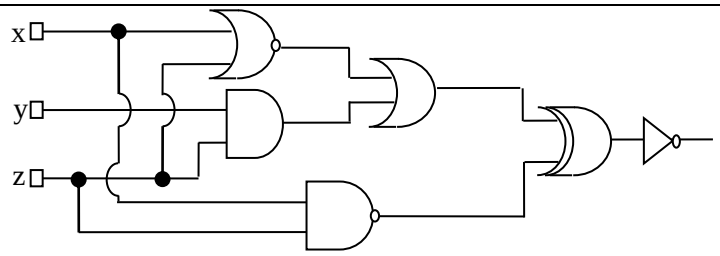
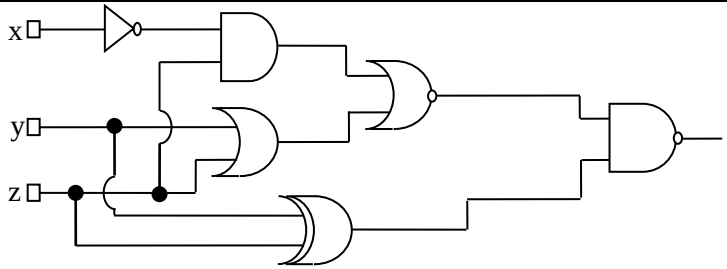
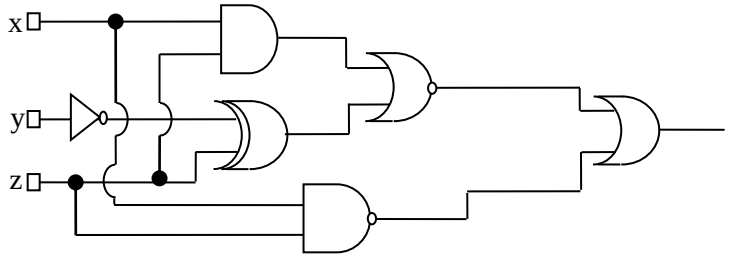
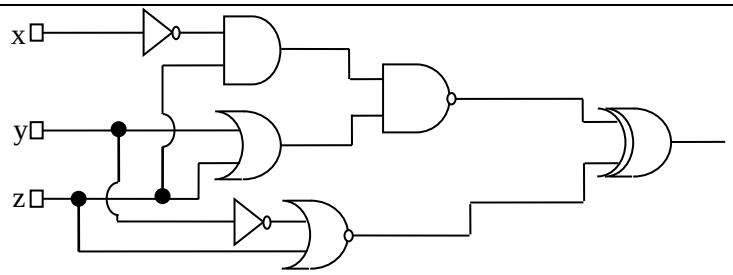
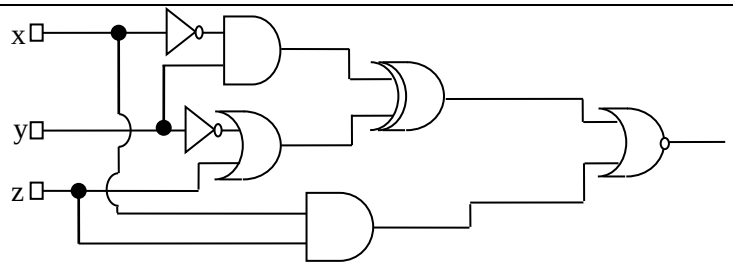
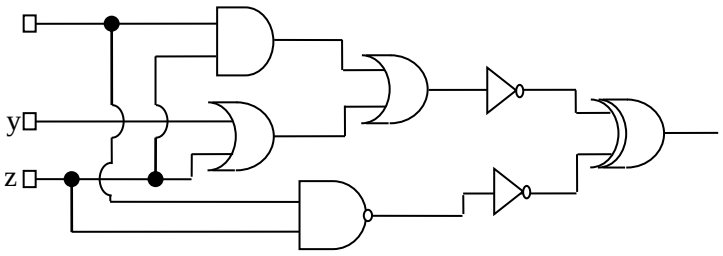
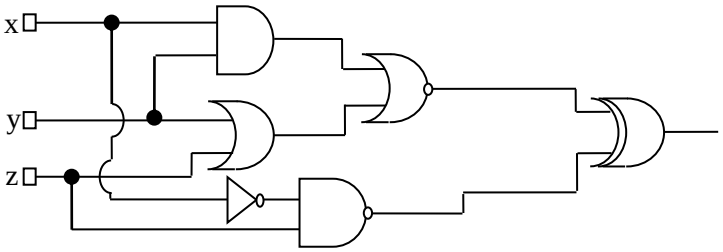
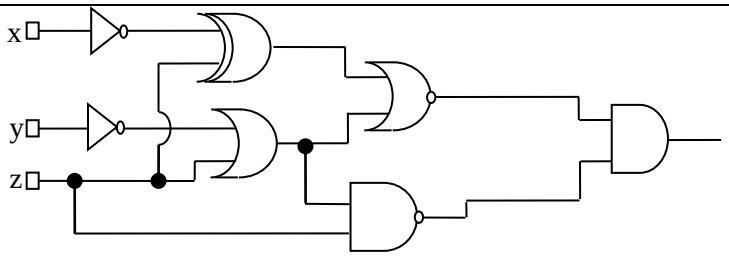
Exercises – Logic circuits

Exercise 8.1.Write the corresponding Boolean function associated to the following logic circuit, simplify it and then draw the simplified circuit using only basic gates:

1.	
2.	
3.	
4.	
5.	

6.	
7	
8.	

Exercise 8.2.
Prove that NOR is a universal gate.

Exercise 8.3.
For each of the following Boolean functions draw the corresponding logic circuit using derived gates, simplify the function and draw the logic circuits associated to all simplified forms of the initial function using only basic gates.

1. $f_1(x, y, z) = x(y \oplus z) \vee y(x \oplus z) \vee x(\bar{y} \downarrow \bar{z}) \vee (x \downarrow y)\bar{z}$;
2. $f_2(x, y, z) = x(y \uparrow z) \vee \bar{x}(\bar{y} \oplus z) \vee y(\bar{x} \oplus \bar{z})$;
3. $f_3(x, y, z) = x(\bar{y} \oplus z) \vee y(\bar{x} \oplus z) \vee \bar{x}(\bar{y} \downarrow z) \vee (\bar{x} \downarrow y)z$;
4. $f_4(x, y, z) = \bar{x}(y \uparrow \bar{z}) \vee x(\bar{y} \oplus z) \vee \bar{y}(\bar{x} \oplus z)$;
5. $f_5(x, y, z) = \bar{x}(y \oplus \bar{z}) \vee \bar{y}(x \oplus z) \vee \bar{x}(y \downarrow z) \vee (\bar{x} \downarrow y)\bar{z}$;
6. $f_6(x, y, z) = x(\bar{y} \uparrow \bar{z}) \vee \bar{x}(y \oplus z) \vee \bar{y}(\bar{x} \oplus z)$;
7. $f_7(x, y, z) = x(y \oplus \bar{z}) \vee y(\bar{x} \oplus z) \vee x(y \downarrow z) \vee (x \downarrow y)\bar{z}$;
8. $f_8(x, y, z) = x(\bar{y} \uparrow z) \vee \bar{x}(\bar{y} \oplus z) \vee y(x \oplus \bar{z})$.

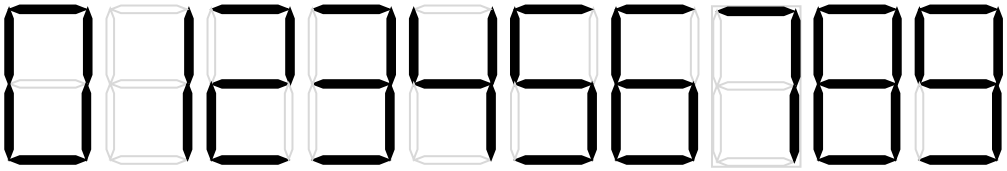
Exercise 8.4.
Draw a logic circuit having 3 input wires and containing all basic and derived gates. Write the corresponding Boolean function, simplify it and then draw a simplified circuit equivalent to the initial one.

Exercise 8.5.
Write a Boolean function of 4 variables given by its table of values, simplify it and draw the logic circuits corresponding to all its simplified forms.

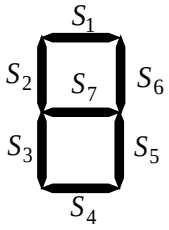
Exercise 8.6.

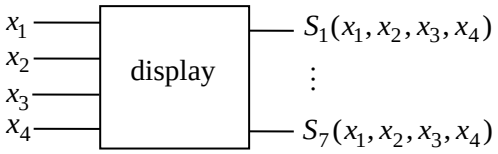
Simplify the Boolean functions (S_1,S_2,S_3,S_4,S_5,S_6,S_7) corresponding to the outputs(segments) of the electronic display of the decimal digits using 7 segments. Draw the logic circuits for the disjunctive/conjunctive simplified forms using only NAND/NOR gates.

Electronic display of the decimal digits using 7 segments (LEDs):



The combinational circuit having the functionality described above has as inputs 4 variables and as outputs 7 functions corresponding to the segments.





The combinations of 4 binary digits which do not correspond to a decimal digit will generate an emission of light only for the segment S_7.

	inputs				outputs						
Decimal digit	x_1	x_2	x_3	x_4	S_1	S_2	S_3	S_4	S_5	S_6	S_7
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	0	0	0	1	1	0
2	0	0	1	0	1	0	1	1	0	1	1
3	0	0	1	1	1	0	0	1	1	1	1
4	0	1	0	0	0	1	0	0	1	1	1
5	0	1	0	1	1	1	0	1	1	0	1
6	0	1	1	0	1	1	1	1	1	0	1
7	0	1	1	1	1	0	0	0	1	1	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	0	1	1	1	1
-	1	0	1	0	0	0	0	0	0	0	1
-	1	0	1	1	0	0	0	0	0	0	1

–	1	1	0	0	0	0	0	0	0	0	1
–	1	1	0	1	0	0	0	0	0	0	1
–	1	1	1	0	0	0	0	0	0	0	1
–	1	1	1	1	0	0	0	0	0	0	1

Exercise 8.7.

Binary codes are used to represent the ten decimal digits on 4 bits. There exist *weighted codes* and *unweighted codes*. In a weighted code each binary digit has a 'weight' and the decimal value is obtained as the weighted sum of all 4 bits.

BCD - Binary Coded Decimal is an example of a weighted code, with the weights for the digits: 8,4,2,1 (from the most significant digit to the least significant one).

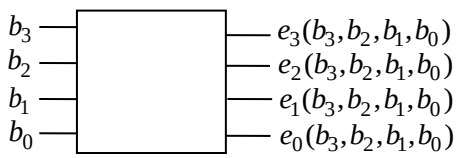
The most used unweighted codes are *Excess 3* (obtained by adding '0011' to each *BCD* word) and the *Gray code* (a cyclic code with the property that all the successive words differ by one single digit).

The table below provides the *BCD*, *Excess 3* and *Gray* code on 4 bits.

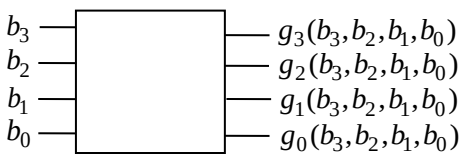
Decimal value	*BCD*				*Excess 3*				*Gray* code			
	b_3 8	b_2 4	b_1 2	b_0 1	e_3	e_2	e_1	e_0	g_3	g_2	g_1	g_0
0	0	0	0	0	0	0	1	1	0	0	0	0
1	0	0	0	1	0	1	0	0	0	0	0	1
2	0	0	1	0	0	1	0	1	0	0	1	1
3	0	0	1	1	0	1	1	0	0	0	1	0
4	0	1	0	0	0	1	1	1	0	1	1	0
5	0	1	0	1	1	0	0	0	0	1	1	1
6	0	1	1	0	1	0	0	1	0	1	0	1
7	0	1	1	1	1	0	1	0	0	1	0	0
8	1	0	0	0	1	0	1	1	1	1	0	0
9	1	0	0	1	1	1	0	0	1	1	0	1
10	1	0	1	0	-	-	-	-	1	1	1	1
11	1	0	1	1	-	-	-	-	1	1	1	0
12	1	1	0	0	-	-	-	-	1	0	1	0
13	1	1	0	1	-	-	-	-	1	0	1	1
14	1	1	1	0	-	-	-	-	1	0	0	1
15	1	1	1	1	-	-	-	-	1	0	0	0

Implement the combinational circuits for converting:

1. from *BCD* to *Excess 3* code;

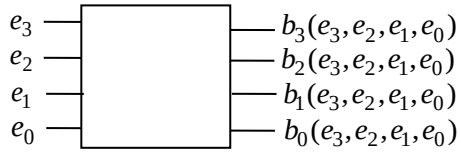


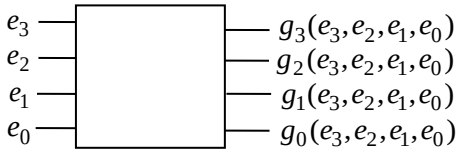
2. from *BCD* to *Gray* code;



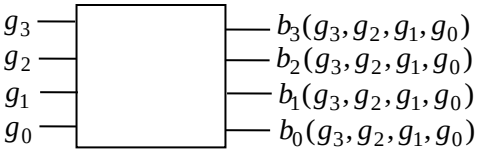
3. from *Excess 3* code to *BCD*;

4. from *Excess 3* code to *Gray* code;





5. from *Gray* code to *BCD*;



6. from *Gray* code to *Excess 3* code;