

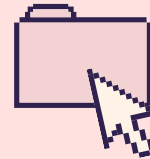


Gates:

NOT

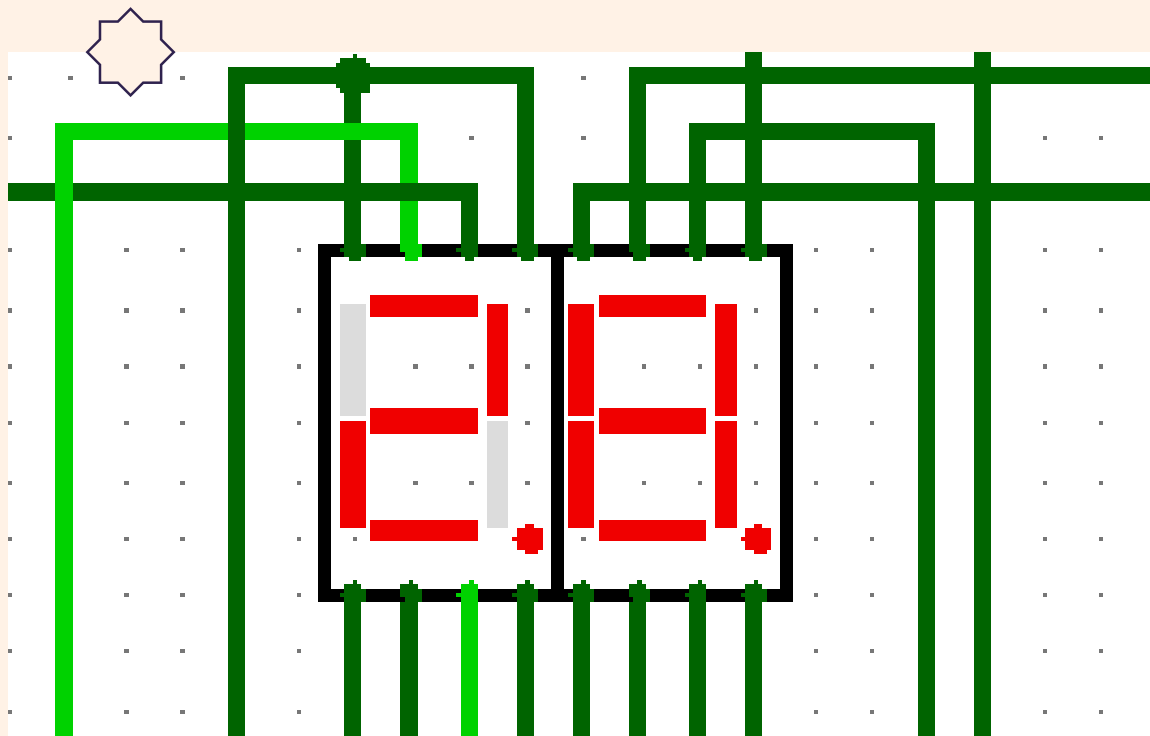
AND

OR



Boolean Algebra Project

Decoders are examined using truth tables and the simplified expressions of the respective K-Maps. Then, a schematic is made on Logisim using logic gates based on the expressions.



Written By: MAHEEN SHOAIB
TEJ 4M0
MR. DI IORIO





Gates:

NOT

AND

OR

Table of Contents



Summary	3
Purpose	3
Testing	4
Process	5
Process Work	6
Truth Table	7
Examining the Truth Table	7
Karnaugh Maps for Display #1	8
Karnaugh Maps for Display #2	9
Simplified Expressions	10
Logism Circuit	11





Gates:

NOT

AND

OR

Purpose

EXAMINE THE USE OF LOGIC GATES IN RELATION TO DECODERS.

In this summative, a common anode display of two digits, capable of displaying any value of 28 and greater was created. I chose the number 28. ABCD are represented in binary where 0001 evaluates to 1, in-turn, 29 (28+1) is displayed. When ABCD are 0010, the value evaluates to 2, in-turn, 30 (28+2) is displayed. Like this, when ABCD are 1111, the value evaluates to 15, in-turn, 43 (28+15) is displayed. In essence, D is 2^0 , C is 2^1 , B is 2^2 , and d is 2^3 . After the truth table is created, their expressions are determined through Karnaugh maps. Then, the expressions are used to make a circuit with two seven segment displays.

Finally, a fifth input, known as the enable bit, is added in the circuit on Logism. When this bit is 0, the entire circuit turns off. In relation, when the bit is 1, the entire circuit stays on without any effect on the rest of the bits.



Testing

THE DISPLAYS ARE NOT ACTIVE ON HIGH.

When 28 is displayed, segment c1 and f1 should be turned on (1). So, the respective outputs of the expressions of c1 and f1 should evaluate to 1. If this is done correct, 28 should be displayed. One can use this concept with the rest of the numbers up until 43 for testing purposes.

In addition, when bit e is on (1), the circuit should be on. In contrast, if bit e is off (0), the circuit should turn off.

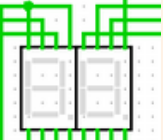
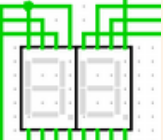
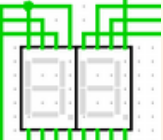
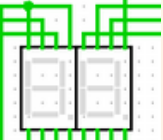
While testing, every number from 28 to 43 was not working, so it was vital for my circuit to be easy to troubleshoot. To ensure this, I first added text to my circuit to label every aspect. The first display is referred to as s1 and the second display is referred to as s2.

Similar to this, each of the bits are labelled: A, B, C, D, E. Then, after every expression was complete, I indicated what segment from the display each of the expressions were referring to.

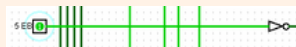



Process

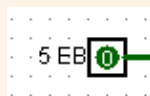
After the truth table was complete, I used the values in the segments to construct Karnaugh maps, which I then used to derive expressions from as learnt in class. Then, I used those expressions to make a circuit on Logism with respect to the two seven-segment displays.

Then, I had to find a way to turn the entire circuit off with bit e. To do this, I first examined the circuit and determined how the entire circuit could turn off with just one bit. Since the display's are not active on high, that means that all of the inputs going into the displays would need to be 1 for it to turn off  (bright green). Knowing this, an and gate can be  used because when the input's are both 1,  the output will be 1 as well, 

allowing the respective output points to be on, turning the displays off. However, since bit e is also directly going into display S1 at outputs DP, G, and B (because they are always on), a 0 would be needed for it to be on when bit e is on. So, bit e can have a not gate at the beginning.



This conflicts with our original and gates. Since bit e will now be 0 after the not gate and the outputs needing to be on will be 1, an or gate can be used because the or gate would be 1 when either input is 1.  So, if bit e were to turn off, the not gate would turn bit e to 1 and since each respective output gate will have an or gate, all of their outputs will be 1, allowing both of the displays to turn off.





Gates:

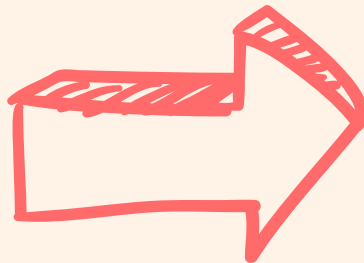
NOT

AND

OR

Truth Table

A	B	C	D	Y	a	b	c	d	e	f	g	a	b	c	d	e	f	g
0	0	0	0	28	0	0	1	0	0	1	0	0	0	0	0	0	0	0
0	0	0	1	29	0	0	1	0	0	1	0	0	0	0	0	1	0	0
0	0	1	0	30	0	0	0	0	1	1	0	0	0	0	0	0	0	1
0	0	1	1	31	0	0	0	0	1	1	0	1	0	0	1	1	1	1
0	1	0	0	32	0	0	0	0	1	1	0	0	0	1	0	0	1	0
0	1	0	1	33	0	0	0	0	1	1	0	0	0	0	0	1	1	0
0	1	1	0	34	0	0	0	0	1	1	0	1	0	0	1	1	0	0
0	1	1	1	35	0	0	0	0	1	1	0	0	1	0	0	1	0	0
1	0	0	0	36	0	0	0	0	1	1	0	0	1	0	0	0	0	0
1	0	0	1	37	0	0	0	0	1	1	0	0	0	0	1	1	1	1
1	0	1	0	38	0	0	0	0	1	1	0	0	0	0	0	0	0	0
1	0	1	1	39	0	0	0	0	1	1	0	0	0	0	0	1	0	0
1	1	0	0	40	1	0	0	1	1	0	0	0	0	0	0	0	0	1
1	1	0	1	41	1	0	0	1	1	0	0	1	0	0	1	1	1	1
1	1	1	0	42	1	0	0	1	1	0	0	0	0	1	0	0	1	0
1	1	1	1	43	1	0	0	1	1	0	0	0	0	0	0	1	1	0





Gates:

NOT

AND

OR

Karnaugh Maps for Display 1

Segment 1: a		A'B'	A'B	AB	AB'
		00	01	11	10
C'D'	00	0	0	1	0
C'D	01	0	0	1	0
CD	11	0	0	1	0
CD'	10	0	0	1	0

Segment 1: b		A'B'	A'B	AB	AB'
		00	01	11	10
C'D'	00	0	0	0	0
C'D	01	0	0	0	0
CD	11	0	0	0	0
CD'	10	0	0	0	0

Segment 1: c		A'B'	A'B	AB	AB'
		00	01	11	10
C'D'	00	1	0	0	0
C'D	01	1	0	0	0
CD	11	0	0	0	0
CD'	10	0	0	0	0

Segment 1: d		A'B'	A'B	AB	AB'
		00	01	11	10
C'D'	00	0	0	1	0
C'D	01	0	0	1	0
CD	11	0	0	1	0
CD'	10	0	0	1	0

Segment 1: e		A'B'	A'B	AB	AB'
		00	01	11	10
C'D'	00	0	1	1	1
C'D	01	0	1	1	1
CD	11	1	1	1	1
CD'	10	1	1	1	1

Segment 1: f		A'B'	A'B	AB	AB'
		00	01	11	10
C'D'	00	1	1	0	1
C'D	01	1	1	0	1
CD	10	1	1	0	1
CD'	11	1	1	0	1

Segment 1: g		A'B'	A'B	AB	AB'
		00	01	11	10
C'D'	00	0	0	0	0
C'D	01	0	0	0	0
CD	11	0	0	0	0
CD'	10	0	0	0	0



Gates:

NOT

AND

OR

Karnaugh Maps for Display 2

Segment 2: a		A'B'	A'B	AB	AB'
		00	01	11	10
C'D'	00	0	0	0	0
C'D	01	0	0	1	0
CD	11	1	0	0	0
CD'	10	0	1	0	0

Segment 2: b		A'B'	A'B	AB	AB'
		00	01	11	10
C'D'	00	0	0	0	1
C'D	01	0	0	0	0
CD	11	0	1	0	0
CD'	10	0	0	0	0

Segment 2: c		A'B'	A'B	AB	AB'
		00	01	11	10
C'D'	00	0	1	0	0
C'D	01	0	0	0	0
CD	11	0	0	0	0
CD'	10	0	0	1	0

Segment 2: d		A'B'	A'B	AB	AB'
		00	01	11	10
C'D'	00	0	0	0	0
C'D	01	0	0	1	1
CD	11	1	0	0	0
CD'	10	0	1	0	0

Segment 2: e		A'B'	A'B	AB	AB'
		00	01	11	10
C'D'	00	0	0	0	0
C'D	01	1	1	1	1
CD	11	1	1	1	1
CD'	10	0	1	0	0

Segment 2: f		A'B'	A'B	AB	AB'
		00	01	11	10
C'D'	00	0	1	0	0
C'D	01	0	1	1	1
CD	11	1	0	1	0
CD'	10	0	0	1	0

Segment 2: g		A'B'	A'B	AB	AB'
		00	01	11	10
C'D'	00	0	0	1	0
C'D	01	0	0	1	1
CD	11	1	0	0	0
CD'	10	1	0	0	0

Gates:NOTANDOR

Simplified Expressions

S1 A: $y = AB$

S1 B: $y = 0$

S1 C: $y = A'B'C'$

S1 D: $y = AB$

S1 E: $y = A + B + C$

S1 F: $y = A' + B'$

S1 G: $y = 0$

S2 A: $y = A'B'CD + A'BCD' + ABC'D$

S2 B: $y = A'BCD + AB'C'D'$

S2 C: $y = A'BC'D' + ABCD'$

S2 D: $y = A'B'CD + A'BCD' + AC'D$

S2 E: $y = D + A'BC$

S2 F: $y = A'B'CD + A'BC' + AC'D + ABC$

S2 G: $y = A'B'C + ABC' + AC'D$



Gates:

NOT

AND

OR

Logism Circuit Design

