60-265-01 Project

by Kim Ficara (104491133) and Zach Shaver (104425444)

1) Implementation

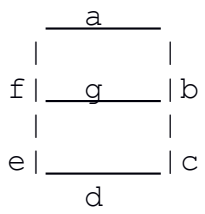
This circuit will take a 4-bit binary number as input and convert it and display as gray code, then convert and display as BCD and 7-segment. It can be very easily expanded to any number of bits.

Assumptions

Gray Code: W X Y Z
(MSB) (LSB)

BCD: B_3 B_2 B_1 B_0
(MSB) (LSB)

7-Segment Display:



This circuit is built for an assumed 4-bit binary input. A 4-bit binary number (1111) produces a 4 bit gray code (1000). which is converted to 5 bit BCD number (1 0101), which produces a 2 digit 7-segment display. If we want a larger amount of bits, we just need to increase the size of the decoder and priority encoder, as well as repeat the MUX arrays for each additional bit or digit produced.

2) Circuit Design

Circuit Design Choices

To satisfy the requirements of the project, we have drawn k-maps and found minimized expressions for a circuit that converts Binary to Grey Code & 7 Segment display. However, our final circuit uses only multiplexers, decoders and encoders to meet the bonus requirement. Skip to page 5 for the final circuit design.

Initial Truth Tables for a Hard-coded Design using Gates

	INPUTS				OUTPUTS																
	Binary				Gray Code				7 Segment (2 digit)												
									Left digit							Right digit					
Dec.	**A**	**B**	**C**	**D**	**W**	**X**	**Y**	**Z**	a_1	b_1	c_1	d_1	e_1	f_1	g_1	a_0	b_0	c_0	d_0	e_0	f_0	g_0
0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	1	1	1	1	1	1	0
1	0	0	0	1	0	0	0	1	1	1	1	1	1	1	0	0	1	1	0	0	0	0
2	0	0	1	0	0	0	1	1	1	1	1	1	1	1	0	1	1	0	1	1	0	1
3	0	0	1	1	0	0	1	0	1	1	1	1	1	1	0	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	1	1	1	1	1	1	0	0	1	1	0	0	1	1
5	0	1	0	1	0	1	1	1	1	1	1	1	1	1	0	1	0	1	1	0	1	1
6	0	1	1	0	0	1	0	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	0	1	0	0	1	1	1	1	1	1	0	1	1	1	0	0	0	0
8	1	0	0	0	1	1	0	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1	0	0	1	1
10	1	0	1	0	1	1	1	1	0	1	1	0	0	0	0	1	1	1	1	1	1	0
11	1	0	1	1	1	1	1	0	0	1	1	0	0	0	0	0	1	1	0	0	0	0
12	1	1	0	0	1	0	1	0	0	1	1	0	0	0	0	1	1	0	1	1	0	1
13	1	1	0	1	1	0	1	1	0	1	1	0	0	0	0	1	1	1	1	0	0	1
14	1	1	1	0	1	0	0	1	0	1	1	0	0	0	0	0	1	1	0	0	1	1
15	1	1	1	1	1	0	0	0	0	1	1	0	0	0	0	1	0	1	1	0	1	1

Minterm Expressions:

Gray Code

W= Σ (8, 9, 10, 11, 12, 13, 14, 15)

X= Σ (4, 5, 6, 7, 8, 9, 10, 11)

Y= Σ (2, 3, 4, 5, 10, 11, 12, 13)

Z= Σ (1, 2, 5, 6, 9, 10, 13, 14)

7 Segment

Left digit

a_1= Σ (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)

b_1= 1

c_1= 1

d_1= Σ (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)

e_1= Σ (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)

f_1= Σ (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)

g_1= 0

Right digit

a_0= Σ (0, 2, 3, 5, 6, 7, 8, 9, 10, 12, 13, 15)

b_0= Σ (0, 1, 2, 3, 4, 7, 8, 9, 10, 11, 12, 13, 14)

c_0= Σ (0, 1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15)

d_0= Σ (0, 2, 3, 5, 6, 8, 10, 12, 13, 15)

e_0= Σ (0, 2, 6, 8, 10, 12)

f_0= Σ (0, 4, 5, 6, 8, 9, 10, 14, 15)

g_0= Σ (2, 3, 4, 5, 6, 8, 9, 12, 13, 14, 15)

K-Maps

7 Segment K-Maps

a_0

AB\CD	00	01	11	10
00	**1**		**1**	**1**
01		**1**	**1**	**1**
11	**1**	**1**	**1**	
10	**1**	**1**		**1**

b_0

AB\CD	00	01	11	10
00	**1**	**1**	**1**	**1**
01	**1**		**1**	
11	**1**	**1**		**1**
10	**1**	**1**	**1**	**1**

c_0

AB\CD	00	01	11	10
00	**1**	**1**	**1**	
01	**1**	**1**	**1**	**1**
11		**1**	**1**	**1**
10	**1**	**1**	**1**	**1**

d_0

AB\CD	00	01	11	10
00	**1**		**1**	**1**
01		**1**		**1**
11	**1**	**1**	**1**	
10	**1**			**1**

e_0

AB\CD	00	01	11	10
00	**1**			**1**
01				**1**
11	**1**			
10	**1**			**1**

f_0

AB\CD	00	01	11	10
00	**1**			
01	**1**	**1**		**1**
11			**1**	**1**
10	**1**	**1**		**1**

g_0

AB\CD	00	01	11	10
00			**1**	**1**
01	**1**	**1**		**1**
11	**1**	**1**	**1**	**1**
10	**1**	**1**		

---

a_1, d_1, e_1, f_1

AB\CD	00	01	11	10
00	**1**	**1**	**1**	**1**
01	**1**	**1**	**1**	**1**
11				
10	**1**	**1**		

b_1, c_1 = 1 *Always on*

g_1 = 0 *Always off*

Gray Code K-Maps

W

AB\CD	00	01	11	10
00				
01				
11	**1**	**1**	**1**	**1**
10	**1**	**1**	**1**	**1**

X

AB\CD	00	01	11	10
00				
01	**1**	**1**	**1**	**1**
11				
10	**1**	**1**	**1**	**1**

Y

AB\CD	00	01	11	10
00			**1**	**1**
01	**1**	**1**		
11	**1**	**1**		
10			**1**	**1**

Z

AB\CD	00	01	11	10
00		**1**		**1**
01		**1**		**1**
11		**1**		**1**
10		**1**		**1**

Simplified Expressions

Gray Code

$W = A$

$X = A'B + AB' = A \oplus B$

$Y = BC' + B'C = B \oplus C$

$Z = C'D + CD' = C \oplus D$

7 Segment

$\{a_1, d_1, e_1, f_1\} = A' + B'C'$

$\{b_1, c_1\} = 1$

$g_1 = 0$

$a_0 = B'D' + AC' + A'C + BD$

$b_0 = D' + AC' + C'D' + A'CD + ACD'$

$c_0 = D + A'C' + A'B + AB' + AC$

$d_0 = B'D' + AC'D' + ABD + BC'D + A'CD' + A'B'C$

$e_0 = A'B'D' + AC'D' + AB'D' + A'CD'$

$f_0 = A'C'D' + A'BC' + AB'C' + BCD' + ACD' + ABC$

$g_0 = A'B'C + BD' + BC' + AC' + AB$

Final Circuit Design

To meet the bonus requirements, as well as make our circuit more flexible, we have decided to use a MUX/decoder/encoder only design instead of the circuit produced by the above K-maps.

Binary to Gray Code Converter

Using a multiplexer for each bit, binary is converted to gray code.

INPUTS				OUTPUTS							
Binary				Gray Code							
					MUX		*MUX*		*MUX*		*MUX*
A	B	C	D	W	*F*	X	*F*	Y	*F*	Z	*F*
0	0	0	0	0	**0**	0	**0**	0	**0**	0	**D**
0	0	0	1	0		0		0		1	
0	0	1	0	0	**0**	0	**0**	1	**1**	1	**~D**
0	0	1	1	0		0		1		0	
0	1	0	0	0	**0**	1	**1**	1	**1**	0	**D**
0	1	0	1	0		1		1		1	
0	1	1	0	0	**0**	1	**1**	0	**0**	1	**~D**
0	1	1	1	0		1		0		0	
1	0	0	0	1	**1**	1	**1**	0	**0**	0	**D**
1	0	0	1	1		1		0		1	
1	0	1	0	1	**1**	1	**1**	1	**1**	1	**~D**
1	0	1	1	1		1		1		0	
1	1	0	0	1	**1**	0	**0**	1	**1**	0	**D**
1	1	0	1	1		0		1		1	
1	1	1	0	1	**1**	0	**0**	0	**0**	1	**~D**
1	1	1	1	1		0		0		0	

Gray Code to BCD Decoder/Encoder

The gray code produced above is sent into a 4-16 decoder. Due to the way decoders work, the bits will be decoded as if the gray code was in regular binary form (d_0=0000, d_1=0001, ..., d_{14}=1110, d_{15}=1111). Then we use a priority encoder to "unscramble" the gray code. One 32-to-5 priority encoder is effectively used as two separate 16-4 encoders (one for each BCD number). The first 16 encoder inputs are used for first 9 decoded outputs, and the next 16 inputs are used for 10-19. In this way we can expand the decoder/encoder combination to however many bits we need, and the patterns stay the same.

4-to-16 Decoder:

INPUTS				OUTPUTS														
Gray Code				Decoded Gray Code														
A	B	C	D	d_0	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}	d_{11}	d_{12}	d_{13}	d_{14}	d_{15}
0	0	0	0	1														
0	0	0	1		1													
0	0	1	1				1											
0	0	1	0			1												
0	1	1	0							1								
0	1	1	1								1							
0	1	0	1						1									
0	1	0	0					1										
1	1	0	0													1		
1	1	0	1														1	
1	1	1	1															1
1	1	1	0															1
1	0	1	0											1				
1	0	1	1												1			
1	0	0	1										1					
1	0	0	0									1						

32-to-5 Priority Encoder:

INPUTS																	OUTPUTS				
Decoded Gray Code																	Encoded BCD				
e_{21}	e_{20}	e_{19}	e_{18}	e_{17}	e_{16}	$e_{15\text{-}10}$	e_9	e_8	e_7	e_6	e_5	e_4	e_3	e_2	e_1	e_0	B_3	B_3	B_2	B_1	B_0
						X										1	0	0	0	0	0
						X									1	X	0	0	0	0	1
						X								1	X	X	0	0	0	1	0
						X							1	X	X	X	0	0	0	1	1
						X						1	X	X	X	X	0	0	1	0	0
						X					1	X	X	X	X	X	0	0	1	0	1
						X				1	X	X	X	X	X	X	0	0	1	1	0
						X			1	X	X	X	X	X	X	X	0	0	1	1	1
						X		1	X	X	X	X	X	X	X	X	0	1	0	0	0
						X	1	X	X	X	X	X	X	X	X	X	0	1	0	0	1
					1	X	X	X	X	X	X	X	X	X	X	X	1	0	0	0	0
				1	X	X	X	X	X	X	X	X	X	X	X	X	1	0	0	0	1
			1	X	X	X	X	X	X	X	X	X	X	X	X	X	1	0	0	1	0
		1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1	0	0	1	1
	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1	0	1	0	0
1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1	0	1	0	1

From the decoder and encoder truth tables, connections are made as follows:

D0 → E0	D4 → E7	D8 → E21	D12 → E8
D1 → E1	D5 → E6	D9 → E20	D13 → E9
D2 → E3	D6 → E4	D10 → E18	D14 → E17
D3 → E2	D7 → E5	D11 → E19	D15 → E16

ex. *Dec* *Gray Code* *BCD*

0 *D0 (0000)* = *E0 (00000)*
. . .
. . .
2 *D3 (0011)* = *E2 (00010)*
. . .
. . .
13 *D11 (1011)* = *E19 (10011)*
. . .
. . .

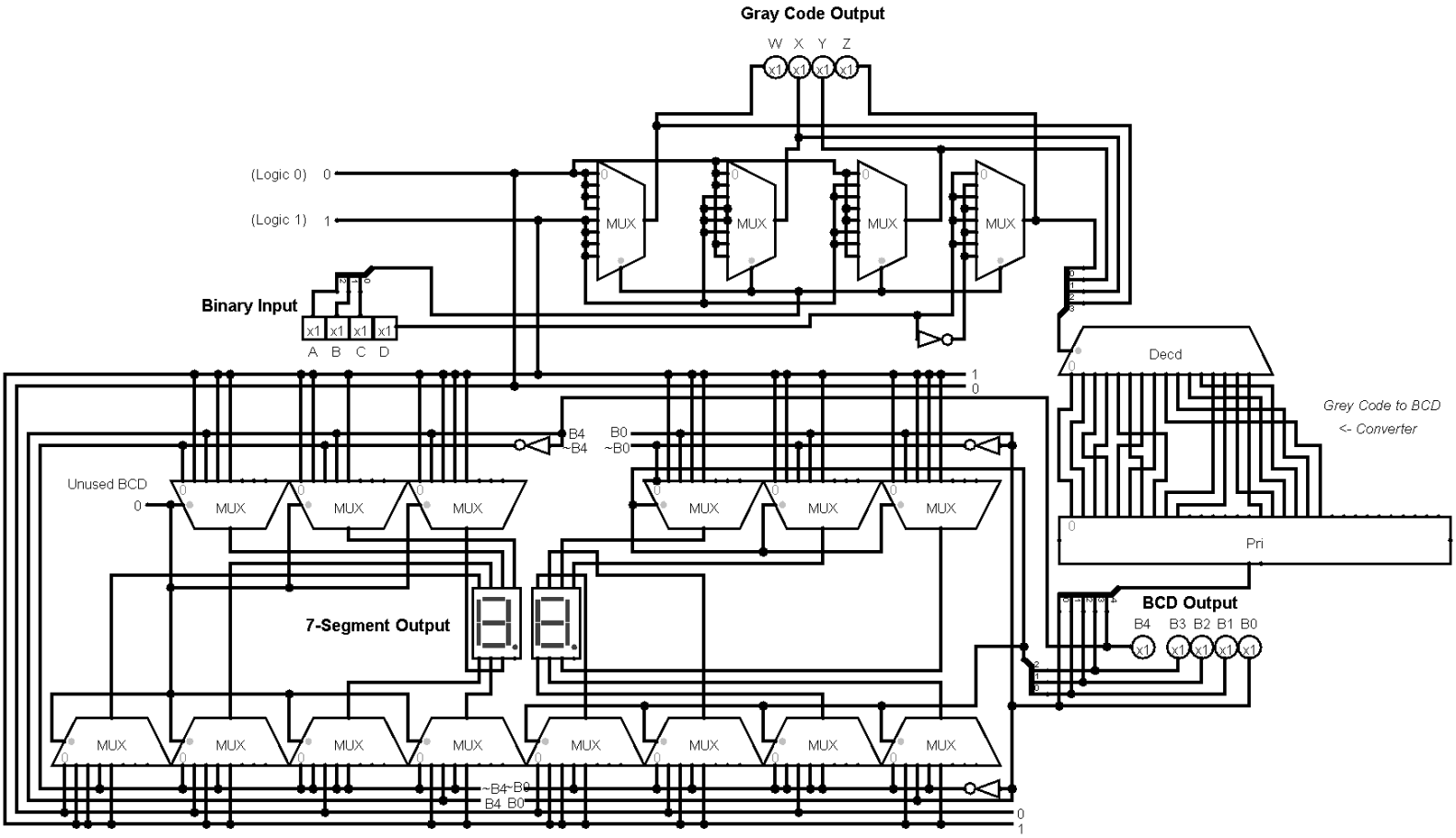
BCD to 7-Segment Converter

Using a multiplexer for each bit, the BCD is converted to 7 segment display. The exact same design is used for each additional digit. Since there are 5 bits of BCD for 4 bits of grey code, the four least significant BCD bits produce the 10^0 digit, and the MSB (B4) becomes the LSB for the next digit (10^1) and is padded with zeros (labelled *"Unused BCD"* in circuit). If there are more BCD digits, there is simply less padding but the design is the same; the LSB is used as the data bit for the MUXs.

INPUTS				OUTPUTS													
Binary				7 Segment													
					MUX		MUX		MUX		MUX		MUX		MUX		MUX
B_3	B_2	B_1	B_0	a_0	F	b_0	F	c_0	F	d_0	F	e_0	F	f_0	F	g_0	F
0	0	0	0	1	$\sim B_0$	1	1	1	1	1	$\sim B_0$	1	$\sim B_0$	1	$\sim B_0$	0	0
0	0	0	1	0		1		1		0		0		0		0	
0	0	1	0	1	1	1	1	0	B_0	1	1	1	$\sim B_0$	0	0	1	1
0	0	1	1	1		1		1		1		0		0		1	
0	1	0	0	0	B_0	1	$\sim B_0$	1	1	0	B_0	0	0	1	1	1	1
0	1	0	1	1		0		1		1		0		1		1	
0	1	1	0	1	1	0	B_0	1	1	1	$\sim B_0$	1	$\sim B_0$	1	$\sim B_0$	1	$\sim B_0$
0	1	1	1	1		1		1		0		0		0		0	
1	0	0	0	1	1	1	1	1	1	1	$\sim B_0$	1	$\sim B_0$	1	1	1	1
1	0	0	1	1		1		1		0		0		1		1	
X	X	X	X	X		X		X		X		X		X		X	

3) Circuit Diagram

Below are snapshots of the final implementation of our circuit. The first is Logisim`s exported image, and the second is a "print screen" of the circuit in action – displaying a 14 in gray code, BCD and 7-segment – with some added colour to highlight the functional areas outlined above. A copy of the circuit file for Logisim is attached (*Bin-to-GC-to-BCD-to-7seg_Logisim.circ*), so that it may be tested.

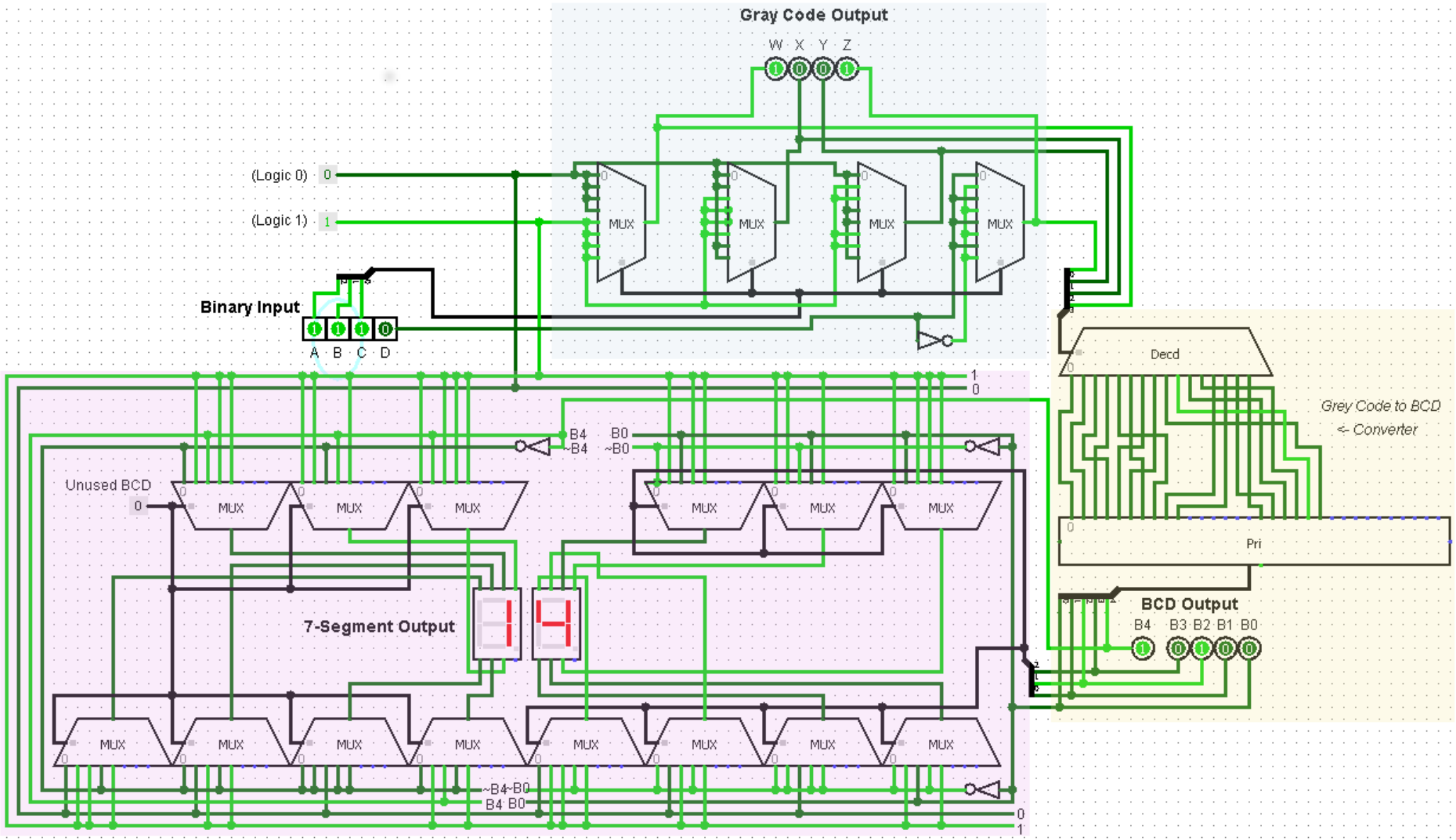


Operation of circuit:

A binary number is input from the user in the top left of the circuit. Logical constants of 1 and 0 are also sent in for use by multiplexers. Multiplexers convert this input to a gray code, which is displayed by LEDs located center-top.

The gray code is split off to a decoder/encoder to be converted to a higher bit BCD number. This BCD number is displayed by LEDs located bottom-right.

Each BCD (4-bit) number is then sent to multiplexers to be converted to a 7-segment display. If there are less than 4 BCD digits, they are padded with 0's for the MUX selection. The combined 7-segment number is displayed in the bottom-left.


Gray Code Output
W X Y Z
(Logic 0) 0
(Logic 1) 1
Binary Input
A B C D
MUX
Decd
Grey Code to BCD
<- Converter
Pri
BCD Output
B4 B3 B2 B1 B0
Unused BCD
0
7-Segment Output
B4
~B4
B0
~B0
~B4~B0
B4 B0