University of Colorado Boulder
ECEE Department

ECEN 2350 - Digital Logic - Fall 2023
Location: Engineering Center, ECCR 1B40, MWF 1:25PM - 2:15PM
Instructor: Dr. Mona ElHelbawy
Lab #1
Lab Title: Lock box
Date of Experiment: September 16th, 2023
Names: Connor Sorrell

Section 1

Description:

- This circuit works as a lock box with a four bit input. It works with boolean logic gates; entering the correct four bit input grants a 1, otherwise it outputs a 0. In this case, the circuit was used to light an LED, however its broad applications are infinite.

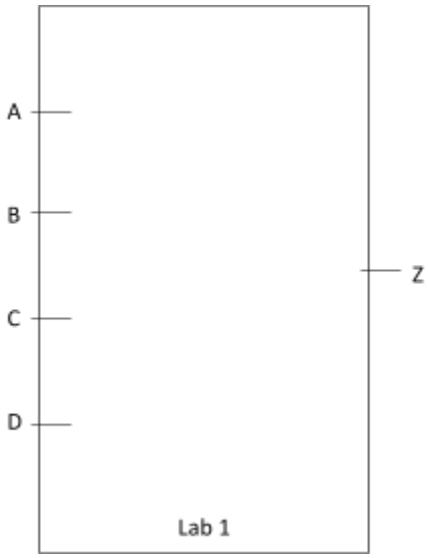

Figure 1.0: Black Box Diagram showing the 4 inputs (A,B,C,D) and single output, Z.

Truth Table:

- This table shows the boolean logic and function values for each combination of inputs. As seen, the lock opens for inputs 5,6,7,9,10,11,13,14,15. This is where the combination of the four inputs results in a 1. Everywhere else, the result is a 0 and the lock box will not open.

index	A	B	C	D	Z	SOP minterm	SOP maxterm
0	0	0	0	0	0		(A+B+C+D)
1	0	0	0	1	0		$(A+B+C+\overline{D})$
2	0	0	1	0	0		$(A+B+\overline{C}+D)$
3	0	0	1	1	0		$(A+B+\overline{C}+\overline{D})$
4	0	1	0	0	0		$(A+\overline{B}+C+D)$

5	0	1	0	1	1	$A\overline{B}\overline{C}D$	
6	0	1	1	0	1	$A\overline{B}CD\overline{\ }$	
7	0	1	1	1	1	$A\overline{B}CD$	
8	1	0	0	0	0		$(\overline{A}+B+C+D)$
9	1	0	0	1	1	$AB\overline{C}D$	
10	1	0	1	0	1	$A\overline{B}CD\overline{\ }$	
11	1	0	1	1	1	$AB\overline{C}D$	
12	1	1	0	0	0		$(\overline{A}+\overline{B}+C+D)$
13	1	1	0	1	1	$AB\overline{C}D$	
14	1	1	1	0	1	$ABCD\overline{\ }$	
15	1	1	1	1	1	ABCD	

Figure 1.0.1: Truth table representing the boolean logic present in the circuit

Compact minterm:

- $Z(A,B,C,D) = \sum(5,6,7,9,10,11,13,14,15)$
  This is obtained from the truth table in every combination that results in a 1.
  $= A\overline{B}\overline{C}D + A\overline{B}CD\overline{\ } + A\overline{B}CD + AB\overline{C}D + A\overline{B}CD\overline{\ } + AB\overline{C}D + ABC\overline{D} + ABCD\overline{\ } + ABCD$

Compact maxterm:

- $Z(A,B,C,D) = \prod(0,1,2,3,4,8,12)$
  This is obtained from the truth table in every combination that results in a 0.
  $= (A+B+C+D) + (A+B+C+\overline{D}) + (A+B+\overline{C}+D) + (A+B+\overline{C}+\overline{D}) + (A+\overline{B}+C+D) + (\overline{A}+B+C+D) + (\overline{A}+\overline{B}+C+D)$

Minimal SOP:

- Z = (C+D)(A+B)

Minimal POS:

- Z = AC + AD + BC + BD

A method of checking these equations is making sure they are equal. This can be done with simple algebra and expanding.

- AC + AD + BC + BD = (C+D)(A+B) = CA + CB + DA + DB

These equations are obtained from simplification of the K-Map representing the truth table (figure 1.0.1)

K-map:

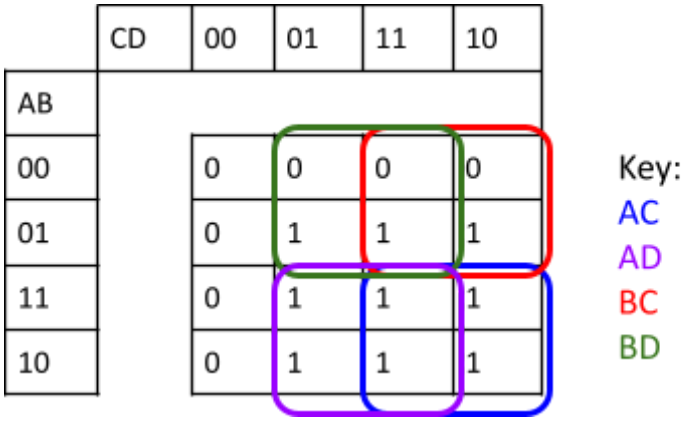

Figure 1.0.2: Karnaugh map representing the truth table, method to simplify boolean expressions.

Using DeMorgans theorem, the NOR/NOR implementation can be found by performing basic boolean algebra on the minimal POS expression.

Taking the minimal POS, $Z = (C+D)(A+B) = \overline{\overline{(C+D)(A+B)}} = \overline{\overline{(C+D)} + \overline{(A+B)}}$

Section 2

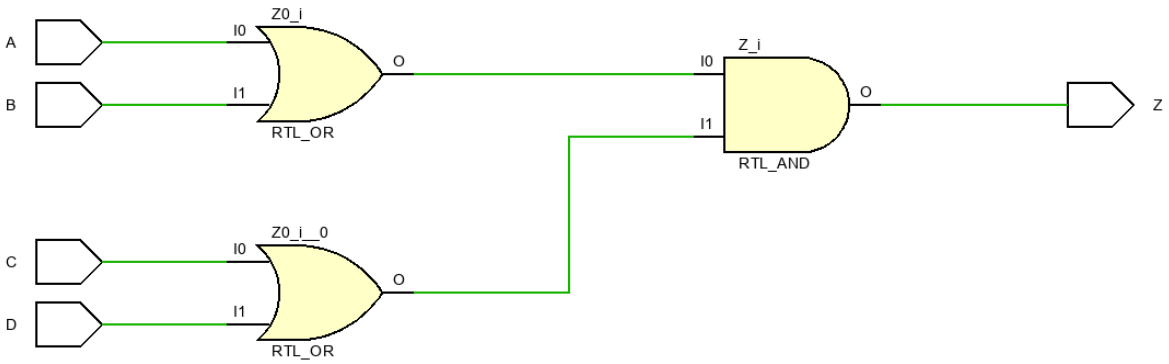

Figure 1.2.0: Circuit schematic for the POS circuit implementation

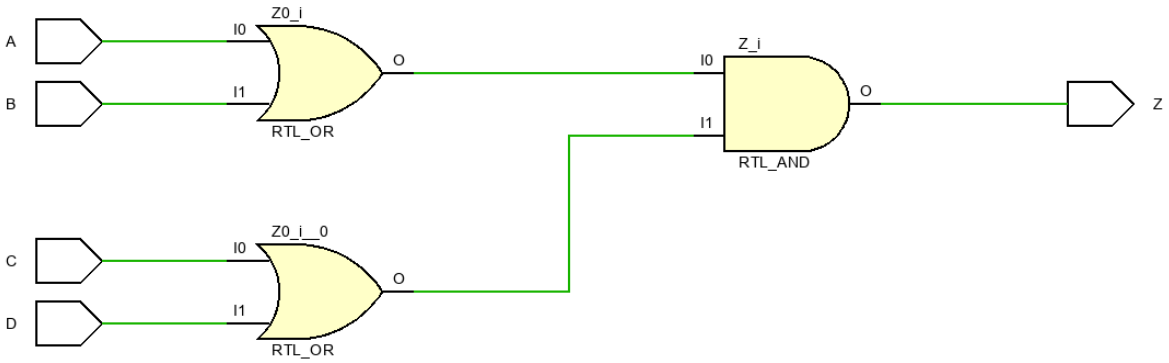

Figure 1.2.1: Circuit schematic for the NOR/NOR circuit implementation

Section 3

Simulated Timing Diagram:

The following screenshot showcases the timing diagram, also known as the simulation waveform, which captures all 16 test cases. It can be seen that the values in which the waveform shows 1 also happen to be the values in which a valid input combination is inputted into the lock box. Therefore, the simulation looks exactly as expected.

It matches the truth table, and shows the 4 inputs [1],[2],[3],[4], and each possible value, and the outcome.

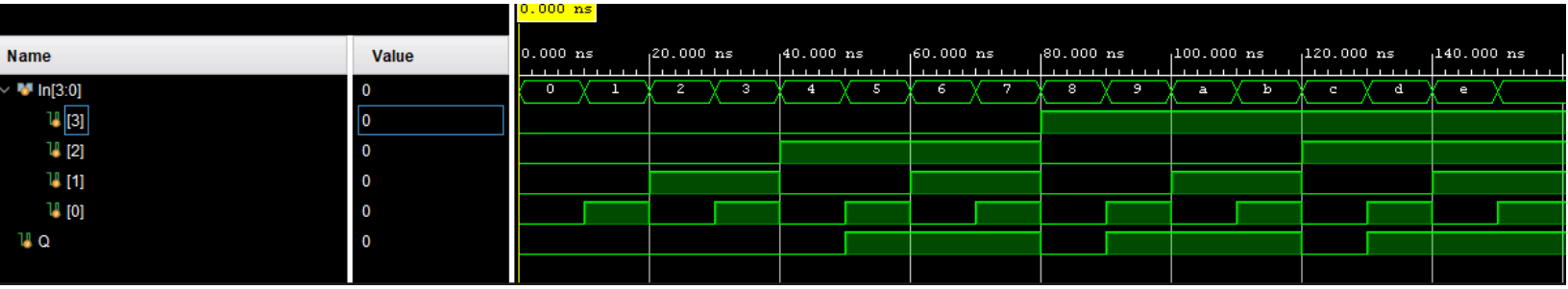

Figure 1.3.0: This figure shows the simulation results of the output Q, after each possible input.

Section 4

```
module lab_1(
    input A,
    input B,
    input C,
    input D,
    output Z
    );
    assign Z = (A|B) & (C|D);

endmodule
```

Figure 1.4.0: Design code for the minimal POS expression

```
module lab_1(
    input A,
    input B,
    input C,
    input D,
    output Z
    );
    assign Z = ~(~(A|B) | ~ (C|D));

endmodule
```

Figure 1.4.1: Design code for the NOR/NOR implementation

```
module labl_sim();
logic [3:0] In;
logic Z;
lab_1 labl_inst(.A(In[3]),.B(In[2]),.C(In[1]),.D(In[0]),.Z(Z));

initial
begin
    for(int i = 0; i < 16; i++)
        begin
            In = i;
            #10;
        end
    end
endmodule
```

Figure 1.4.2: Simulation (test bench) Code including all 16 test cases