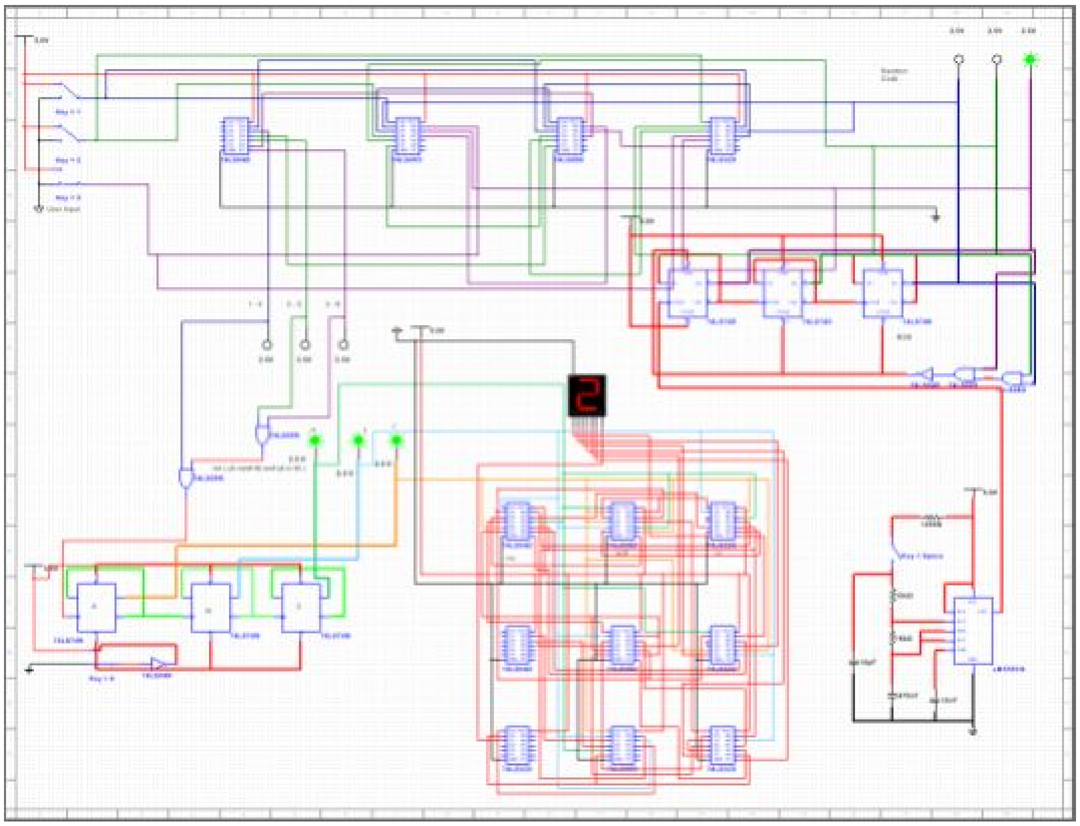
Project 2.1.5 - AOI Circuit Design

Code Breaker



By: Aarush Aitha, Shreejith Ravisankar, Sragvi Vadali, Vikas Ummadisetty
Teacher: Ms. Chou
Honors Digital Electronics Period 7

Table of Contents

Design Brief

Designer	Aarush Aitha, Shreejith Ravisankar, Sragvi Vadali, Vikas Ummadisetty
Client	Consumers who are interested in a game similar to Mastermind but want an electric version that only requires one player.
Problem Statement	People cannot try out board games at stores (health hazard for the pandemic), so they are forced to play online games with screens (which are detrimental to the health of kids)
Design Statement	Create, design and test a working lock-picking key combination puzzle that follows the constraints given.
Constraints	This project must ● have at least six unique inputs for the user and three output ● be easily split for each designer so that each circuit contains at least three inputs and one output ● have a clear purpose and use ● have condition logic ● have sequential logic if chosen
Individual Deliverables	Each designer must have finished and annotated a multisim for their corresponding part of the circuit, build their multisims using a protoboard, and help with the documentation and individual logs
Deliverables	Each team will need to submit documentation of their project on a Google document, and a physical prototype of the design.

Design Specifications

Task: Create a lock-picking game circuit that randomly generates hidden three-bit keys and takes three-bit user guesses that try to match the hidden key. If the user correctly guesses the key, a digital scoreboard will increment by one until reaching the max score of 7, whereby the user can reset the scoreboard to 0.

The design specifications below show how the task was distributed into four subsections: the Key Randomizer, Conditional Logic Circuit, Number Increment Counter, and Digital Displayer. The complete lock-picking game circuit is created by combining these four subsystems in succession, meaning that each subcircuit takes its inputs from the previous subcircuit's outputs. Below are the specifications for the full game's circuit, as well as the specifications for each individual's subcircuit.

<u>Full Lock-Picking Game Circuit (7 User Inputs and 7 Outputs to User)</u>

ON/OFF Switch to Entire Circuit **Input from User**	0 = The circuit will entirely turn off
	1 = The circuit will entirely turn on
Randomizer Toggler **Input from User**	0 = The random number generator will continue to randomly shuffle through three-bit keys
	1 = The random number generator will start to show its shuffling until settling on a random number
ON/OFF Switch To sub-circuit **Input from User**	0 = The circuit will entirely turn off
	1 = The circuit will entirely turn on
User Guess for First Bit of Secret Key **Input from User**	0 = The user guess for the first bit of the secret key is OFF
	1 = The user guess for the first bit of the secret key is ON
User Guess for Second Bit of Secret Key **Input from User**	0 = The user guess for the second bit of the secret key is OFF
	1 = The user guess for the second bit of the secret key is ON

User Guess for Third Bit of Secret Key **Input from User**	The user guess for the third bit of the secret key is OFF
	The user guess for the third bit of the secret key is ON
Scoreboard Incrementer **Input from User**	0 = The user does not prompt the circuit to increment the scoreboard. The scoreboard will never increment with this OFF.
	1 = The user prompts the circuit to increment the scoreboard. If the user guess is correct, scoreboard increments.
Digital Display Letter A **Output to User**	0 = The LED correlated to the position A will be OFF
	1 = The LED correlated to the position A will be ON
Digital Display Letter B **Output to User**	0 = The LED correlated to the position B will be OFF
	1 = The LED correlated to the position B will be ON
Digital Display Letter C **Output to User**	0 = The LED correlated to the position C will be OFF
	1 = The LED correlated to the position C will be ON
Digital Display Letter D **Output to User**	0 = The LED correlated to the position D will be OFF
	1 = The LED correlated to the position D will be ON
Digital Display Letter E **Output to User**	0 = The LED correlated to the position E will be OFF
	1 = The LED correlated to the position E will be ON
Digital Display Letter F **Output to User**	0 = The LED correlated to the position B will be OFF
	1 = The LED correlated to the position F will be ON
Digital Display Letter G **Output to User**	0 = The LED correlated to the position G will be OFF
	1 = The LED correlated to the position G will be ON

Key Randomizer Circuit (Shree)

ON/OFF Switch to Circuit **Input from User**	0 = The circuit will entirely turn off
	1 = The circuit will entirely turn on
Randomizer Toggler **Input from User**	0 = The random number generator will continue to randomly shuffle through three-bit keys
	1 = The random number generator will start to show its shuffling until settling on a random number
Bit Counter Clock Input from 555 Timer Oscillator **Input hidden from User**	0 = The 555 Timer oscillation is at the LOW position
	1 = The 555 Timer oscillation is at the HIGH position, and pushes a signal through the flip flop
First Bit of Secret Key **Output hidden from User**	0 = The random number generated key correlated to the first position in the user key combination is OFF
	1 = The random number generated key correlated to the first position in the user key combination is ON
First Bit of Secret Key **Output hidden from User**	0 = The first bit of the secret key is ON
	1 = The first bit of the secret key is OFF
Second Bit of Secret Key **Output hidden from User**	0 = The second bit of the secret key is OFF
	1 = The second bit of the secret key is ON
Third Bit of Secret Key **Output hidden from User**	0 = The third bit of the secret key is OFF
	1 = The third bit of the secret key is ON

Combinational Logic Circuit (Aarush)

ON/OFF Switch To circuit **Input from User**	0 = Power will turn off to this subcircuit
	1 = Power will turn on to this subcircuit
User Guess for First Bit of Secret Key **Input from User**	0 = The user guess for the first bit of the secret key is OFF
	1 = The user guess for the first bit of the secret key is ON
User Guess for Second Bit of Secret Key **Input from User**	0 = The user guess for the second bit of the secret key is OFF
	1 = The user guess for the second bit of the secret key is ON
User Guess for Third Bit of Secret Key **Input from User**	0 = The user guess for the third bit of the secret key is OFF
	1 = The user guess for the third bit of the secret key is ON
Correctness of User Guess for First Bit **Output hidden from User**	0 = The user guess for the first bit is WRONG
	1 = The user guess for the first bit is CORRECT
Correctness of User Guess for Second Bit **Output hidden from User**	0 = The user guess for the second bit is WRONG
	1 = The user guess for the second bit is CORRECT
Correctness of User Guess for Third Bit **Output hidden from User**	0 = The user guess for the third bit is WRONG
	1 = The user guess for the third bit is CORRECT

<u>Number Increment Counter Circuit (Sragvi)</u>

Correctness of User Guess for First Bit **Input hidden from User**	0 = The user guess for the first bit is WRONG
	1 = The user guess for the first bit is CORRECT
Correctness of User Guess for Second Bit **Input hidden from User**	0 = The user guess for the second bit is WRONG
	1 = The user guess for the second bit is CORRECT
Correctness of User Guess for Third Bit **Input hidden from User**	0 = The user guess for the third bit is WRONG
	1 = The user guess for the third bit is CORRECT
Scoreboard Incrementer **Input from User**	0 = The user does not prompt the circuit to increment the scoreboard. The scoreboard will never increment with this OFF.
	1 = The user prompts the circuit to increment the scoreboard. If the user guess is correct, scoreboard increments.
Scoreboard Clearer **Input from User**	0 = The scoreboard will function normally
	1 = The scoreboard will reset to 0.
Second Bit of the User Score **Output hidden from User**	0 = The second bit of the user score is OFF
	1 = The second bit of the user score is ON
Third Bit of the User Score **Output hidden from User**	0 = The least significant bit of the user score is OFF
	1 = The least significant bit of the user score is ON

Digital Displayer (Vikas)

First Bit of the User Score **Input hidden from User**	0 = The most significant bit of the user score is OFF
	1 = The most significant bit of the user score is ON
Second Bit of the User Score **Input hidden from User**	0 = The second bit of the user score is OFF
	1 = The second bit of the user score is ON
Third Bit of the User Score **Input hidden from User**	0 = The least significant bit of the user score is OFF
	1 = The least significant bit of the user score is ON
Digital Display Letter A **Output to User**	0 = The LED correlated to the position A will be OFF
	1 = The LED correlated to the position A will be ON
Digital Display Letter B **Output to User**	0 = The LED correlated to the position B will be OFF
	1 = The LED correlated to the position B will be ON
Digital Display Letter C **Output to User**	0 = The LED correlated to the position C will be OFF
	1 = The LED correlated to the position C will be ON
Digital Display Letter D **Output to User**	0 = The LED correlated to the position D will be OFF
	1 = The LED correlated to the position D will be ON
Digital Display Letter E **Output to User**	0 = The LED correlated to the position E will be OFF
	1 = The LED correlated to the position E will be ON
Digital Display Letter F **Output to User**	0 = The LED correlated to the position B will be OFF
	1 = The LED correlated to the position F will be ON
Digital Display Letter G **Output to User**	0 = The LED correlated to the position G will be OFF
	1 = The LED correlated to the position G will be ON

Truth Tables & AOI Logic Expressions

Combinational Logic:

This part of the project is checking whether the user inputted guess is equal to the randomly generated code (output of Shree's part). In order to do this, we will need to check whether each position of the user's guess is equal to each position of the randomly generated combination. For example (if the random code is 101):

	Position 1	Position 2	Position 3
User Input	1	0	0
Random Code (from Shree's part)	1	0	1

This would result in an output of "110" from my part of the circuit, since the first two positions are correct but the last one isn't. However, if the user guesses the code correctly:

	Position 1	Position 2	Position 3
User Input	1	0	1
Random Code (from Shree's part)	1	0	1

This would result in an output of "111" since every single position is correct.

In order to check whether each position is equal to each other (for example, checking whether the two yellow boxes in the table above are equal to each other), we cannot use a simple gate such as AND or OR. This is because AND will only result in a value of 1 if both inputs are 1. However, we want the logic to output a 1 if both inputs are equal to each other (1 and 1 OR 0 and 0). Similar issues arise when considering the OR gate. Therefore, we need a way to check whether they are equal or not. Considering the XOR gate, which results in a value of "1" if the inputs are different from each other, we can check whether the two inputs are the same as each other simply by inverting the XOR (making XNOR).

Unfortunately, we were not provided with XOR or XNOR gates in the spare parts box (or maybe one of our group members did not check properly), so we had to generate our own logic. Using inputs A and B, the logic for an XOR gate using solely AND, OR, and NOT IC's is as follows:

(A NAND B) AND (A OR B)

To get XNOR, we simply have to NOT the entire formula:

NOT((A NAND B) AND (A OR B))

Now, let's set up some denotations for the boolean values that we'll be using in the truth tables:

- U1 means the 1st position of the user input.
- U2 means the 2nd position of the user input.
- U3 means the 3rd position of the user input.
- R1 means the 1st position of the random code.
- R2 means the 2nd position of the random code.
- R3 means the 3rd position of the random code.
- O1 is the result of applying XNOR to U1 and R1 (returns 1 if U1 = R1)
- O2 is the result of applying XNOR to U2 and R2 (returns 1 if U2 = R2)
- O3 is the result of applying XNOR to U3 and R3 (returns 1 if U3 = R3)

Truth Tables:

U1 XNOR R1		
U1	R1	O1
0	0	1
0	1	0
1	0	0
1	1	1

U2 XNOR R2		
U2	R2	O2
0	0	1
0	1	0
1	0	0
1	1	1

U3 XNOR R3		
U3	R3	O3
0	0	1
0	1	0
1	0	0
1	1	1

O1, O2, and O3 will now be used as inputs for the next section.

Apart from the inputs denoted in the truth tables above, we also have an additional switch (S3 in the breadboard as shown in the final design, page 17) to turn on or turn off the combinational logic entirely. This is because if we leave the combinational logic on at all times, then the digital outputs will start incrementing because there's a random chance that the user input and the random code will be equal to each other as the random generator cycles through all possible values.

Number Incrementation:

This part of the circuit is used for checking if the outputs from the conditional logic are all high, and creating a counter for the number of times the user gets the combination correct. This part of the circuit also allows the user to clear the number of times they picked the circuit. The circuit uses the variable:

- O1 to represent the output of the first position from the conditional logic.
- O2 to represent the output of the second position from the conditional logic.
- O3 to represent the output of the third position from the conditional logic.
- P to represent if the user has picked the combination
- H to represent the MSB of the 3 bit counter
- I to represent the middle bit of the 3 bit counter
- J to represent the LSB of the 3 bit counter
- Cl to represent if the user toggles the clear switch
- CLR to represent the CLR pin of the D flip-flops

This circuit uses O1, O2, and O3 as inputs and P is not visibly shown in the circuit. The outputted values of H, I, J are used as inputs for the next section of the breadboard

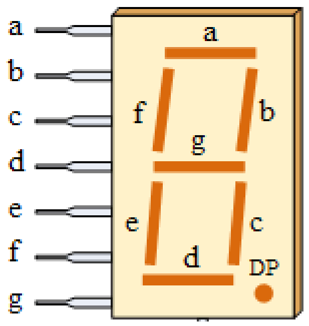
$P = O_1O_2O_3$				
O1	O2	O3		P
0	0	0		0
0	0	1		0
0	1	0		0
0	1	1		0
1	0	0		0
1	0	1		0
1	1	0		0
1	1	1		1

P		H	I	J
0		0	0	0
1		0	0	1
0		0	0	1
1		0	1	0
0		0	1	0
1		0	1	1
0		0	1	1
1		1	0	0
0		1	0	0
1		1	0	1
0		1	0	1
1		1	1	0
0		1	1	0
1		1	1	1
0		1	1	1
1		0	0	0

Cl’ = CLR		
Cl		CLR
0		1
1		0

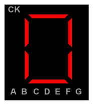
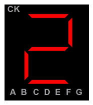
Digital Display Truth Table

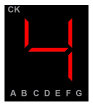
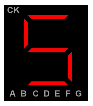
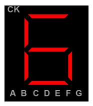
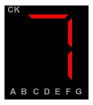
After the three-bit counts are created expressing the number of times the user has picked the lock, we use a digital display to show the value in decimal. The digital display takes seven inputs corresponding to the state of each of its seven dashes. An example of a digital display is shown below:



Source: https://www.electronics-tutorials.ws/blog/7-segment-display-tutorial.html

We utilize combinational logic to convert the three-bit counts into a decimal-showing digital display, mapping the three bit inputs (H, I, J) to seven digital display pin outputs (A, B, C, D, E, F, G). The truth table exhibits all combinations of this system. NUM represents the resulting image of the digital display.

Inputs				Outputs							
H	I	J		A	B	C	D	E	F	G	NUM
0	0	0		1	1	1	1	1	1	0	
0	0	1		0	1	1	0	0	0	0	
0	1	0		1	1	0	1	1	0	1	
0	1	1		1	1	1	1	0	0	1	

1	0	0		0	1	1	0	0	1	1	
1	0	1		1	0	1	1	0	1	1	
1	1	0		1	0	1	1	1	1	1	
1	1	1		1	1	1	0	0	0	0	

With these 8 combinations for displaying decimal values from 0 (000_2) to 7 (111_2), we draw relationships between the three inputs and seven outputs with boolean logic. We simplify seven boolean expressions for each of the display's inputs (A-G) below. Due to the large size of this circuit, simplification is of paramount importance to reduce the number of wires and integrated circuits needed, due to limited electronic components.

Output	Simplification Steps	Final Simplification
A	$A = (H + I + \bar{J})(\bar{H} + I + J)$ $A = H\bar{H} + HI + HJ + I\bar{H} + II + IJ + \bar{J}\bar{H} + \bar{J}I + \bar{J}J$ $A = HI + HJ + I\bar{H} + I + IJ + \bar{J}\bar{H} + \bar{J}I$ $A = HJ + I + \bar{J}\,\bar{H}$ $A = HJ + I + \overline{J + H}$	$A = HJ + I + \overline{J + H}$
B	$B = (\bar{H} + I + \bar{J})(\bar{H} + \bar{I} + J)$ $B = \overline{H\,H} + \bar{H}\bar{I} + \bar{H}J + I\bar{H} + I\bar{I} + IJ + \bar{J}\bar{H} + \bar{J}\bar{I} + \bar{J}J$ $B = \bar{H} + \bar{H}\bar{I} + \bar{H}J + I\bar{H} + IJ + \bar{J}\bar{H} + \bar{J}\bar{I}$ $B = \bar{H} + IJ + \bar{J}\bar{I}$ $B = \bar{H} + IJ + \overline{J + I}$	$B = \bar{H} + IJ + \overline{J + I}$
C	$C = H + \bar{I} + J$	$C = H + \bar{I} + J$

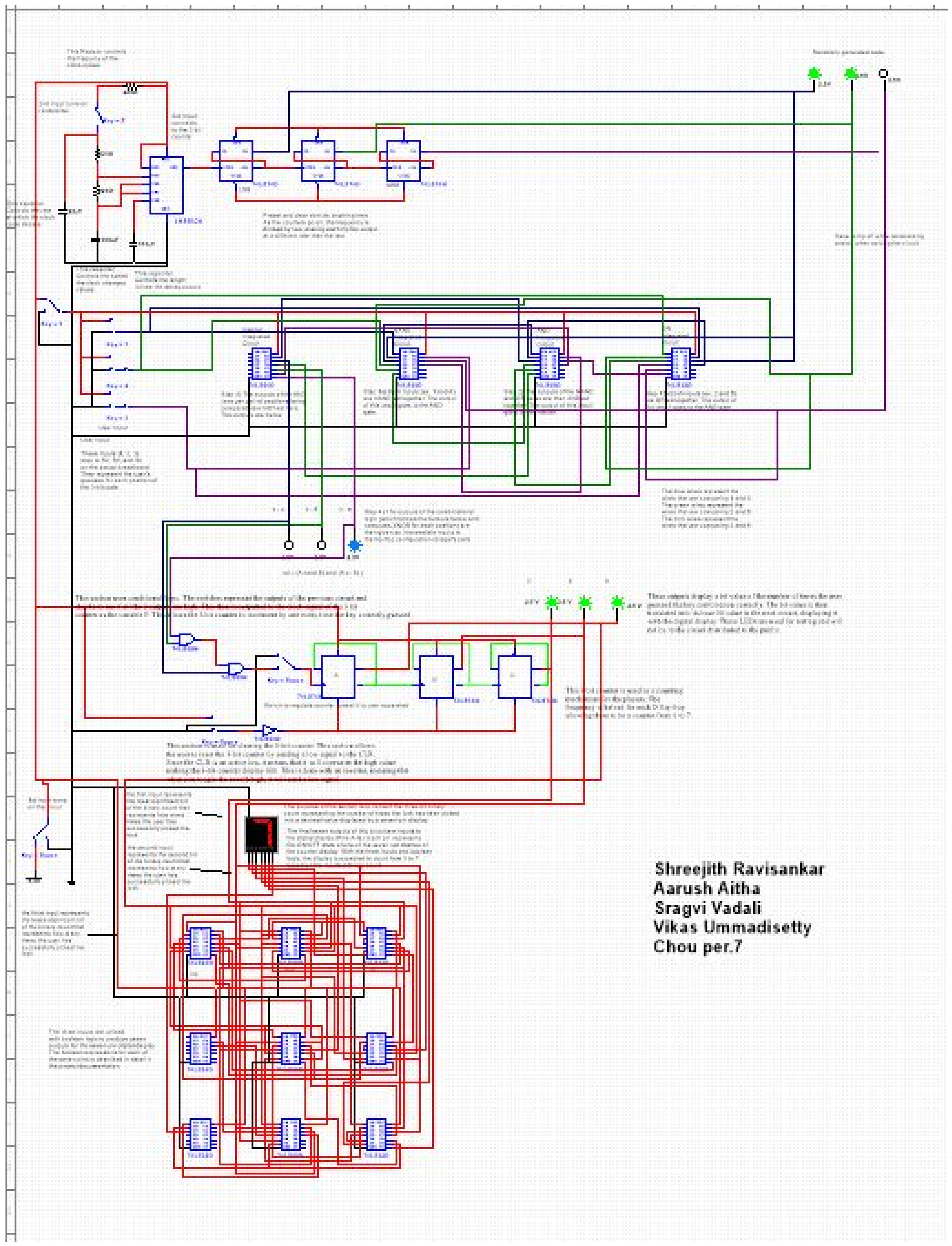
D	$D=(H+I+\bar{J})(\bar{H}+I+J)(\bar{H}+\bar{I}+\bar{J})$ $D=(H\bar{H}+HI+HJ+I\bar{H}+II+IJ+\bar{J}\bar{H}+\bar{J}I+\bar{J}J)(\bar{H}+\bar{I}+\bar{J})$ $D=(HI+HJ+I\bar{H}+I+IJ+\bar{J}\bar{H}+\bar{J}I)(\bar{H}+\bar{I}+\bar{J})$ $D=(HJ+I+\bar{J}\bar{H})(\bar{H}+\bar{I}+\bar{J})$ $D=HJ\bar{H}+HJ\bar{I}+HJ\bar{J}+I\bar{H}+I\bar{I}+I\bar{J}+\bar{J}\bar{H}\bar{H}+\bar{J}\bar{H}\bar{I}+\bar{J}\bar{H}\bar{J}$ $D=HJ\bar{I}+I\bar{H}+I\bar{J}+\bar{J}\,\bar{H}+\bar{J}\bar{H}\bar{I}+\bar{H}\bar{J}$ $D=HJ\bar{I}+I(\bar{H}+\bar{J})+\bar{J}\bar{H}$ $D=HJ\bar{I}+I\overline{HJ}+\bar{J}\bar{H}$ XOR solution: $D=I\oplus(HJ)+\bar{J}\bar{H}$	$D=HJ\bar{I}+I\overline{HJ}+\bar{J}\bar{H}$
E	$E=\bar{H}\bar{I}\bar{J}+\bar{H}I\bar{J}+HI\bar{J}$ $E=\bar{H}\bar{J}(I+\bar{I})+HI\bar{J}$ $E=\bar{H}\bar{J}+HI\bar{J}$ $E=\bar{J}(\bar{H}+HI)$ $E=\bar{J}(\bar{H}+I)$	$E=\bar{J}(\bar{H}+I)$
F	$F=\bar{H}\bar{I}\bar{J}+H\bar{I}\bar{J}+H\bar{I}J+HI\bar{J}$ $F=\bar{I}\bar{J}(\bar{H}+H)+H\bar{I}J+HI\bar{J}$ $F=\bar{I}\bar{J}+H\bar{I}J+HI\bar{J}$ $F=\bar{I}(\bar{J}+HJ)+HI\bar{J}$ $F=\bar{I}(\bar{J}+H)+HI\bar{J}$ $F=\bar{I}\bar{J}+\bar{I}H+HI\bar{J}$ $F=\bar{I}\bar{J}+H(\bar{I}+I\bar{J})$ $F=\bar{I}\bar{J}+H(\bar{I}+\bar{J})$ $F=\bar{I}\bar{J}+H\overline{IJ}$	$F=\bar{I}\bar{J}+H\overline{IJ}$
G	$G=(H+I+J)(H+I+\bar{J})(\bar{H}+\bar{I}+\bar{J})$ $G=(HH+HI+H\bar{J}+IH+II+I\bar{J}+JH+JI+J\bar{J})(\bar{H}+\bar{I}+\bar{J})$ $G=(H+I)(\bar{H}+\bar{I}+\bar{J})$ $G=H\bar{H}+H\bar{I}+H\bar{J}+I\bar{H}+I\bar{I}+I\bar{J}$	$G=H\overline{IJ}+\bar{H}I$

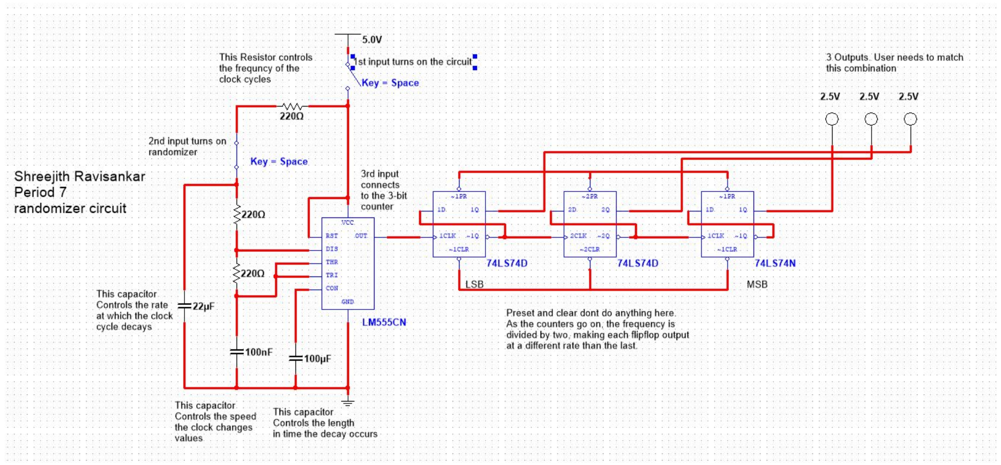
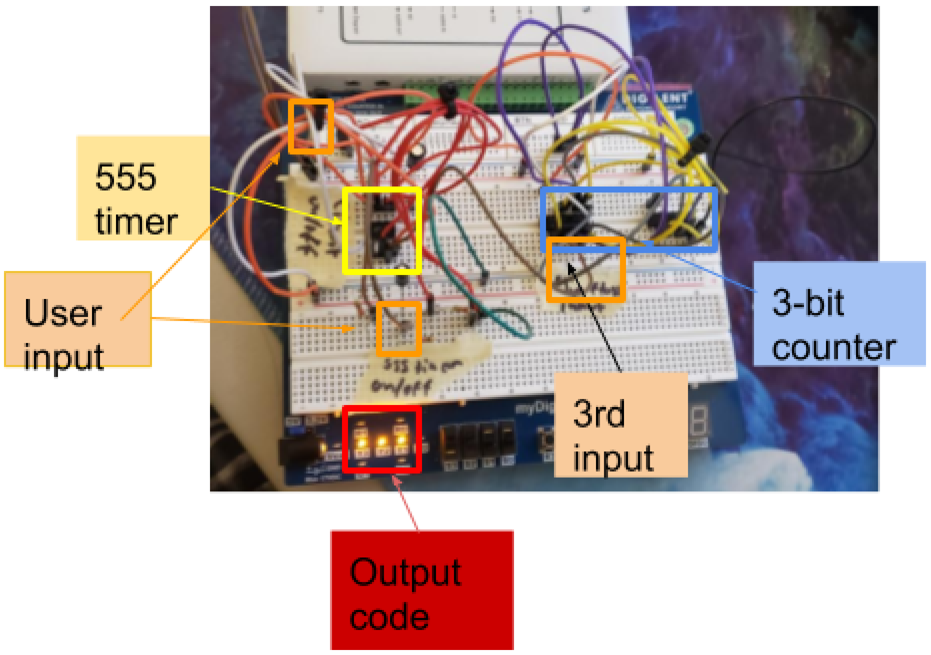
	$G = H\overline{I} + H\overline{J} + I\overline{H} + I\overline{J}$ $G = H\overline{I} + H\overline{J} + \overline{H}I + (H + \overline{H})I\overline{J}$ $G = H\overline{I} + H\overline{J} + \overline{H}I + HI\overline{J} + \overline{H}I\overline{J}$ $G = H\overline{I} + H\overline{J} + \overline{H}I + HI\overline{J} + \overline{H}I\overline{J}$ $G = H\overline{I} + H\overline{J}(I + 1) + \overline{H}I(1 + \overline{J})$ $G = H\overline{I} + H\overline{J} + \overline{H}I$ or $G = H\oplus I + H\overline{J}$ $G = H(\overline{I} + \overline{J}) + \overline{H}I$ $G = H\overline{IJ} + \overline{H}I$ XOR solution: $G = H\oplus I + H\overline{J}$	

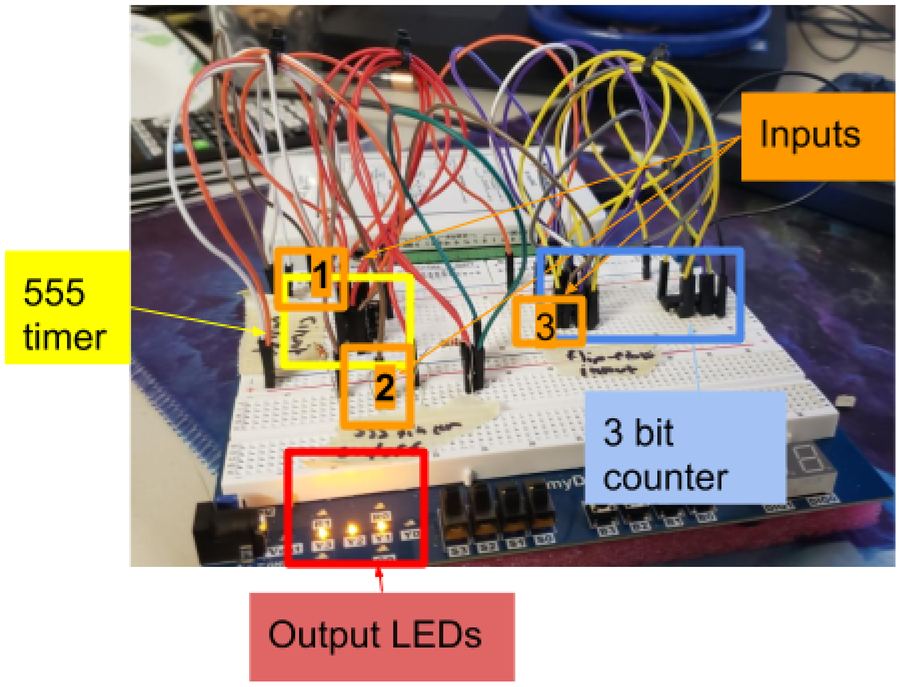
These seven combinational logic expressions are used to build the final circuit. To further optimize the build of the circuit, overlapping boolean terms are reused to avoid constructing redundant repeats.

Final Design

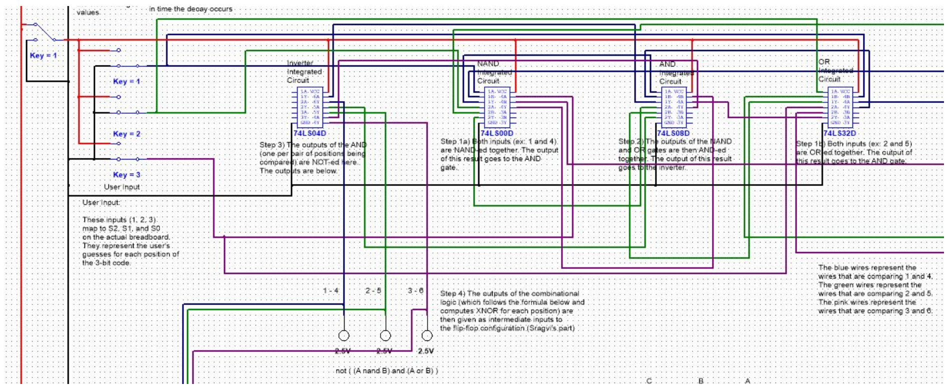
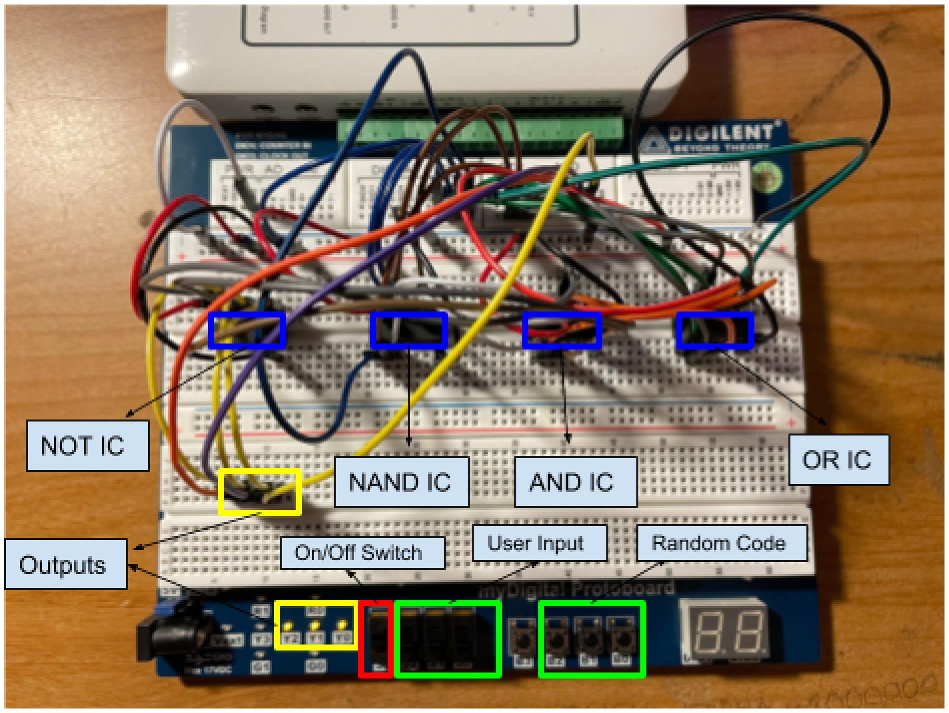
Final Design Multisim:

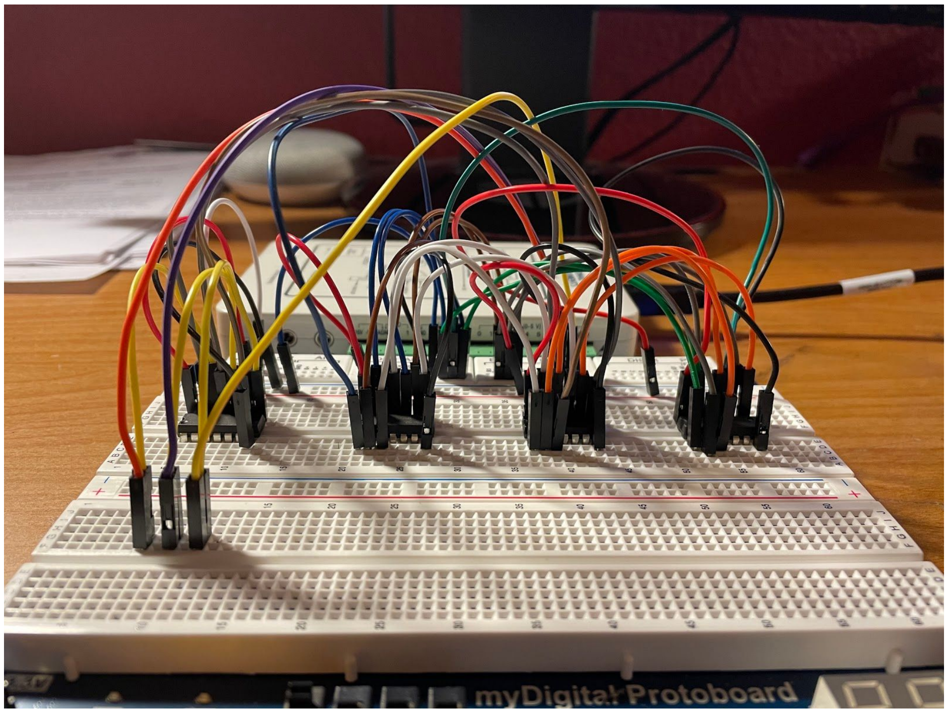


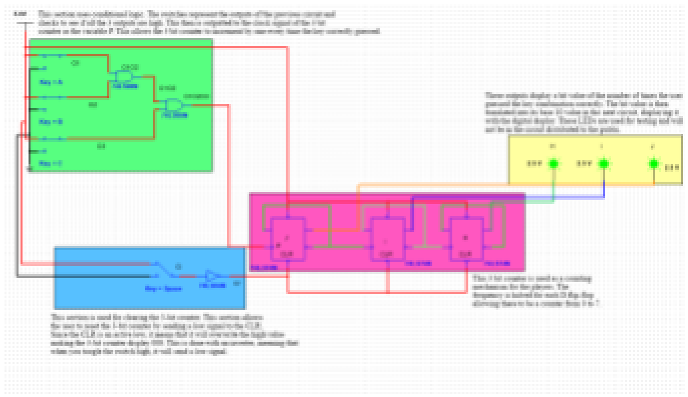
Randomizer circuit	
Multisim	 5.0V 1st input turns on the circuit Key = Space This Resistor controls the frequncy of the clock cycles 220Ω 2nd input turns on randomizer Key = Space Shreejith Ravisankar Period 7 randomizer circuit 220Ω 220Ω 3rd input connects to the 3-bit counter VCC RST OUT DIS THR TRI CON GND LM555CN This capacitor Controls the rate at which the clock cycle decays 22µF 100nF 100µF This capacitor Controls the speed the clock changes values This capacitor Controls the length in time the decay occurs ~1PR 1D 1Q 1CLK ~1Q ~1CLR 74LS74D ~2PR 2D 2Q 2CLK ~2Q ~2CLR 74LS74D 74LS74N LSB MSB Preset and clear dont do anything here. As the counters go on, the frequency is divided by two, making each flipflop output at a different rate than the last. 3 Outputs. User needs to match this combination 2.5V 2.5V 2.5V
Top View	 555 timer User input 3-bit counter 3rd input Output code

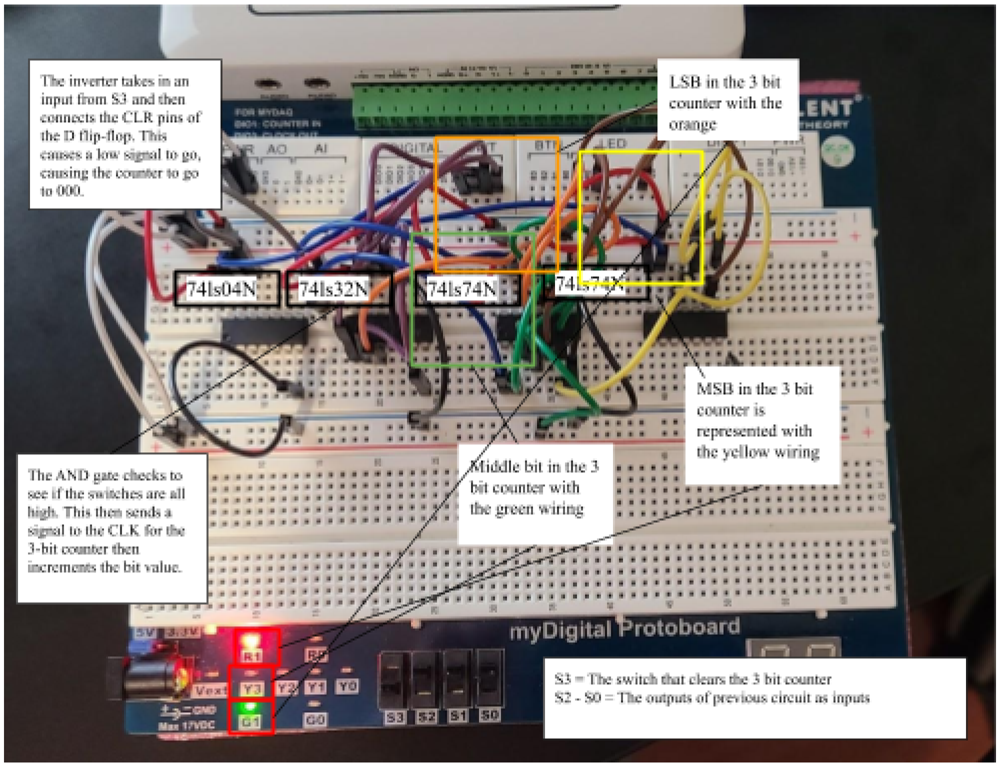
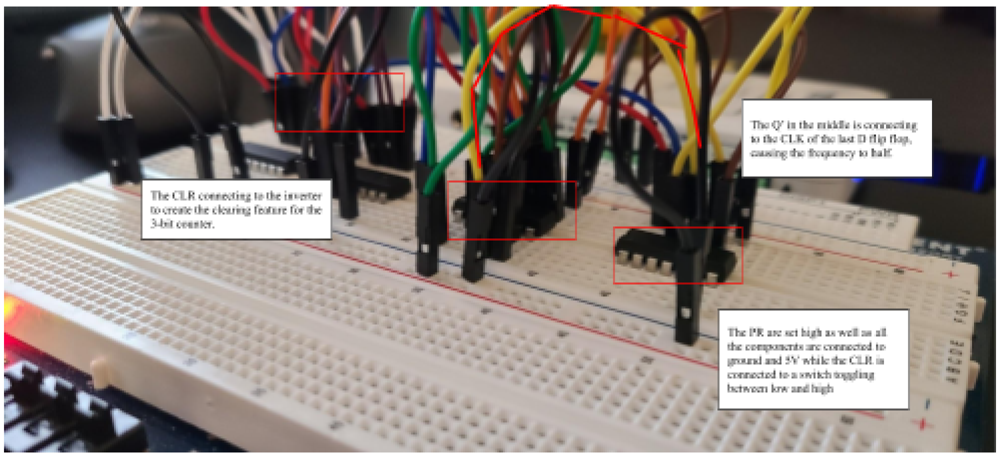
Side View	 Inputs 555 timer 1 2 3 3 bit counter Output LEDs The user inputs in this circuit are the on/off switch for the entire circuit connected directly to power, and the randomizer switch for the 555 timer. The 3rd input is the input from the 555 timer to the flip flop and it is not user controlled. The 2 user inputs are wires that you plug and unplug. All the inputs are also labled and used with brown wires for easy identification. To use the circuit, the user has to first plug in the power (the first input). After power is turned on, the randomizer input has to be plugged in and unplugged after a couple seconds (the second input). Afterwards the current is sent to the 3 bit counter and after a couple more seconds the leds at the bottom will stop switching and settle on a combination. For the 555 timer, I made sure to use the lowest resistors I had, and I connected the smallest capacitor to the trigger and threshold to make the output leds to switch as fast as possible so the randomness is exemplified. The 3 bit counter is only used so that each output LED changes at a different rate from each other.

Conditional Logic Circuit	
Multisim	User Input:

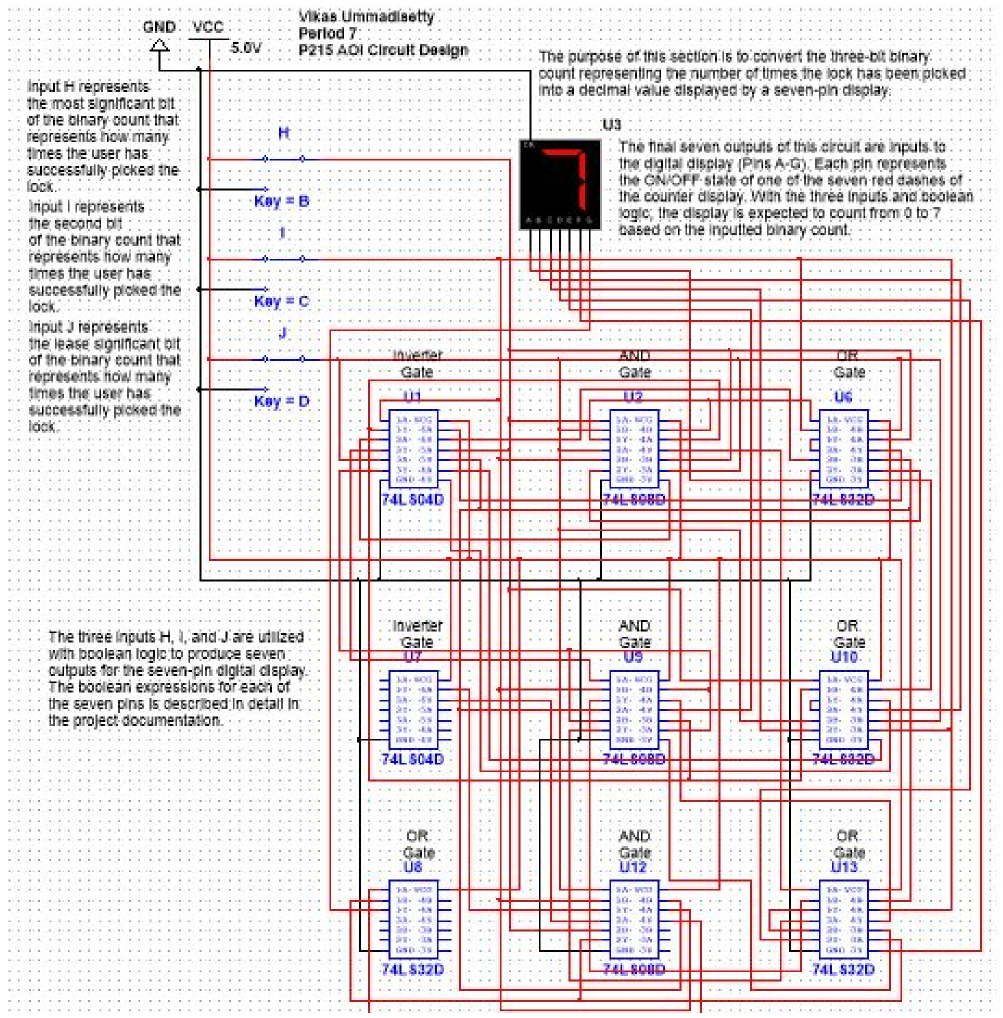
	 values in time the decay occurs Key = 1 Key = 1 Key = 2 Key = 3 User Input Inverter Integrated Circuit 74LS04D NAND Integrated Circuit 74LS00D AND Integrated Circuit 74LS08D OR Integrated Circuit 74LS32D 1 - 4 2 - 5 3 - 6 2.5V 2.5V 2.5V not ((A nand B) and (A or B)) C B A User Input: These inputs (1, 2, 3) map to S2, S1, and S0 on the actual breadboard. They represent the user's guesses for each position of the 3-bit code. Step 3) The outputs of the AND (one per pair of positions being compared) are NOT-ed here. The outputs are below. Step 1a) Both inputs (ex: 1 and 4) are NAND-ed together. The output of this result goes to the AND gate. Step 2) The outputs of the NAND and OR gates are then AND-ed together. The output of this result goes to the inverter. Step 1b) Both inputs (ex: 2 and 5) are OR-ed together. The output of this result goes to the AND gate. Step 4) The outputs of the combinational logic (which follows the formula below and computes XNOR for each position) are then given as intermediate inputs to the flip-flop configuration (Sragvi's part) The blue wires represent the wires that are comparing 1 and 4. The green wires represent the wires that are comparing 2 and 5. The pink wires represent the wires that are comparing 3 and 6.
Top View	 NOT IC NAND IC AND IC OR IC Outputs On/Off Switch User Input Random Code In this picture, we can see how the XNOR logic has been applied to each position (S2 XNOR B2, S1 XNOR B1, S0 XNOR B0). As stated earlier in the documentation, each of these switches and buttons correspond to the positions of the user's guess and the randomly generated code (respectively). First, the user input and the random code are NAND-ed together. In the same step, the user input and the random code are OR-ed together. Then, the outputs from the NAND and OR gates are then AND-ed together in the next step. Finally, the outputs from the AND gate are inverted by the NOT integrated circuit. These outputs lead to LED's Y2, Y1, and Y0 to show whether the user has guessed those positions correctly. The on/off switch S3 controls the entire circuit. If it is turned off, then the combinational logic is also turned off entirely.

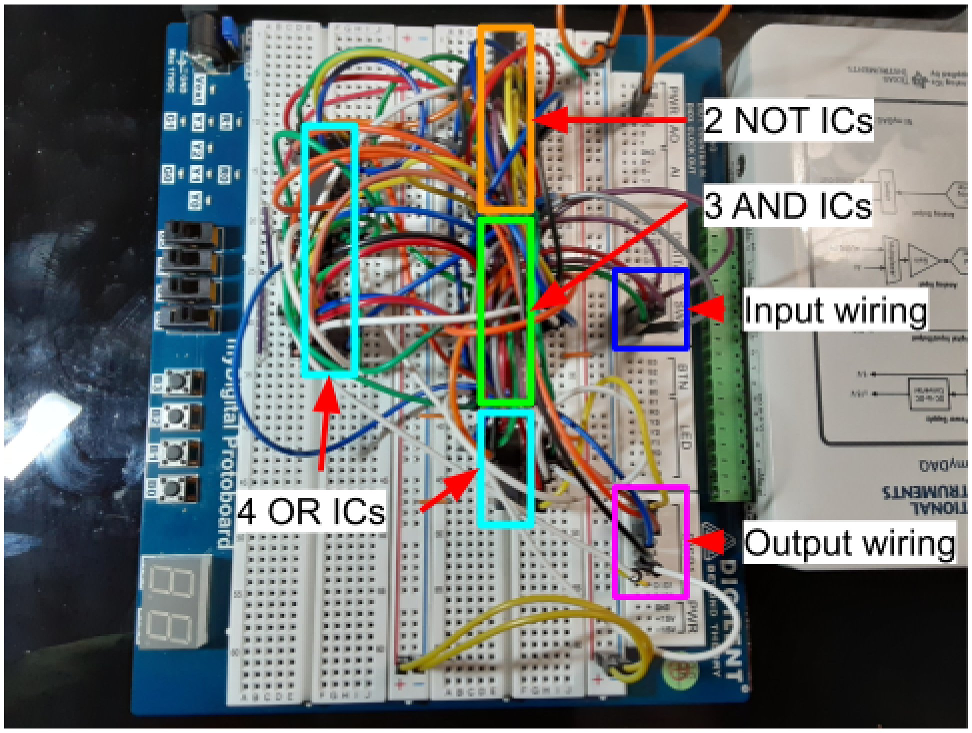
Side View(s)	 (Extra view of the combinational logic breadboard)
Signature: Aarush Aitha	Date: 12/05/2020

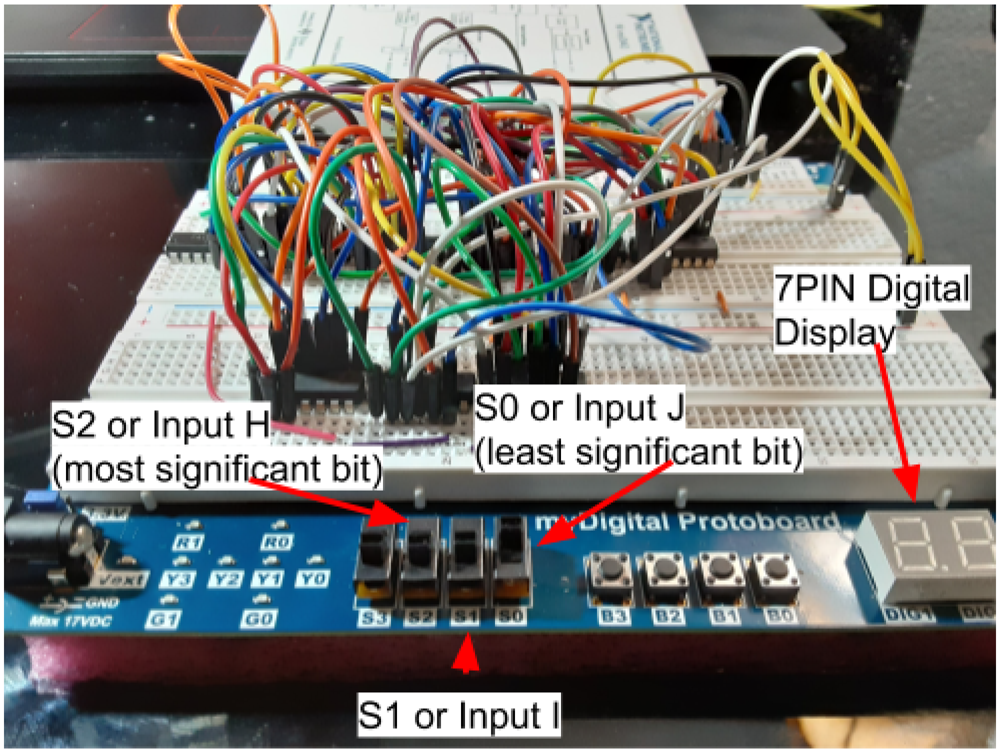
Number Incrementation	
Multisim	

Top View	 The inverter takes in an input from S3 and then connects the CLR pins of the D flip-flop. This causes a low signal to go, causing the counter to go to 000. LSB in the 3 bit counter with the orange MSB in the 3 bit counter is represented with the yellow wiring Middle bit in the 3 bit counter with the green wiring The AND gate checks to see if the switches are all high. This then sends a signal to the CLK for the 3-bit counter then increments the bit value. S3 = The switch that clears the 3 bit counter S2 - S0 = The outputs of previous circuit as inputs 74ls04N 74ls32N 74ls74N 74ls74N myDigital Protoboard
Side View	 The Q' in the middle is connecting to the CLK of the last D flip flop, causing the frequency to half. The CLR connecting to the inverter to create the clearing feature for the 3-bit counter. The PR are set high as well as all the components are connected to ground and 5V while the CLR is connected to a switch toggling between low and high

Signature: [illegible] Date: 12/05/2020

Digital Display Counter	
Multisim	 Vikas Ummadisetty Period 7 P215 AOI Circuit Design The purpose of this section is to convert the three-bit binary count representing the number of times the lock has been picked into a decimal value displayed by a seven-pin display. Input H represents the most significant bit of the binary count that represents how many times the user has successfully picked the lock. Input I represents the second bit of the binary count that represents how many times the user has successfully picked the lock. Input J represents the lease significant bit of the binary count that represents how many times the user has successfully picked the lock. The final seven outputs of this circuit are inputs to the digital display (Pins A-G). Each pin represents the ON/OFF state of one of the seven red dashes of the counter display. With the three inputs and boolean logic, the display is expected to count from 0 to 7 based on the inputted binary count. The three inputs H, I, and J are utilized with boolean logic to produce seven outputs for the seven-pin digital display. The boolean expressions for each of the seven pins is described in detail in the project documentation. GND VCC 5.0V H Key = B I Key = C J Key = D U3 Inverter Gate U1 74LS04D AND Gate U2 74LS08D OR Gate U6 74LS32D Inverter Gate U7 74LS04D AND Gate U9 74LS08D OR Gate U10 74LS32D OR Gate U8 74LS32D AND Gate U12 74LS08D OR Gate U13 74LS32D

Top View	 2 NOT ICs 3 AND ICs Input wiring 4 OR ICs Output wiring The image above shows the top view of the digital display circuit. The system uses 2 Inverter ICs, 3 AND ICs, and 4 OR ICs. When designing the circuit, the IC components were placed as close to one another as possible to minimize the distance travelled by each wire. The image also shows the relative areas where the input and outputs were wired to before and after the combinational logic. To further optimize the circuit, we reused many boolean terms to avoid the need to reconstruct redundant repetitions.
Signature: Vikas Ummadisetty	Date: 12/05/2020

Side View	 7PIN Digital Display S2 or Input H (most significant bit) S0 or Input J (least significant bit) S1 or Input I This side view shows the input and outputs of the digital display circuit. This circuit takes a three-bit count as an input, which represents the number of times the user has successfully picked the lock. S2 represents Input H or the most significant bit, S1 represents Input I, and S0 represents Input J or the least significant bit. The final output translates this three-bit count into a decimal value on the digital display DIG1, which takes seven pin inputs (A-G). Each pin (A-G) represents the ON/OFF state of each of the dashes of the display.

Final Solution Summary Paragraph:

Overall, our circuit created a game in which a user may first generate a random code (using sequential logic). Then, they can try to guess what the code is using switches (these inputs will be compared to the randomly generated code using combinational logic). Finally, if the user guesses the current code correctly, the score will increment itself by 1 (using chained flip-flops and the digital display on the proto-board). The sequential logic part comes first, as the user must first generate a random code in order to start the game. A button (SPST) will be pressed in order to trigger the stoppage of the clock signal coming from the 555 timer. This will make the flip-flop configuration land on a random 3-bit number from 000 to 111. The user will then turn on the combinational logic using a switch (SPDT). The combinational logic will check whether each position of the user's input (3 SPDT's) is equal to its corresponding position in the randomly generated code. The 3 outputs for this circuit (one for each position) will be sent to the number incrementation part of the circuit. The number incrementation (a combination of a flip-flop sequence and a digital display) will increment the user's score by 1 once they have guessed it correctly. This counts as 7 outputs because each digital display has 7 LED "pegs". Some major issues we had were that the random number generator, if continuously on, would cycle through and increment through the number incrementation, causing the digital display to display "7". This means that the user would have "won" the game 7 times when they won the game. We fix this issue by adding a switch to the 3 bit counter so that the random number generator would then generate a number with no power added to the counter. With this, it allowed the 3-bit counter to not take any power for the clock, allowing it to not increment as the random number generator is shuffling through its values and not increment to the number incrementation. Another issue we had was when building the number incrementation. When building the number incrementation on the breadboard, the 3-bit counter would function correctly with the clock voltage, but would not work when using a switch for the clock. The switches in real life, there are many variables involving duration of the switch, as some instantaneously switch while the others do not. With the switches, human error on the speeds of toggling the switch caused a grey area, making the switch not follow the standard pattern of the 3 bit counter. In the multisim, we realized that the switches have an instantaneous power to the clock voltage, allowing that grey area to be too small, allowing the circuit to work with high accuracy. If we were given more time, we would have added more inputs and made the 3-bit counter for number incrementation into a 4 bit counter. With the increase in the number of bits in the number incrementation, we would have needed to add another digital display, causing there to be more gates. This would have made our design more expensive. Given more time, we would have also tried to find a different solution for the 3 bit counter incrementation as our current design only theoretically worked.

Real-Life Cost Analysis:

Below is the cost analysis of the final lock-picking circuit, broken down by each subsystem. We take into AND ICs, OR ICs, NOT ICs, NAND ICs, D Flip Flops, 555 Timers, SPDTs, SPST, DIG Displays, and LEDs. We do not take into account intermediate components (extra probes and switches) in order to perform a cost analysis of the real-life lock-picking circuit.

Estimated Cost of Circuit										
	AND ICs (74LS08)	OR ICs (74LS32)	NOT ICs (74LS04)	NAND ICs (74LS00D)	D Flip Flop (74LS74)	555 Timer (LM555CN)	SPDT	SPST	DIG Display	LED
Randomizer Circuit	0	0	0	0	3	1	1	1	0	0
Combinational Logic Circuit	1	1	1	1	0	0	4	0	0	3
Number Incrementation Circuit	2	0	1	0	3	0	2	0	0	0
Digital Displayer Circuit	3	4	2	0	0	0	0	0	1	0
Total Gates	6	5	4	1	6	1	7	1	1	3
Cost per Part	$0.69	$0.59	$0.69	$1.09	$0.59	$0.85	$0.65	$0.28	$1.02	$0.07
Subtotal Cost	$4.14	$2.95	$2.76	$1.09	$3.54	$0.85	$4.55	$0.28	$1.02	$0.21

Total Cost	$21.39

Real-Life Implementation:
In real life, this circuit can be implemented as an electronic game, in which users can enter guesses to try to “pick” the lock. This electronic game can be built as an electronic board game, an arcade game machine, or a carnival game. For instance, in a carnival game, the carnival game operator can control the scoreboard resetter, the scoreboard incrementer, the circuit ON/OFF switch, and the randomizer, while the user can control their three-bit guesses. The user guess inputs can also be modified to mimic lock picking by integrating larger switches (six pole single throw) or even external hardware (scroll wheel number pins).

The key real life learned through this electronic game could be for actual lockpicking where you need to guess the position of each pin in the lock in order to open it. Most commonly, locksmiths use a pick to set the position of the pins in a lock to the position they think is right, but many times when they are right, the lock will make a clicking sound just like the LEDs in this circuit and how they tell you whether you are correct or not. And since locks are generally made differently depending on brand and model, there is also a certain level of randomness involved meaning similar to how the circuit creates a random combination every single time. Because of this Locksmiths need to be able to be flexible and adapt to new locks and pick them so they won't have to outright break a lock to help their customers get what they want.

This is the link to the video:
https://drive.google.com/file/d/1MsqzAPOFxcx4fV_H5PtHPa4OArSXpz0R/view?usp=sharing