Pa**wo*d is a weak Password

Micro Project

Parth Maradia

INDEX

PROJECT BACKGROUND

Privacy and information security is one of the main aims of Computing technology. Several fields like information theory, cryptography etc. were used to store and process information. In today's world, one of the main ways to conceal or limit access to information is the concept of a password.

The concept of a password is not a new one. Several ancient societies used 'spoken' passwords to know the accountability of a person.

However, if we look at the modern scenario, we find that a password is a string of alpha-numerals often involving special characters like '~, @, ^' etc. In fact, the 'strength' of a password is said to be directly proportional to the number of special characters it contains!

However, this brings us to our aforementioned mysterious concept of a 'strength' of a password. With the proliferation of computing technology, people with less honourable motives developed a new discipline - " Cracking ". However, one of the main ways to break into information guarded by a password is by 'guessing' . So, it is vital for anyone who wishes to safeguard his information to choose a strong password. This means, the password must contain special characters and several other characters which are generally not used in common text.

And thus, we reach the goal of the project. Our aim is to design a circuit which recognises a strong password from a weak one.

OBJECTIVE

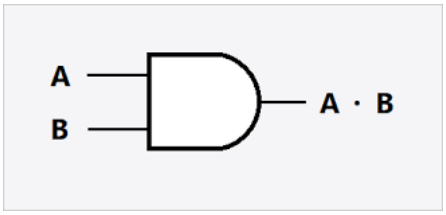
To design a valid logic circuit that accepts a password of 7 characters as input and gives output as 1 if the given password satisfies certain conditions, otherwise it gives 0. These conditions are:

- At Least one capital letter (A-Z)
- At Least one numeral (0-9)
- At Least one special character from the set (& , * , $, - , #)
- No other letters or characters

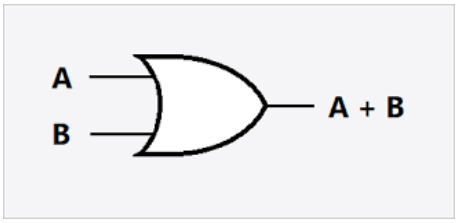
INTRODUCTION

Logic gates are the basic building blocks of any digital system. It is an electronic circuit having one or more than one input and only one output. The relationship between the input and the output is based on a certain logic. Based on this, the most commonly used logic gates are **AND** gate, **OR** gate, **NOT** gate etc.

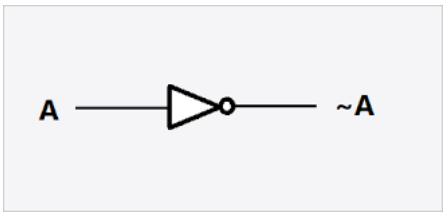
AND Gate:



OR Gate:



NOT Gate:



DESIGN METHODOLOGY

We started our project by doing a detailed analysis of the project specifications and the objectives given to us. We then went on to thinking and familiarizing ourselves with the powerful concept of **Karnaugh Maps.** This helped us create a rough design and then we went about optimising and refining our design.

We had two options to design our circuit- a modular design or a simple circuit. Even though for small circuits it is advised to focus on silicon real estate, we chose the modular design. This was simply because **Modular Designs** are more organized and easier to repair in case of damages.

To make a **Modular Design** we decided to create three separate **Blocks** to validate each specification given to us. Since we were given 7 character passwords in the specifications, we realized that we needed 21 blocks in this circuit (3 per character).

While designing the **Blocks** we observed that the first bit for every character specified to us was 0 so we decided not to use it during the verification process in order to reduce the number of inputs and make the block more efficient.

BLOCK - 1 (Special Characters)

We decided to start designing the easiest block first in order to familiarize ourselves with this concept , so we started with the block that validates a particular set of special characters only (**$, # , - , & , ***) . We analysed the **Binary Values** of these characters first and found that the second , third and the fourth bits of their binary representation were common (0 1 0) so we designed a logic gate circuit to verify them specifically. Then we used the Karnaugh Map to derive a simple **Boolean Expression** which we further simplified using logic.For every combination of bits in the Kmap that gave output as 1, we included them in our original boolean expression for the logic circuit.

We denoted the last four bits as A , B , C , D and constructed the KMAP as follows.

TRUTH TABLE - 1:-

Bit-1	Bit-2	Bit-3	Bit-4	Special Characters	Truth Value
0	0	0	1	!	0
0	0	1	0	"	0
0	0	1	1	#	1
0	1	0	0	$	1
0	1	0	1	%	0
0	1	1	0	&	1
1	0	1	1	;	0
1	0	0	0	(	0
1	0	0	1	)	0
1	0	1	0	*	1
1	0	1	1	+	0
1	1	0	0	,	0
1	1	0	1	-	1
1	1	1	0	.	0
1	1	1	1	/	0

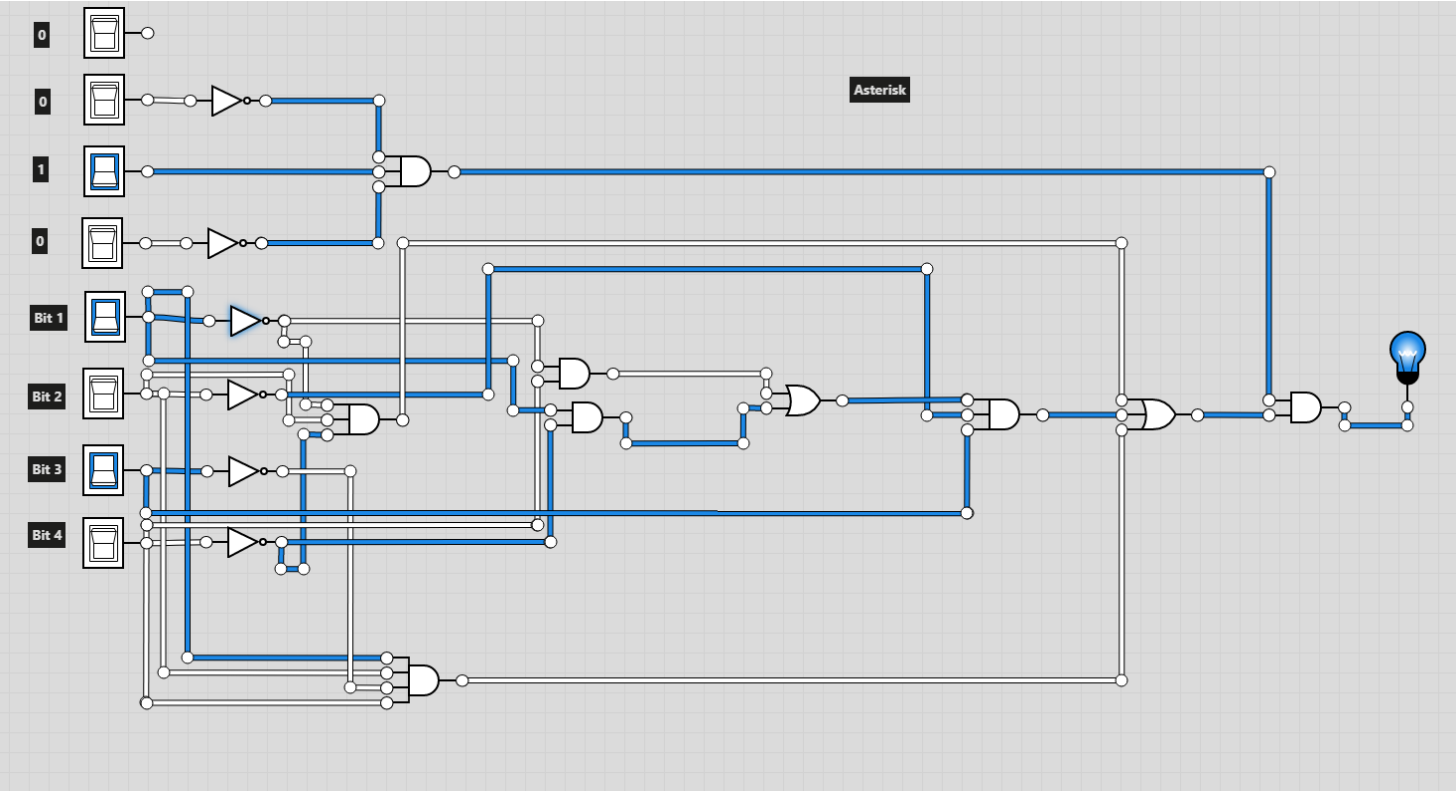
KMAP - 1:-

	CD	→			
AB		**00**	**01**	**11**	**10**
↓	**00**	0	0	1	0
	01	1	0	0	1
	11	0	1	0	0
	10	0	0	0	1

Original Equation derived from the **KMAP** : A’BC’D’ + ABC’D + A’B’CD + AB’CD’ + A’BCD’

Final simplified **Boolean Expression :** A’BD’ + B’C(A’D + AD’) + ABC’D

In Block-1 we used six NOT gates , seven AND gates and two OR gates and we assembled them in the following way.



After deriving the final boolean expression we tested it on an online simulator to verify it.

BLOCK - 2 (Numerals)

We again analyzed the Binary Values of all the numerals and we observed a pattern that the second , third and the fourth bits were common in all of them (0 1 1). For the remaining four bits we used the Karnaugh Map to derive a simple **Boolean Expression** which we further simplified using logic. First , we paired all the 1s in the first and second row(2^3) and obtained A' . Then we paired the first two 1s in the first and last row and we obtained B'C' .

TRUTH TABLE:-

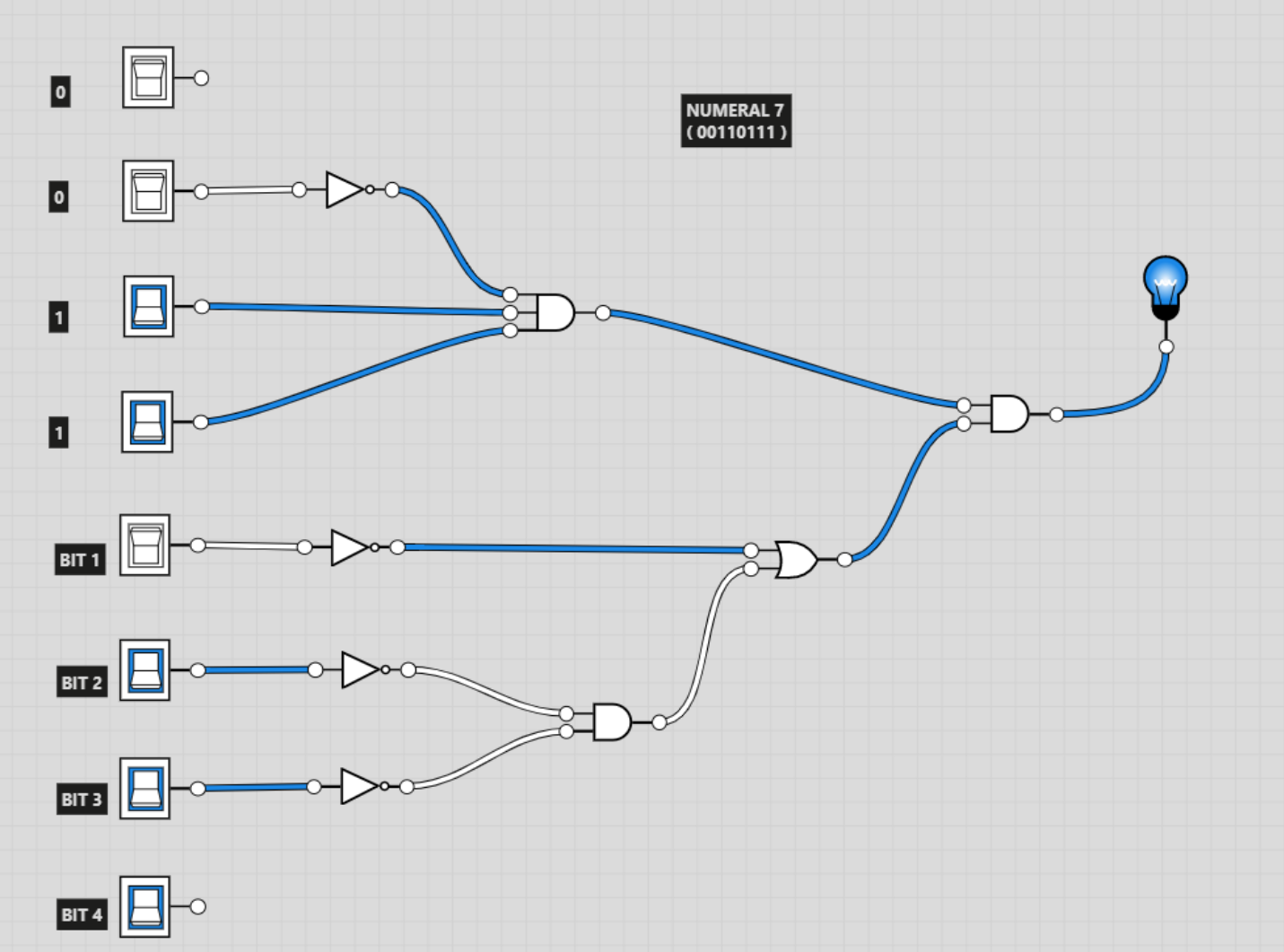
Bit-1	Bit-2	Bit-3	Bit-4	Numbers	Truth Value
0	0	0	0	0	1
0	0	0	1	1	1
0	0	1	0	2	1
0	0	1	1	3	1
0	1	0	0	4	1
0	1	0	1	5	1
0	1	1	0	6	1
0	1	1	1	7	1
1	0	0	0	8	1
1	0	0	1	9	1

KMAP - 2:-

	CD	→			
AB		**00**	**01**	**11**	**10**
↓	**00**	1	1	1	1
	01	1	1	1	1
	11	0	0	0	0
	10	1	1	0	0

Final Simplified Boolean Equation derived using **KMAP** : A' + B'C'

In Block-1 we used four NOT gates , three AND gates and one OR gates and we assembled them in the following way.



https://logic.ly/

The Boolean Expression is independent of the last bit (Bit 4).

After deriving the final boolean expression we tested it on an online simulator to verify it.

BLOCK - 3 (Capital Letter)

We analyzed the binary values of all the Capital letters and we saw a similar pattern, that the second and third bit values were the same(0 , 1) for all the capital letters but there was another thing that we observed, that this pattern is also valid for a few special characters ([, ^ , _ , @ ,etc.) . We tackled this problem by deriving a **Boolean Expression** using KMAP for the last five bits. To derive the Boolean Expression from KMAPS ,

KMAP-3:-

	CD	→							
AB		**000**	**001**	**011**	**010**	**110**	**111**	**101**	**100**
↓	**00**	0	1	1	1	1	1	1	1
	01	1	1	1	1	1	1	1	1
	11	1	1	0	1	0	0	0	0
	10	1	1	1	1	1	1	1	1

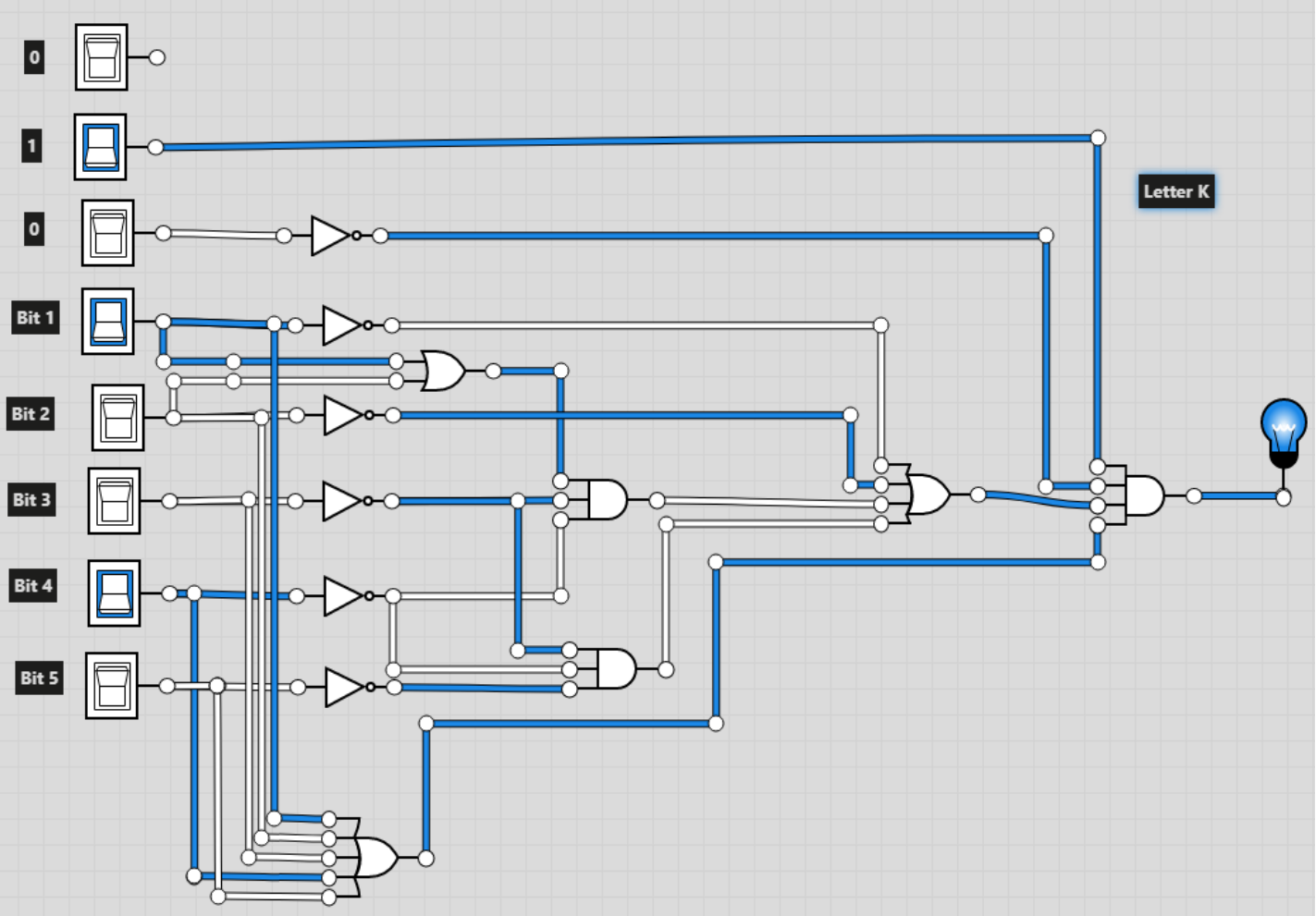
TRUTH TABLE-3:-

	BIT 1	BIT 2	BIT 3	BIT 4	BIT 5	OUTPUT
@	0	0	0	0	0	0
A	0	0	0	0	1	1
B	0	0	0	1	0	1
C	0	0	0	1	1	1
D	0	0	1	0	0	1
E	0	0	1	0	1	1
F	0	0	1	1	0	1
G	0	0	1	1	1	1
H	0	1	0	0	0	1
I	0	1	0	0	1	1
J	0	1	0	1	0	1
K	0	1	0	1	1	1
L	0	1	1	0	0	1
M	0	1	1	0	1	1
N	0	1	1	1	0	1
O	0	1	1	1	1	1
P	1	0	0	0	0	1
Q	1	0	0	0	1	1
R	1	0	0	1	0	1
S	1	0	0	1	1	1
T	1	0	1	0	0	1
U	1	0	1	0	1	1
V	1	0	1	1	0	1
W	1	0	1	1	1	1
X	1	1	0	0	0	1
Y	1	1	0	0	1	1
Z	1	1	0	1	0	1
[	1	1	0	1	1	0
\	1	1	1	0	0	0
]	1	1	1	0	1	0
^	1	1	1	1	0	0
_	1	1	1	1	1	0

The Original **Boolean Expression**: A'B'(C + C'(D + D'E)) + A'B + AB' + ABC'(D' + DE)

Final Simplified Boolean Equation derived using **KMAP:** A' + B' + C'D'(A + B) + C'DE'

In Block-1 we used six NOT gates , three AND gates and three OR gates and we assembled them in the following way.



https://logic.ly/

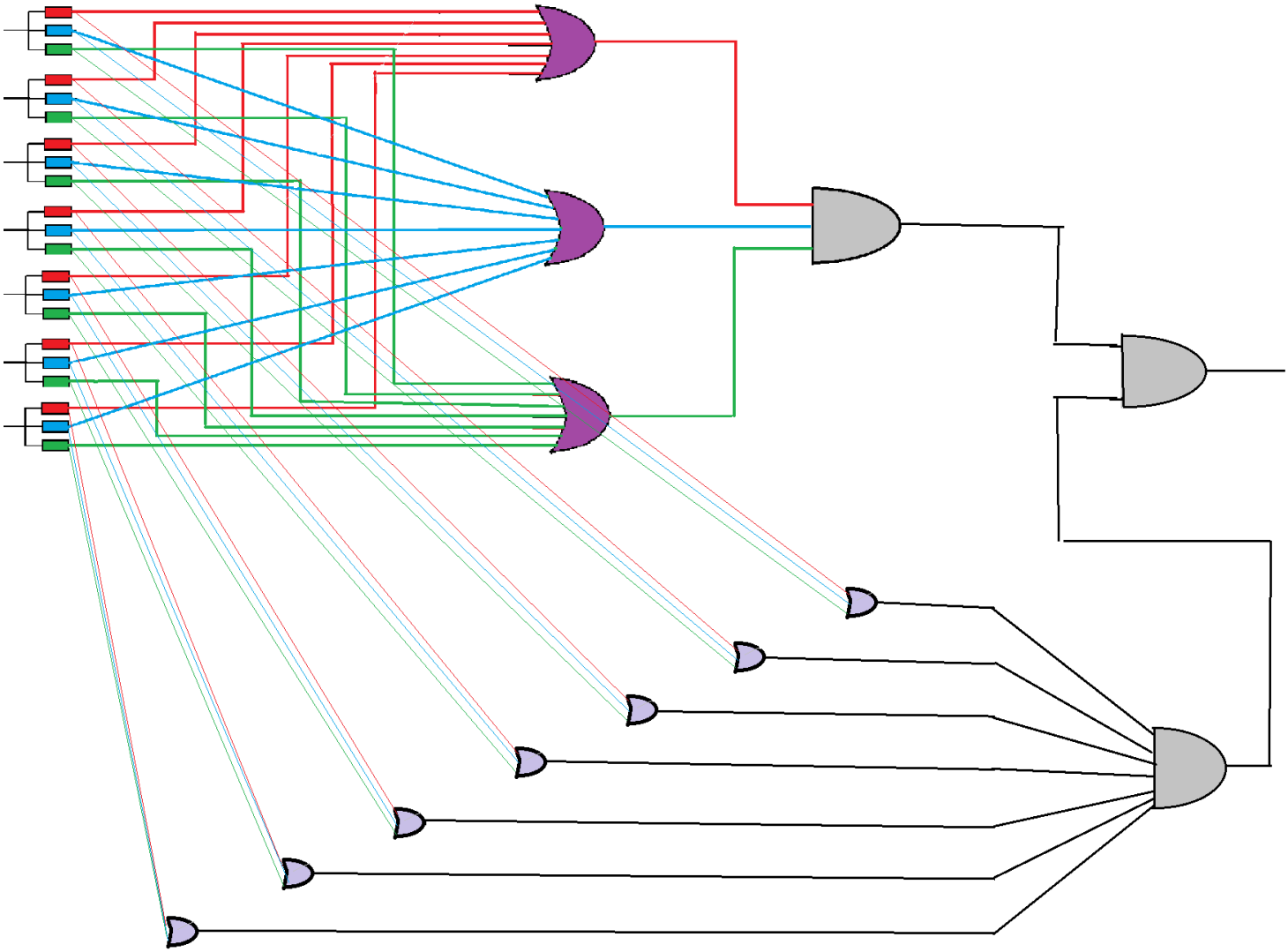
Binary value of K is (01001011).

We linked the last 5 bits to a OR gate to solve the issue wherein our block was giving 1 as output for a special character (@).

After deriving the final boolean expression we tested it on an online simulator to verify it.

MODULAR CIRCUIT DESIGN

Block-1, Block-2, Block-3 have been represented with the colors red, blue and green respectively. The seven characters of the password have their own unique ASCII binary sequence that is taken as input, which is sent into all the three blocks for verification individually. We passed the outputs from all the seven block-1s, block-2s and block-3s through three separate OR gates. This is to ensure that atleast one of the seven block-1s, atleast one of the seven block-2s and atleast one of the seven block-3s is true. These three OR gates are given as inputs to a 3-input AND gate to make sure that all the three OR gates have true outputs. Parallel to this, we have also passed the three blocks assigned to each of the characters through another 3-input OR gate separately to ensure that no other characters apart from those assigned to our group are accepted. Finally, the output from the first AND gate and the outputs from the seven 3-input OR gates is sent as input to a 8-input AND gate. If the output of this gate is 1, then the password is valid, else it is invalid.



RESULT

Through this project, we were able to concretize our conceptual knowledge of logic gates, Karnaugh maps, and Boolean algebra. We were able to appreciate the engineering effort which goes into designing appliances.

From this project, we were able to learn and study the design and working of logic circuits.