a high signal for 4 seconds and a low signal for 4 seconds, the below circuit was used.

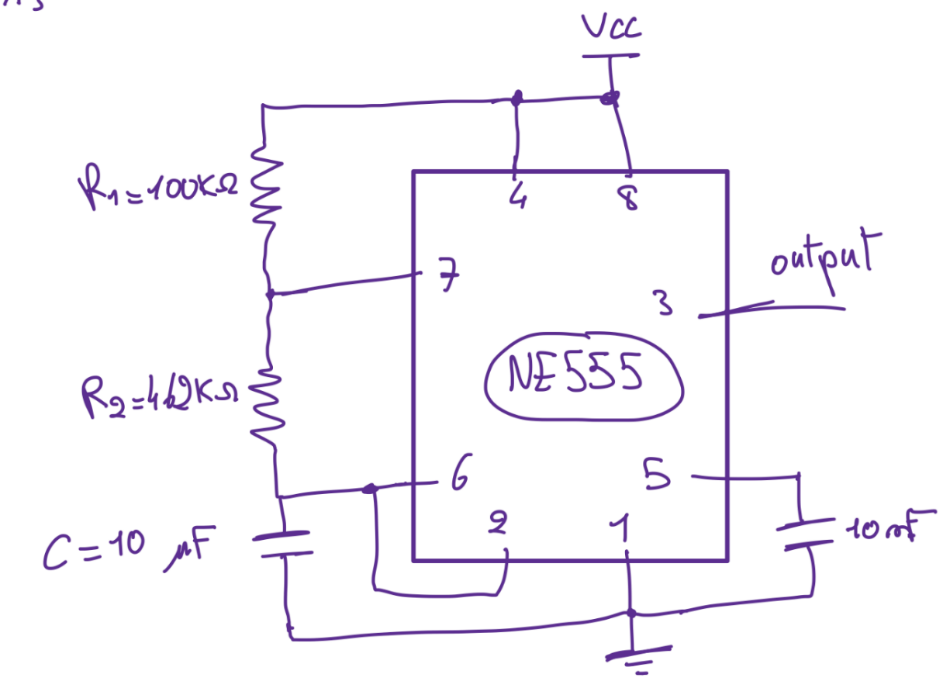


Figure 20: NE555 timer diagram for

The final circuit combined all the previous blocks.

A diagram of a led circuit

Description automatically generated

Figure 21: Final circuit diagram

# Quartus Design and Analysis

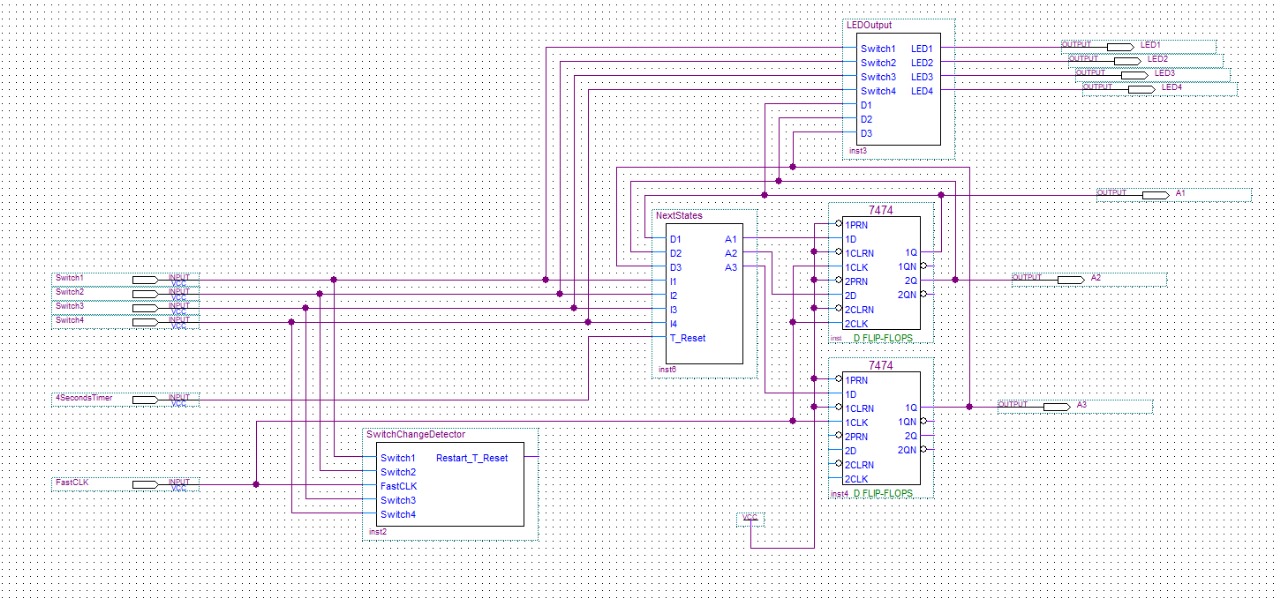


Figure 22: Quartus design for overall circuit

In this design, we implemented our circuit using different blocks to simplify and clarify the circuits.

The 555timer was not found so we did not use it, instead we used the signal 4SecondsTimer input to be able to change sequences. The fastCLK is used for the D Flip-Flops.

A1, A2 and A3 outputs are the current state shown in the waveform.



Figure 23: Quartus block diagram for A1

This circuit implements the equation of the first bit (MSB) of the states (first part).

A diagram of a computer

Description automatically generated

Figure 24: Quartus block diagram for A2 and A3

This circuit is the second part of the previous circuit; it implements the equations of the second and third bits of the states. The equations were based on the state table of our design. The outputs of this circuit were used as inputs to the D Flip-Flops to determine the next state in the actual circuit. Then the output of the D Flip-Flops were the inputs for this circuit. We used an 8:1 MUX implemented by us (as shown later).

As you can see the MSB for the state was not implemented using a MUX since we did not have an additional IC MUX.

A diagram of a computer circuit

Description automatically generated

Figure 25: Quartus block diagram for LEDs

This circuit implements the equations for the 4 LEDs, using gates and four 8:1 MUX.

A diagram of a circuit board

Description automatically generated

Figure 26: Quartus block diagram for 8:1 multiplexer using 2:1 multiplexers

This is the 8:1 MUX we designed using two 4:1 MUX and one 2:1 MUX. (We used an 8:1 MUX in the physical circuit)

A screenshot of a computer

Description automatically generated

Figure 27: Quartus vector waveform showing the switching between sequences and the current state

This is the waveform simulation of the circuit. It shows the transition from BOOT state to sequence 1, it shows the behavior of the circuit when we follow the sequence of switching in the demo, which is identical to the state table. A1, A2 and A3 are the current state that shows which sequence we are in.

A screenshot of a computer

Description automatically generated

Figure 28: Quartus vector waveform showing the simulation of the overall circuit

This is the waveform simulation of the overall circuit. It shows the transition from BOOT state to Lock State, it shows the behavior of the circuit when we change the switches. Clearly showing that the lock state works normally. A1, A2 and A3 are the current state that shows which sequence we are in.

In these waveforms, we used fastCLK input as an input to the D Flip-Flops, the 4SecondTimer input is the output of the 555timer, we used it to transition to next sequences.

A diagram of a computer circuit

Description automatically generated

Figure 29: Quartus block diagram for switch change detection

This circuit was used to detect a change in the switches; it was also implemented on Circuitos (see [Breardboard design section](#_Breadboard_Design_and)).

Whenever there is a change in any switch, the circuit detects it and sends a high spike output (0 -> 1 -> 0). The D Flip-Flops were used to save the state of the switch and based on it send a signal.

A diagram of a molecule

Description automatically generated

Figure 30: Mealy state diagram for switch change detection

Each switch has its own implementation of the state diagram.

Table 5: Mealy state table for switch change detection circuit

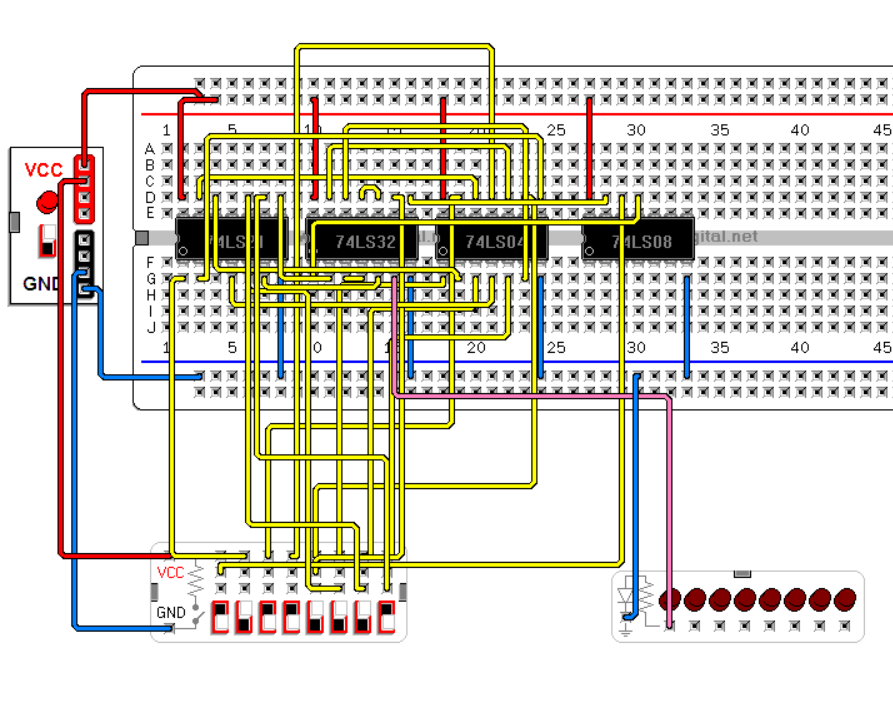
|  |  |  |  |
| --- | --- | --- | --- |
| **Current State** | **Switch** | **Next State** | **Output** |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

The outputs were ORed together to make a single signal. This signal will reset the timer by using it in the reset pin of the 555timer chip. It will also be ANDed with the T\_Reset Signal of the 555-timer and will be used as the 4SecondsTimer input.

# Breadboard Design and Analysis

Everything realized in Quartus was designed on the breadboard exactly as is, except for the 555 timers since it wasn’t available in Quartus and the 8:1 multiplexer. Each component (A1, A2, A3, LED1, LED2, LED3, LED4, then the entire circuit) will be shown separately on the breadboard.

For A1, recall that we implemented it using basic gates:

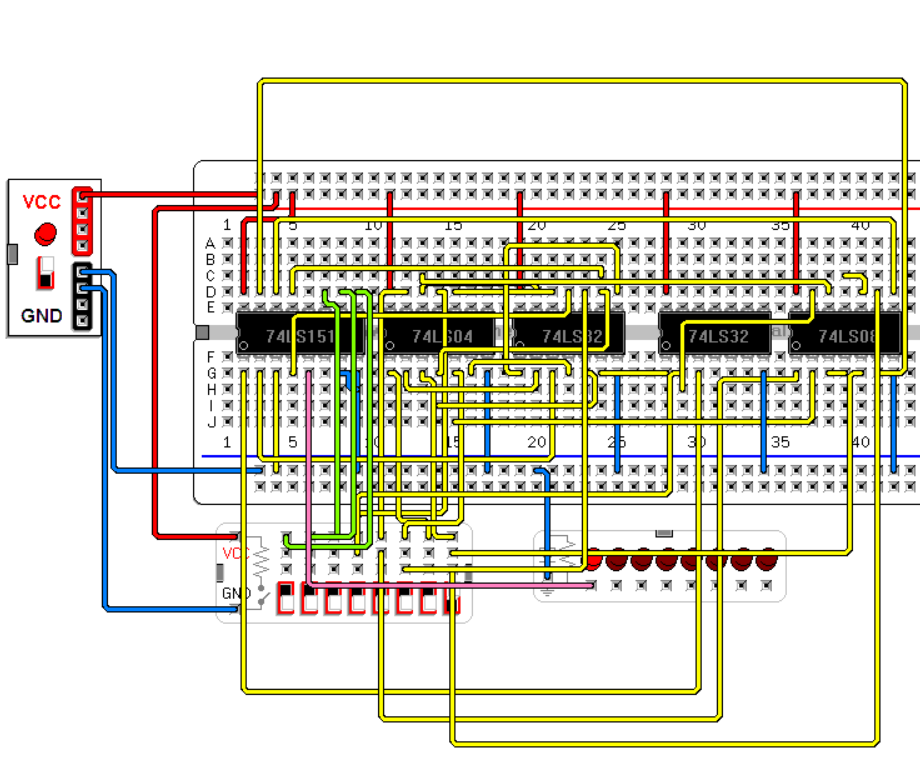


A1

D1 D2 D3 I1 I2 I3 I4 T

Figure 31: Breadboard design for A1

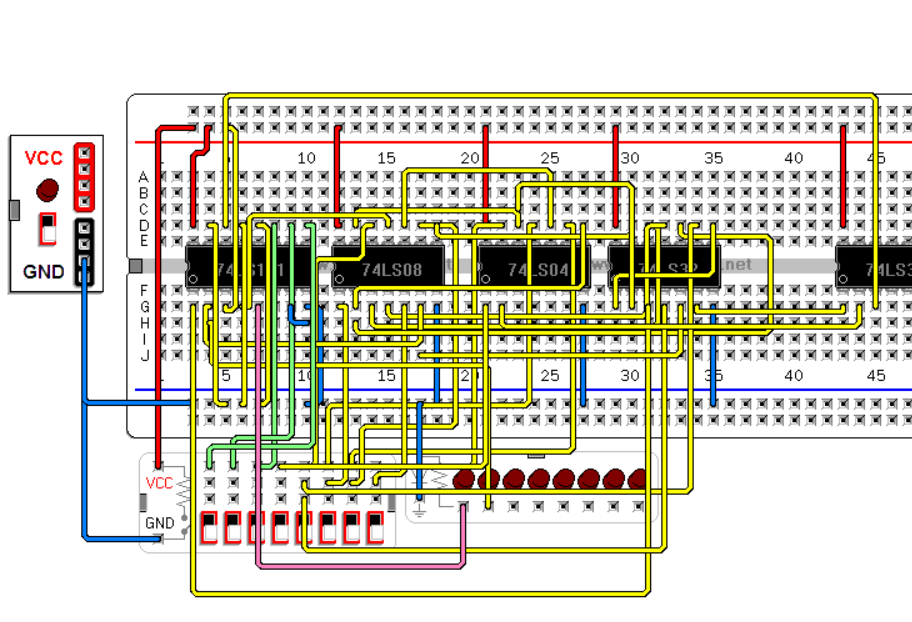
For A2, recall that we used the 8:1 multiplexer with current states as select lines and combinational logic function of the switches and reset timer as inputs:



D1 D2 D3 I1 I2 I3 I4 T

Figure 32: Breadboard design for A2

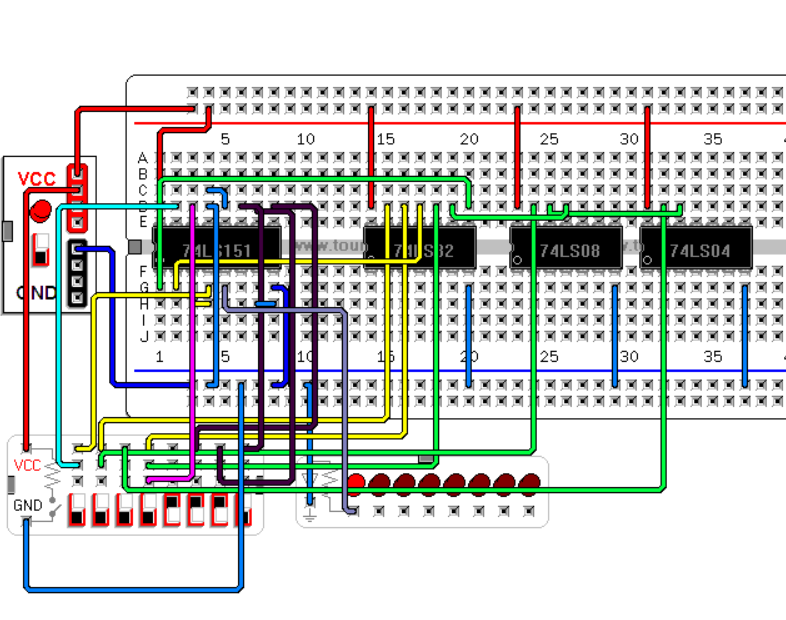
For A3. recall that we use the 8:1 multiplexer with current states as select lines and combinational logic function of the switches and reset timer as inputs:



D1 D2 D3 I1 I2 I3 I4 T

Figure 33: Breadboard design for A3

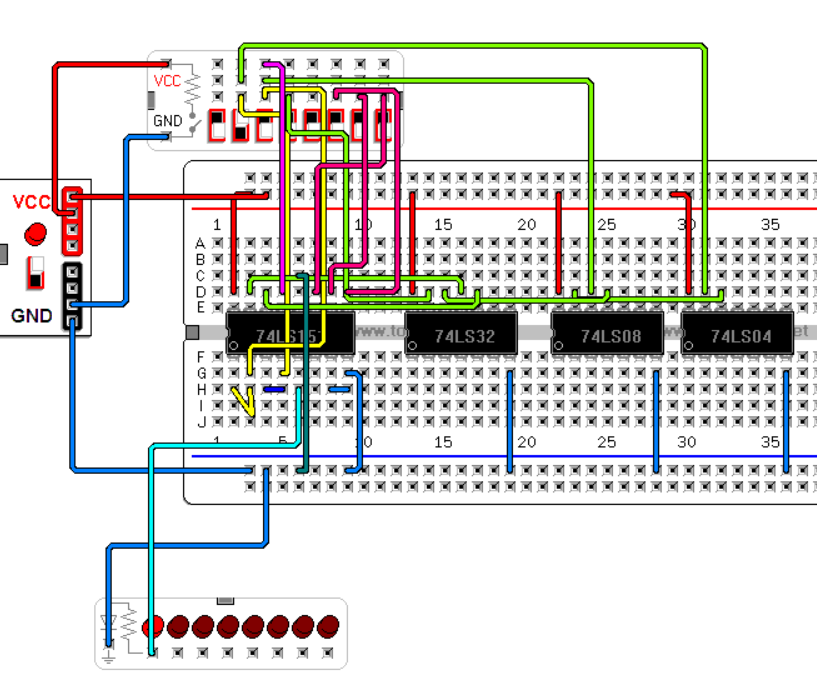
For LED1:



I1 I2 I3 I4 D1 D2 D3

Figure 34: Breadboard design for LED1

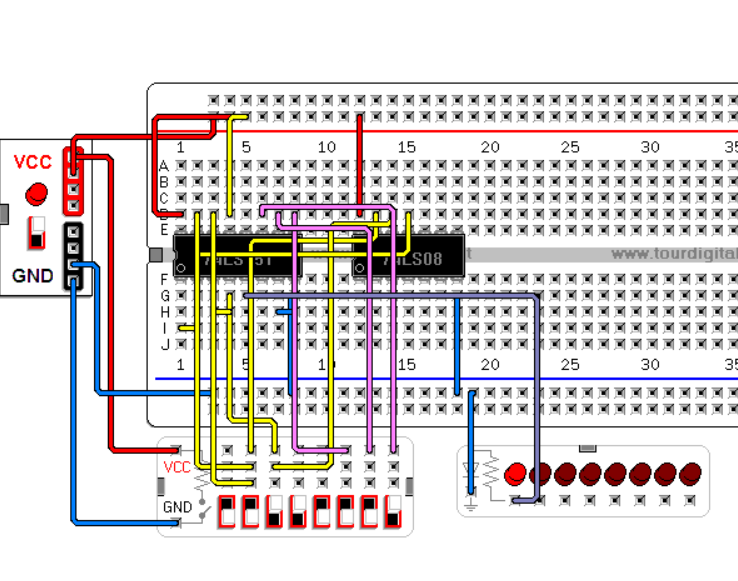
For LED2:



I1 I2 I3 I4 D1 D2 D3

Figure 35: Breadboard design for LED2

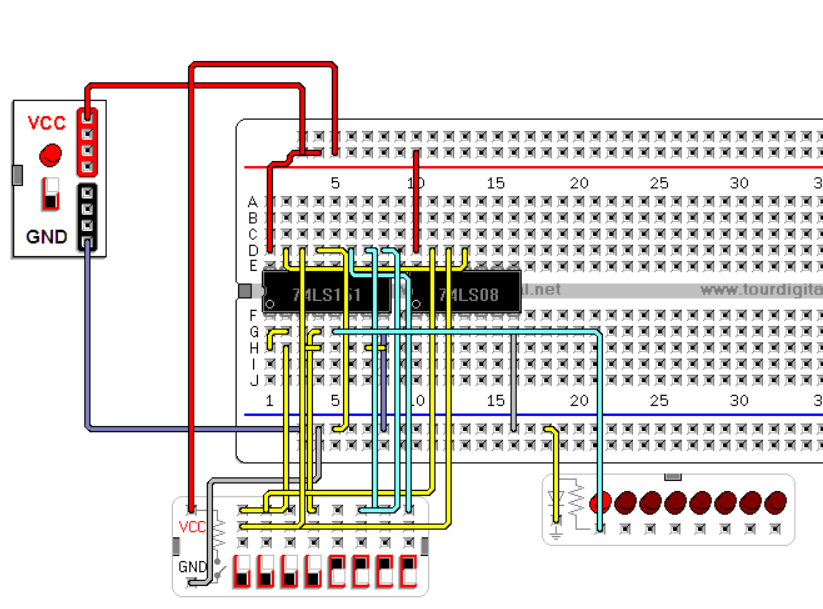
For LED3:



I1 I2 I3 I4 D1 D2 D3

Figure 36: Breadboard design for LED3

For LED4:



I1 I2 I3 I4 D1 D2 D3

Figure 37: Breadboard design for LED4

For detecting a change in the switches to reset the timer:



D1 D2 D3 I1 I2 I3 I4 T

Figure 38: Breadboard design for switch detection circuit

The bit change in the switches is detected through the following methodology (Mealy state diagram):

A diagram of a molecule

AI-generated content may be incorrect.

Figure 39: Mealy state diagram for switch change detection circuit

Whenever the switch changes from 0 to 1, it outputs a 1 (current state is 0 and switch input is 1 = ) then the flipflop takes the input of the switch and outputs the 1 to the XOR gate (now the current state becomes 1 and the switch is 1 = ), now whenever we change the switch back to 0, (current state is 1 and switch is 0 = ) then the flipflops takes the input and outputs a 0 to the XOR gate giving an output of 0 () then repeating the cycle.

# Financial Study

For this project, we made sure to minimize the usage of chips by utilizing already existing expressions.

Below is a bill of quantities displaying the cost of each chip and the total cost:

Table 6: Bill of materials showing materials and total costs

|  |  |  |  |
| --- | --- | --- | --- |
| **Chip** | **Cost per chip** | **Quantity** | **Total Cost** |
| 74LS08 | 0.5$ | 3 | 1.5$ |
| 74LS32 | 0.5$ | 4 | 2.0$ |
| 74LS04 | 0.5$ | 3 | 2.0$ |
| 74LS02 | 0.5$ | 1 | 0.5$ |
| 74LS86 | 0.5$ | 1 | 0.5$ |
| 74LS74 | 0.5$ | 4 | 2$ |
| 74LS151 | 0.5$ | 6 | 3$ |
| 555-timer | 0.5$ | 1 | 0.5$ |
| Breadboard | 1$ | 6 | 6$ |
| Batteries | 10$ | 4 | 40$ |
| Jumping wires | 3$ | 9 | 18$ |
| Capacitors | 0.1 | 4 | 0.4$ |
| Resistors | 0.1 | 15 | 1.5$ |
| **Total** | | | 78$ |

Note that we bought extra components for backups in case we have a malfunction and used several components we already owned. As can be seen, the bulk of the project’s budget lies in the batteries.

# Delay Calculation

For the delay, it can be directly seen from the Quartus implementation:

A screenshot of a computer

AI-generated content may be incorrect.

Figure 40: Delay in Quartus

The circuit experiences the largest delay at LED2, when all the blocks are properly connected together to form the final circuit.

The delay can be calculated as:

🡪

# Power Consumption Analysis

In this part, power consumption is estimated by examining the various components in the circuit. The datasheets of each IC were inspected to obtain power consumption, and the table summarizes the total system consumption, taking the highest power consumption of each chip. Another important assumption is that at any time, we consider 2 gates per chip output HIGH and 2 output LOW, so, we take the average of the current at HIGH and LOW as the typical current for each chip.

Table 7: Power consumption summary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component** | **Quantity** | **Typical Current (mA)** | **Total Current (mA)** | **Total Power (mW)** |
| 74LS08 | 3 | 6.6 | 19.8 | 99 |
| 74LS32 | 4 | 7.35 | 29.4 | 147 |
| 74LS151 | 6 | 8 | 48 | 240 |
| 74LS21 | 1 | 17 | 17 | 85 |
| 74LS74 | 3 | 6 | 18 | 90 |
| 555-timer | 2 | 15 | 30 | 150 |
| 74LS04 | 1 | 5.1 | 5.1 | 25.5 |
| 74LS86 | 1 | 12 | 12 | 60 |
| Total Power consumption | | | | 896.5 |

# Problems Faced

There were several issues we faced during the project, mainly related to poor visibility in the breadboard, lack of access to equipment, and electrical issues (delays, overheating). The issues are further broken down:

1. Shortage and unavailability of some components in shops due to closure and high demand for certain parts. An important factor in our project was mobility: most members of the team do not have a driving license, so it was difficult for us to organize a ride to collect the necessary material from shops far away from Byblos.
2. Small number of breadboards.
3. Difficulty in time management due to other projects and final exams clashing.
4. A larger than usual truth table with 8 inputs and 7 outputs.
5. Creating a feasible design that does not use many chips to decrease the cost of the project.
6. Using the as an input in the table made the state table bigger and a lot harder to come up with a better design, which lead to coming up with the switch change detection circuit and using it as a reset for the 555 timer and a trigger for switching states after 4 seconds (555 timer output = 1). We believe we could have done a simpler design with more flip flops and multiplexers had we been able to obtain the equipment.
7. Creating the 555-timer fast clock for the D Flip-Flops proved challenging.
8. Resetting the 4 seconds timer each time a switch changes.
9. Interference of wires.
10. Overheating of some components.

# Strengths of Our Project

The project presents several strengths:

1. We ensured the use of a minimum number of chips.
2. Our design is modular and scalable (can be scaled to more than 7 states, thus adding more tricks).
3. We leveraged multiplexers to simplify the circuit.
4. We benefitted from two 555-timers to successfully transition states and reset the clock when necessary.
5. The design is compact and able to be packaged in a box.
6. The switch change detector circuit that we came up with is innovative ,simple, and uses principles learned from the lab.

# Bonus

-The GitHub repository with a README is available.

-The NE555 timer datasheet in Latex is also available.

# Conclusion

This project scientifically demonstrates the application of sequential logic and finite state machines (FSMs) to create an interactive system with state-dependent lamp activation. Dynamic mapping ensures functional consistency despite physical reconfiguration, while cap removal introduces interrupt-like behavior affecting system state. The project tangibly illustrates abstract digital logic principles and their relevance to complex real-world applications.