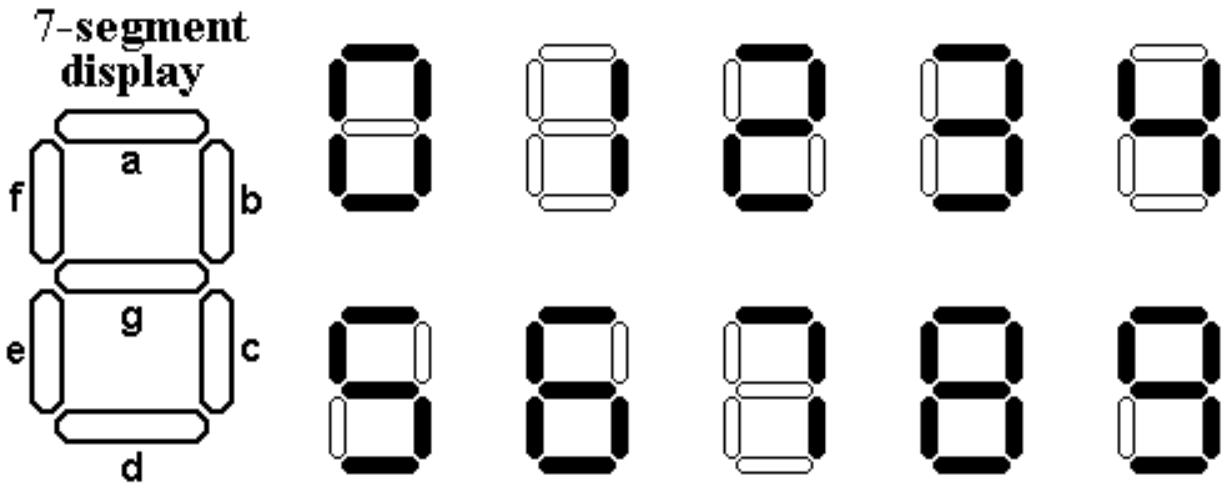
Developing the Gate Level Logic for a Seven-Segment Display

Soyeon Kim

Overview

The seven-segment display is an electronic device used to display decimal numbers (from 0 to 9), where each of the seven segments (labeled “a” through “d”) are either turned on or off to display a certain number. Because we need to display decimal numbers through an electronic device that operates on a binary system, I need to derive a gate level logic.



Methodology

Because the seven-segment display operates on a binary system, I first need to convert the decimal values to binary with a total of 4 placeholders. Binary number system is a number system that represents numbers using only two digits, 0 and 1. Therefore, to represent all 10 decimal number digits through binary, we need 4 place holders ($2^4 = 16, 16 > 9$) in total. The table below shows the conversion from decimal to binary and the segments used to display each of the numbers. Because a seven-segment display can only display decimal numbers from 0 to 9, we label the rest of the binary numbers greater than 1010 as ‘don’t cares’. These are the inputs that don’t affect the output.

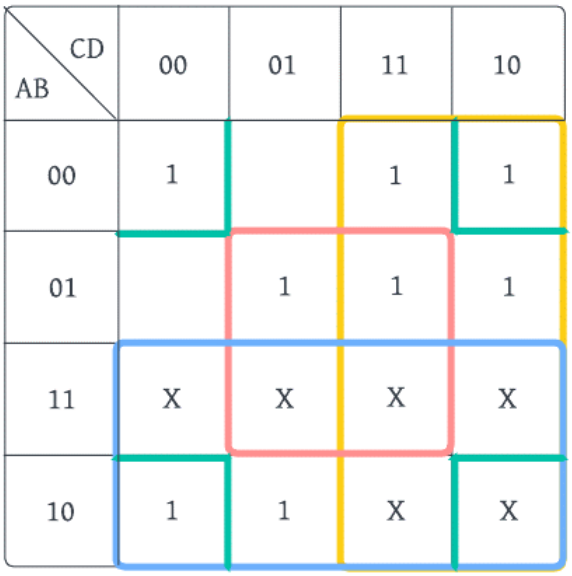
Decimal Numbers	Binary Equivalent (ABCD)	Segments Used
0	0000	a, b, c, d, e, f
1	0001	b, c
2	0010	a, b, d, e, g
3	0011	a, b, c, d, g
4	0100	b, c, f, g
5	0101	a, c, d, f, g
6	0110	a, c, d, e, f, g
7	0111	a, b, c
8	1000	a, b, c, d, e, f, g
9	1001	a, b, c, d, f, g
10	1010	“don’t cares”
11	1011	
12	1100	
13	1101	
14	1110	
15	1111	

Each of the placeholders in binary equivalent expressions are represented through wires labeled A, B, C, and D, and these are the inputs of the system that we can control to display a desired decimal number. ‘0’ and ‘1’ in binary expressions represent ‘on’ and ‘off’ status of the electric signal through each wire. Using the table, I can create a Karnaugh Map for each segment, which will allow me to find the optimized combinations of the gateway logic.

My Karnaugh Maps consist of 4 rows for wires A and B and 4 columns for wires C and D, both labeled 00, 01, 11, and 10. The order of these 4 labels is important because I need to use cell adjacency to group the cells, reducing the complexity of the circuit logic. Then, I label each cell

as ‘1’ if the segment is used in the binary equivalent and ‘X’ for ‘don’t cares.’ For example, segment ‘a’ is used in binary equivalent of 0000, 0010, 0011, 0101, 0110, 0111, 1000, and 1001, so I label equivalent cells as ‘1’ and anything from 1010 are labeled ‘X’. Once labeling is completed, I can group the cells to simplify the logic. To group the cells, I find 2, 4, or 8 adjacent cells in the shape of a rectangle. Because cell adjacency wraps around the Karnaugh Map, each of the four corners can be grouped and edges can connect to edges on the opposite side of the map. After grouping the cells, I can produce a logical expression that represents the logic for the segment as shown below in the Karnaugh Map for segment ‘a’. To create a logical expression, I identify the shared input within a group, separating each shared input by an ‘and’ (represented by multiplication symbol) and groups by an ‘or’ (represented by addition symbol). Larger groups share less common input, so my goal is to create as little groups as possible while grouping them in big blocks. Karnaugh Map for each segment is shown below.

Karnaugh Map for Segment ‘a’

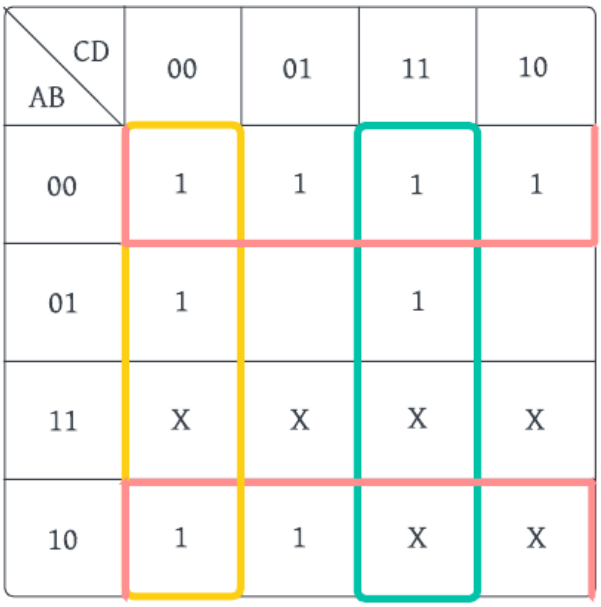


AB \ CD	00	01	11	10
00	1		1	1
01		1	1	1
11	X	X	X	X
10	1	1	X	X

$$A + C + BD + \bar{B}\bar{D}$$

I repeat the same process for all other segments. Below are the Karnaugh Maps and logical expressions for each segment.

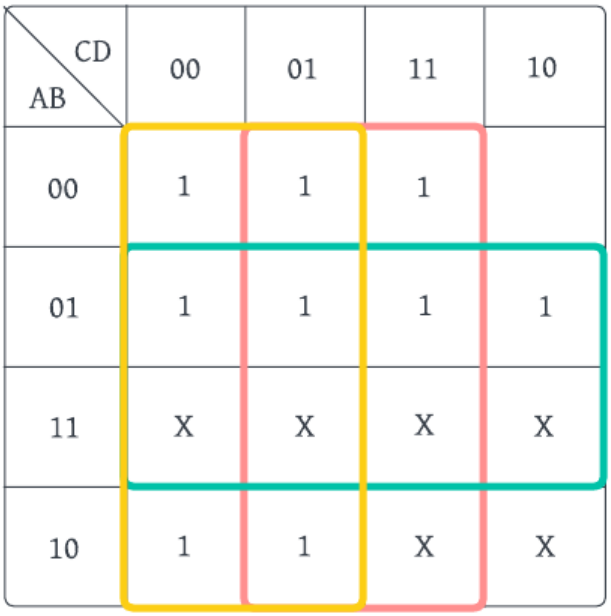
Karnaugh Map for Segment ‘b’



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

$$\bar{B} + \bar{C}\bar{D} + CD$$

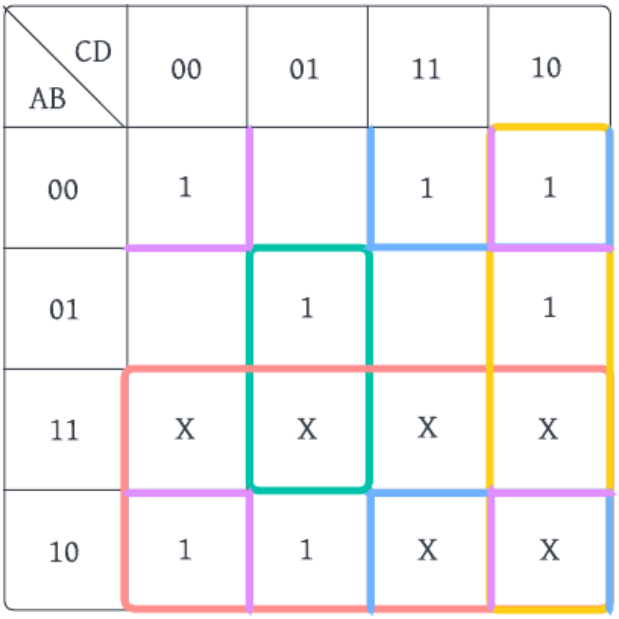
Karnaugh Map for Segment ‘c’



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

$$B + \bar{C} + D$$

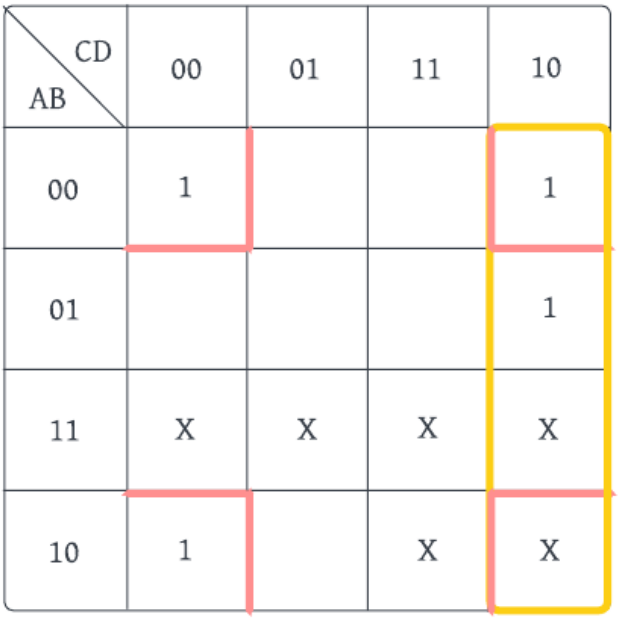
Karnaugh Map for Segment ‘d’



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

$$A + \bar{B}\bar{D} + \bar{B}C + B\bar{C}D + C\bar{D}$$

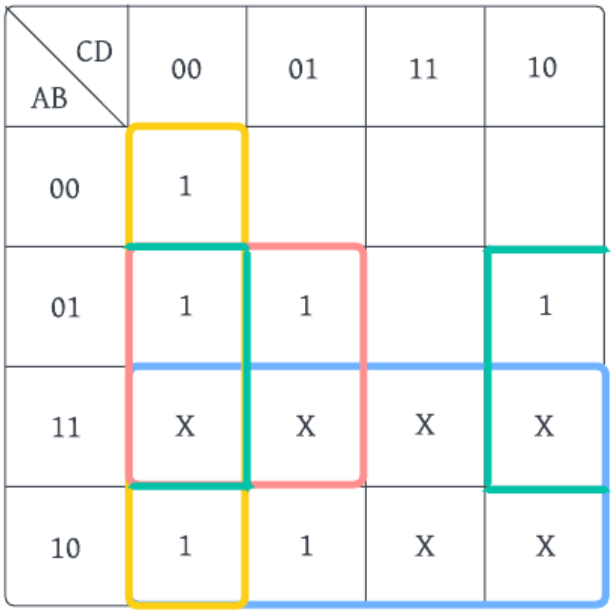
Karnaugh Map for Segment ‘e’



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

$$C\bar{D} + \bar{B}\bar{D}$$

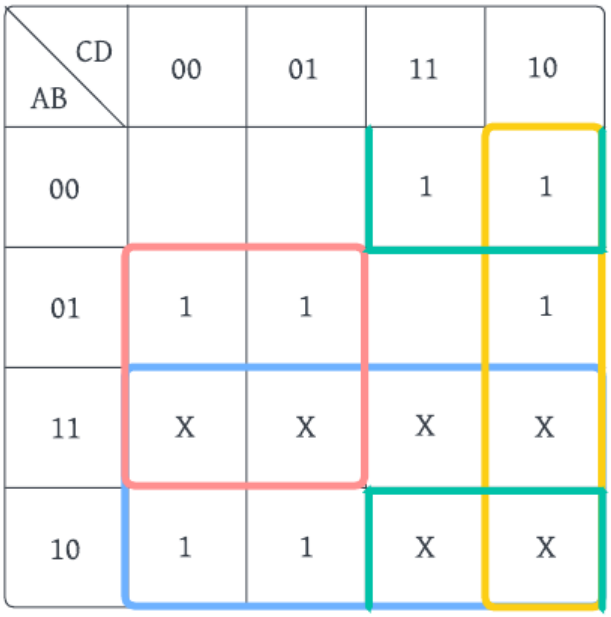
Karnaugh Map for Segment ‘f’



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

$$A + B\bar{D} + \bar{C}\bar{D} + B\bar{D}$$

Karnaugh Map for Segment 'g’

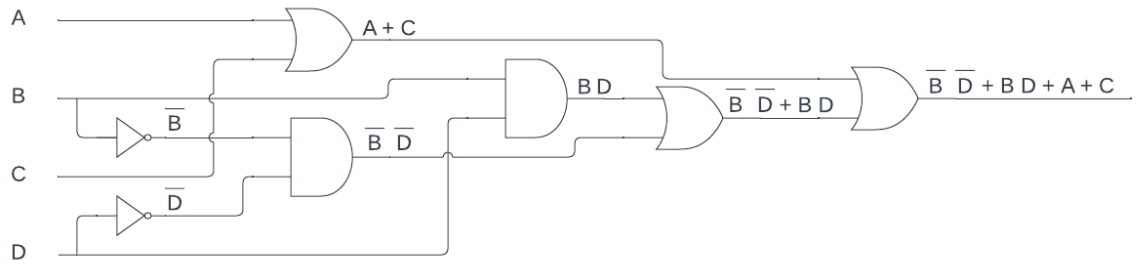


AB \ CD	00	01	11	10
00			1	1
01	1	1		1
11	X	X	X	X
10	1	1	X	X

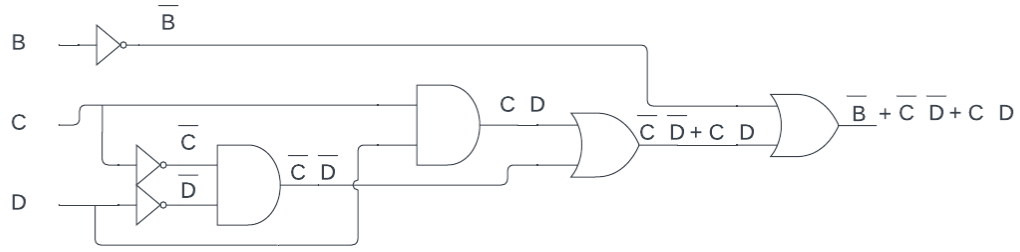
$$A + B\bar{C} + C\bar{D} + \bar{B}C$$

Now, using the logical expressions derived from the Karnaugh Maps, I draw a gateway logic for each segment, as follows.

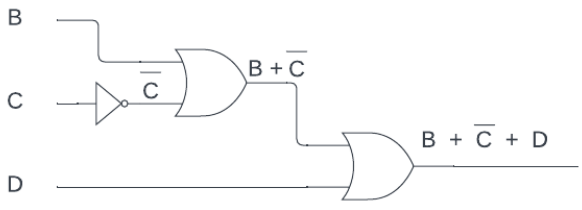
Gateway Logic for Segment ‘a’



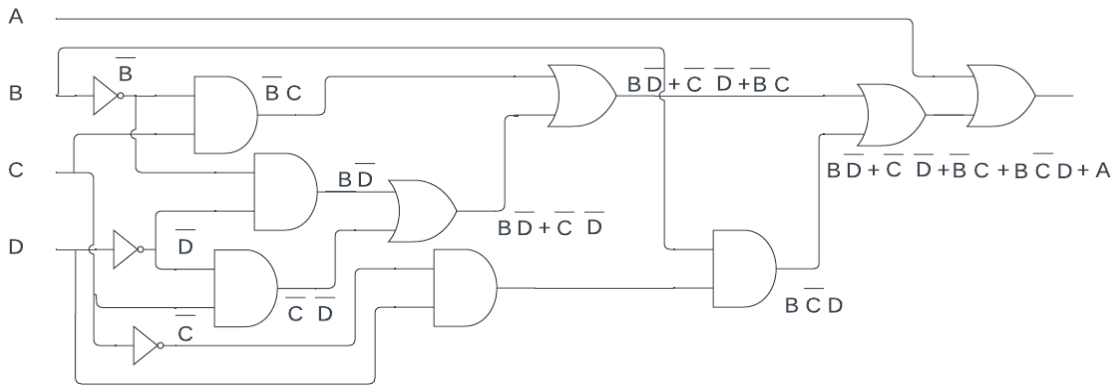
Gateway Logic for Segment ‘b’



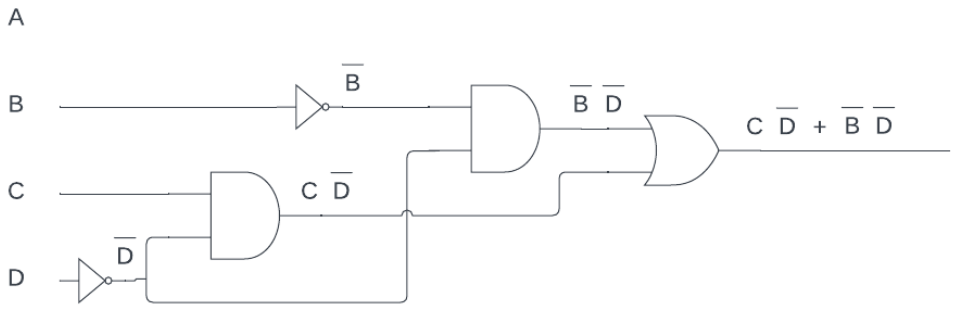
Gateway Logic for Segment ‘c’



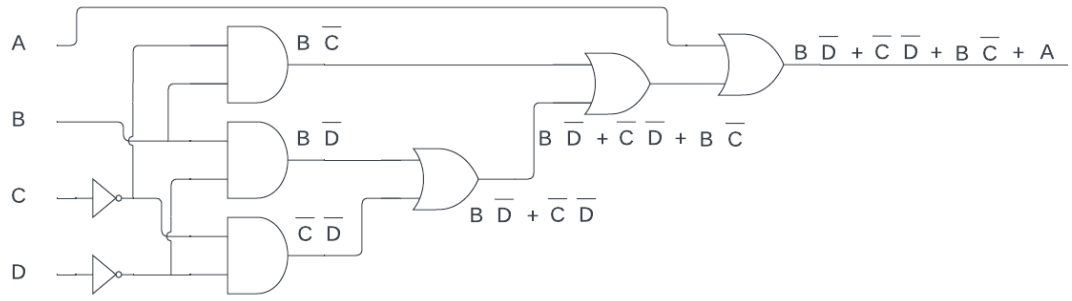
Gateway Logic for Segment 'd’



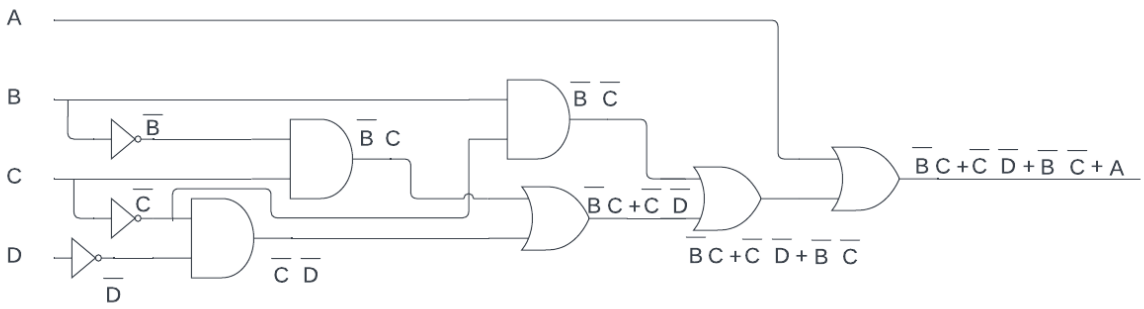
Gateway Logic for Segment ‘e’



Gateway Logic for Segment ‘f’



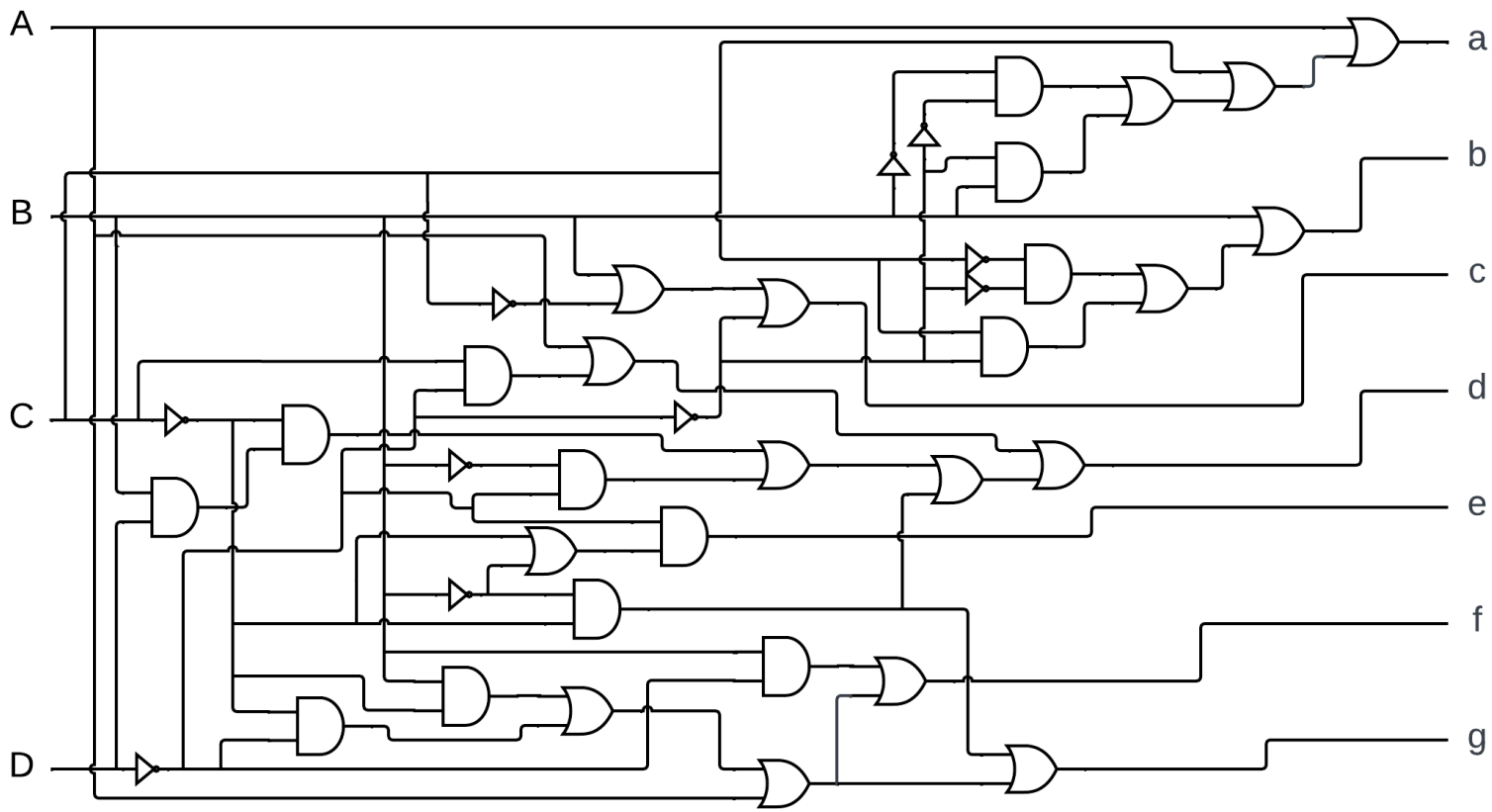
Gateway Logic for Segment ‘g’



Discussion

Using the gateway logic for each segment, I need to develop an overall logic for the entire system. The logic should have the same four inputs (A, B, C, and D) but 7 outputs (a, b, c, d, e, f, g), for each of the segments in the seven-segment display. My goal is to minimize the number of jumps and gates and to create the simplest gateway logic. Most likely, the number of jumps and gates are inversely related, as jumps can be avoided by using more inverters and vice versa. Because the outputs must be in alphabetical order, I position each Boolean combination of input wires near the segment which it is used in. In addition, some of the combinations that are used in

more than one segment are branched off to reduce the number of gates. After a few tries, I was able to reduce the number of jumps to 27 and gates to 39. Using this final gateway logic, I can display all 10 decimal numbers on a seven-segment display.


A
B
C
D
a
b
c
d
e
f
g