Digital Design Laboratory

Dr. Oniga István
University of Debrecen, Faculty of Informatics

This work was supported by the construction EFOP-3.4.3-16-2016-00021.
The project was supported by the European Union, co-financed by the European Social Fund.

3. Laboratory assignments

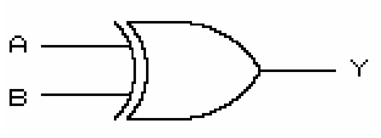
- Two level logic
- SOP implementation
- Logic function simplification

Lab3_1 assignment:

Two level logic implementation
- XOR function implementation -

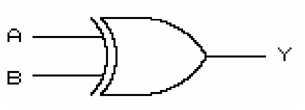
- Create a new project
- Add a new "schematic” source
- Draw the schematics presented on next slide.
- Add and adapt the constraints file Nexysx.UCF
  - Inputs: ***sw[2:0]*** *(A and B on figure)*
  - Outputs: ***led[4:0]***

A	B	Y=A⊕B
L	L	L
L	H	H
H	L	H
H	H	L

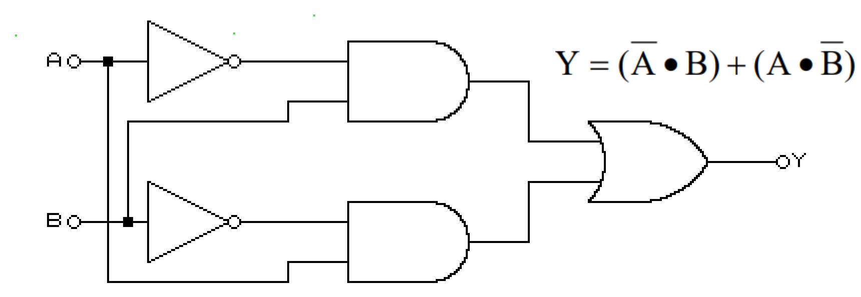


XOR function implementation

A	B	Y=A⊕B
L	L	L
L	H	H
H	L	H
H	H	L

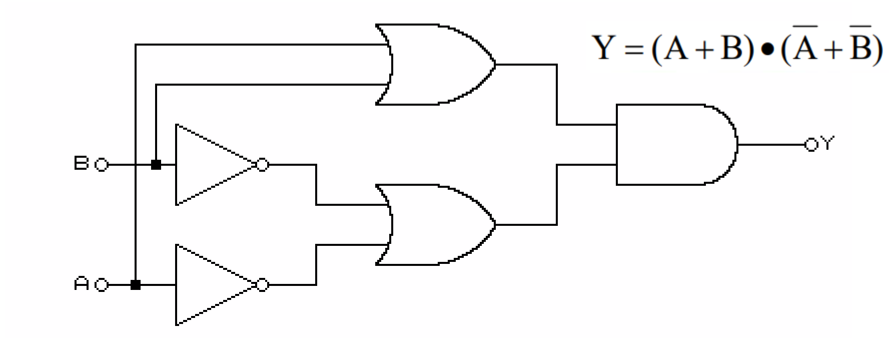


SOP Implementation



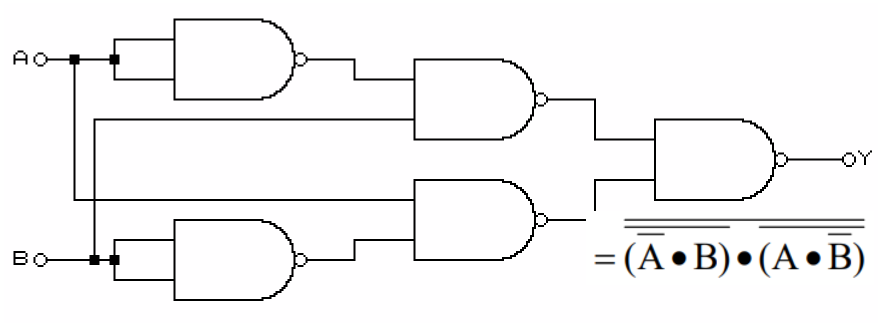
$Y = (\overline{A} \bullet B) + (A \bullet \overline{B})$

POS Implementation



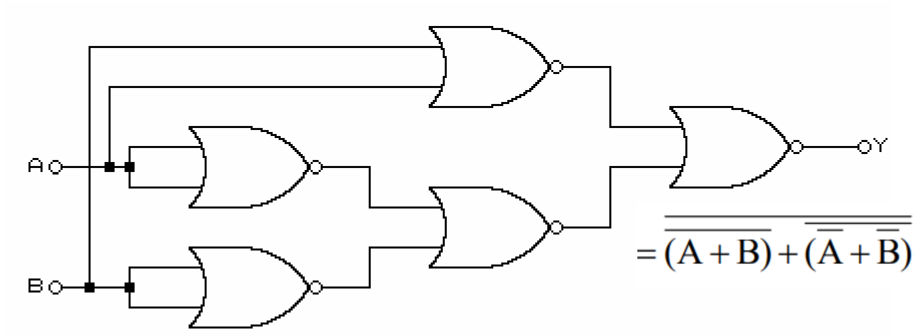
$Y = (A + B) \bullet (\overline{A} + \overline{B})$

NAND Implementation



$= \overline{\overline{(\overline{A} \bullet B)} \bullet \overline{(A \bullet \overline{B})}}$

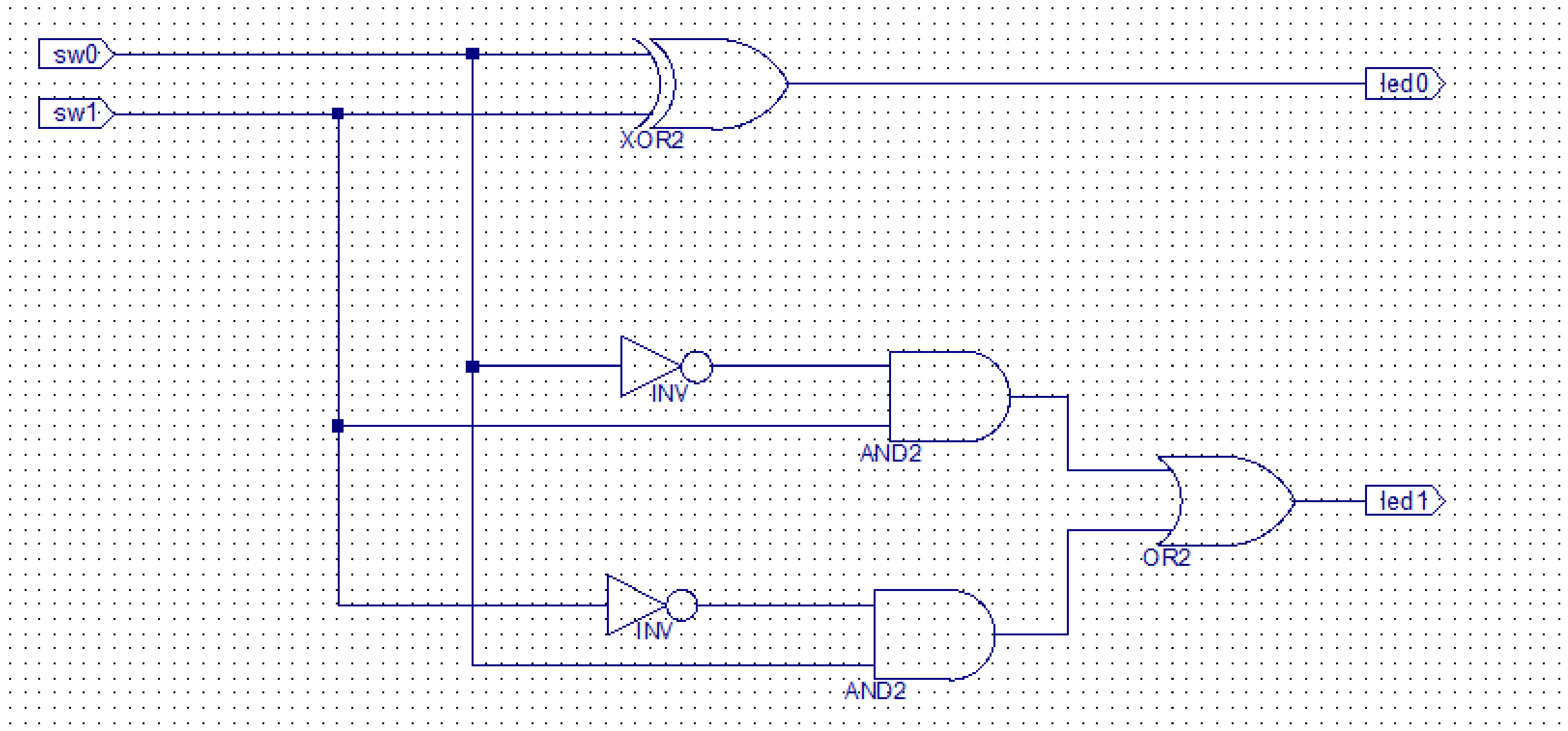
NOR Implementation



$= \overline{\overline{(A + B)} + \overline{(\overline{A} + \overline{B})}}$

XOR function implementation

- Complete the schematic with next 3 implementation.



- You can download the this part of the schematic from Digital Design Laboratory page.

Lab3_1 Results

- Generate the configuration file, download to board, test the project
- Using switches sw0 and sw1 make all 4 possible combinations and note the corresponding state of the LEDs on the following table
- Note your experience in the Laboratory's Report Questionnaire
  - All outputs are identical?
  - Which implementation od XOR function is most advantageous and why?

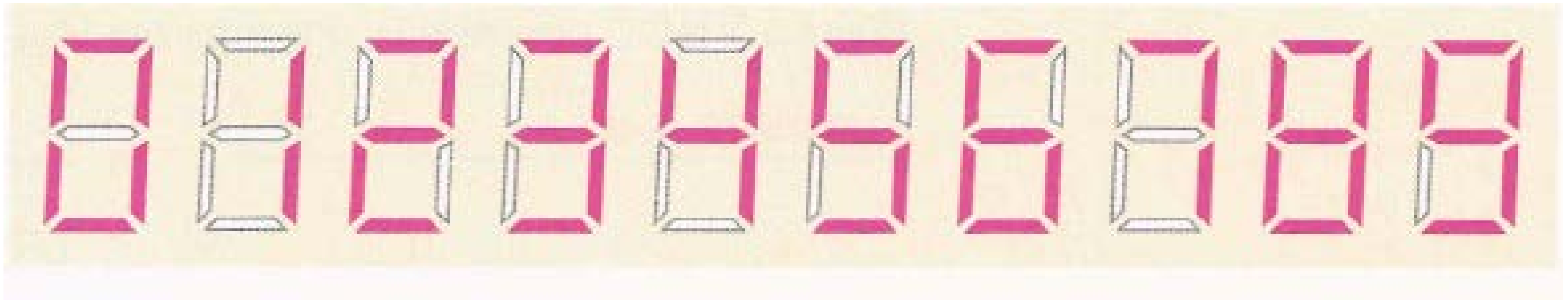
sw0	sw1	led0 AxorB	led1 (SOP)	led2 (POS)	led3 ("NAND”)	led4 ("NOR”)
0	0					
0	1					
1	0					
1	1					

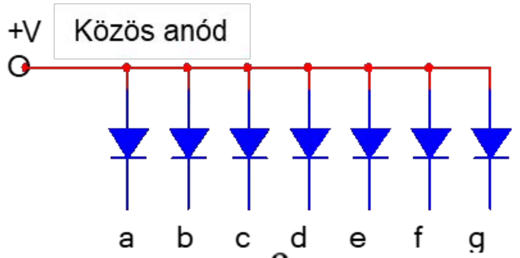
Lab3_2 assignment:

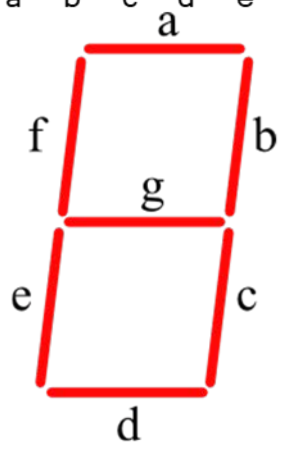
BCD – 7 segments decoder „a” segment

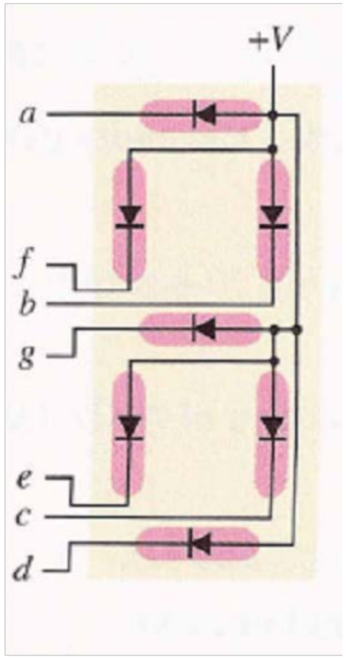
- Create a new project
- Add a new "schematic” source
- Design the “a” segment for a BCD – 7 segment decoder
- Draw all 3 circuits on the same schematic page.
  - Inputs BCD cod: ***sw3, sw2, sw1, sw0***
  - Outputs:
    - ***„ca”*** *segment cathode =* ***negated output*** *of the first circuit*
      (because the CA..CG signals are driven low when active)
    - ***„an0”*** *common anode of the first 7 segment digit – must be connected to GND*
      (the AN0..7 signals are driven low when active)
    - ***led1*** *and* ***led0*** *outputs of the second and third circuits*
- Add and adapt the Nexysx.UCF file

BCD – 7 segments decoder „a” segment









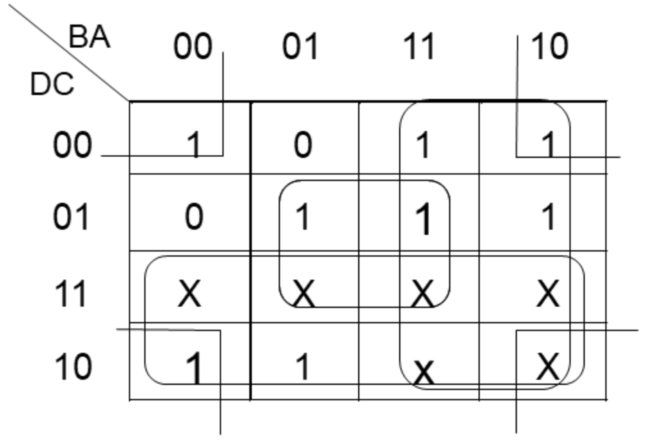
Decimális	D	C	B	A	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
10	1	0	1	0	X	X	X	X	X	X	X
11	1	0	1	1	X	X	X	X	X	X	X
12	1	1	0	0	X	X	X	X	X	X	X
13	1	1	0	1	X	X	X	X	X	X	X
14	1	1	1	0	X	X	X	X	X	X	X
15	1	1	1	1	X	X	X	X	X	X	X

$$a = \overline{D}\,\overline{C}\,\overline{B}\,\overline{A} + \overline{D}\,\overline{C}B\overline{A} + \overline{D}\,\overline{C}BA + \overline{D}C\overline{B}A + \overline{D}CB\overline{A} + \overline{D}CBA + D\overline{C}\,\overline{B}\,\overline{A} + D\overline{C}\,\overline{B}A$$

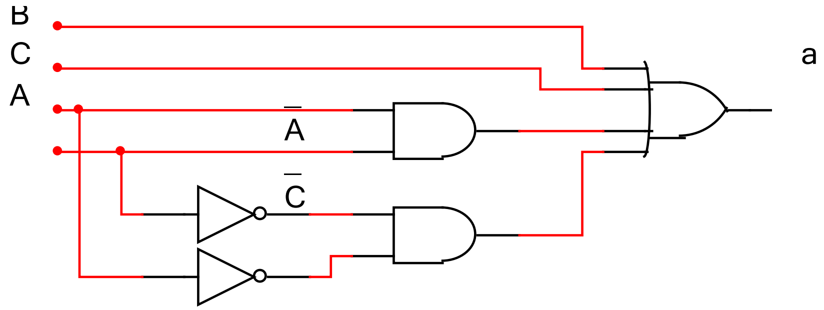
BCD – 7 segments decoder „a” segment

$$a=\overline{D}\,\overline{C}\,\overline{B}\,\overline{A}+\overline{D}\,\overline{C}\,B\,\overline{A}+\overline{D}\,\overline{C}\,BA+\overline{D}C\,\overline{B}A+\overline{D}CB\overline{A}+\overline{D}CBA+D\overline{C}\,\overline{B}\,\overline{A}+D\overline{C}\,\overline{B}A$$

$$a=D+B+CA+\overline{C}\,\overline{A}$$



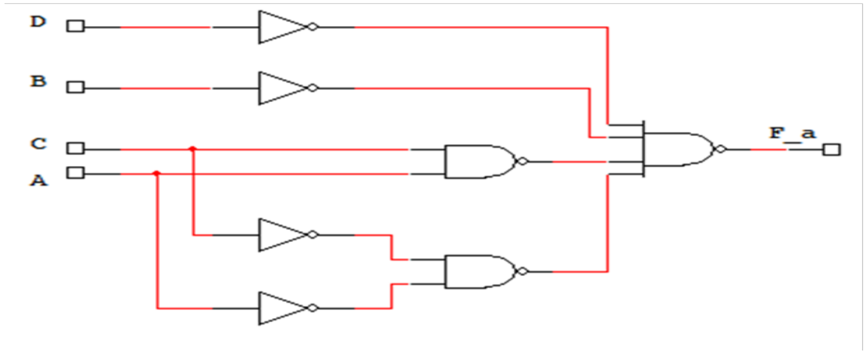
DC \ BA	00	01	11	10
00	1	0	1	1
01	0	1	1	1
11	X	X	X	X
10	1	1	x	X

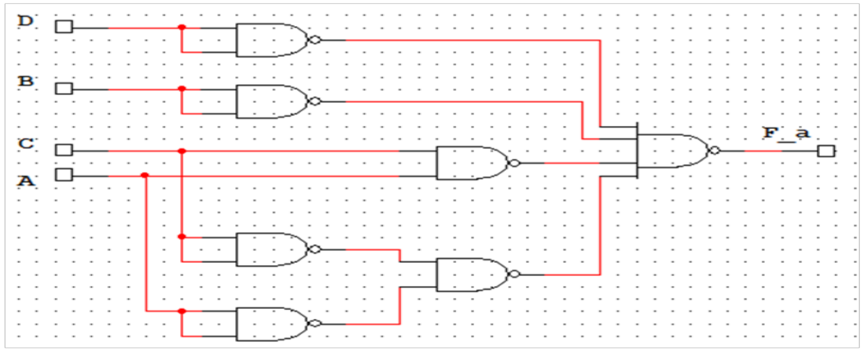


Implementation using NAND gates

Previous function could be transformed using De Morgan rules:

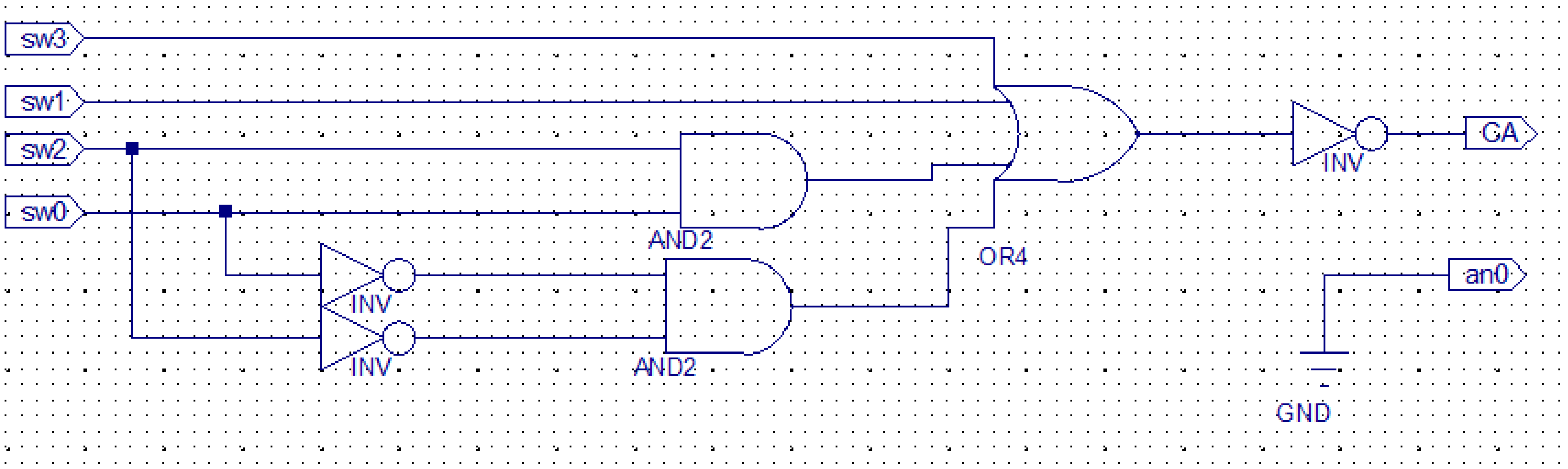
$$a=D+B+CA+\overline{C}\,\overline{A}=\overline{\overline{D+B+CA+\overline{C}\,\overline{A}}}=\overline{\overline{D}\bullet\overline{B}\bullet\overline{C\bullet A}\bullet\overline{\overline{C}\bullet\overline{A}}}$$





BCD – 7 segments decoder „a” segment

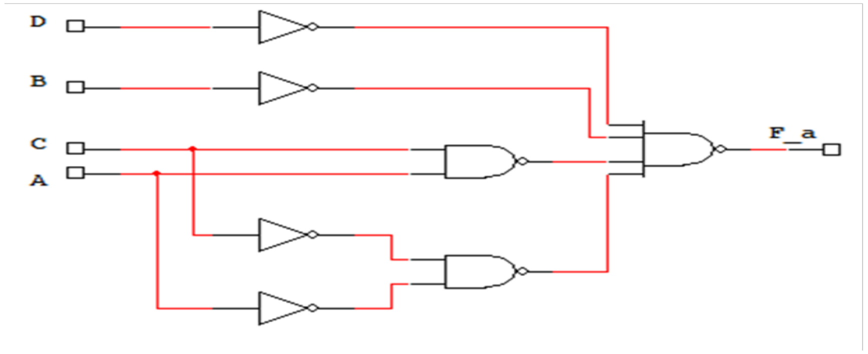
$$a = D + B + CA + \overline{C}\,\overline{A}$$

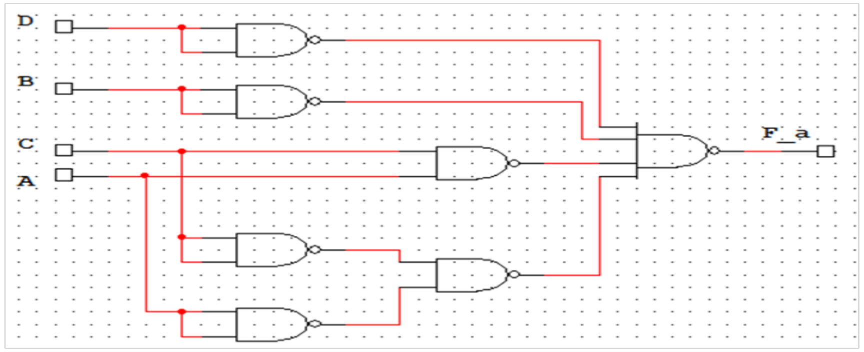


Implementation using NAND gates

Previous function could be transformed using De Morgan rules:

$$a = D + B + CA + \overline{C}\,\overline{A} = \overline{\overline{D + B + CA + \overline{C}\,\overline{A}}} = \overline{\overline{D} \bullet \overline{B} \bullet \overline{C \bullet A} \bullet \overline{\overline{C} \bullet \overline{A}}}$$





Lab3_2 Results

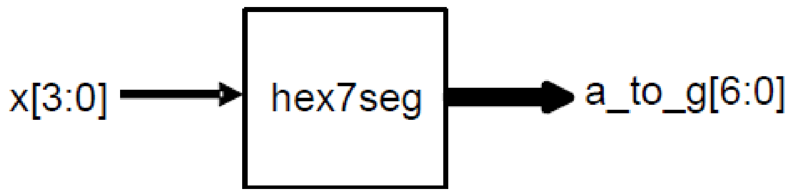
- Generate the configuration file, download to board, test the project
- Using switches sw0, sw1, sw2 and sw3 make all possible combinations and note the corresponding states of the segment "a" and led0 on the following table

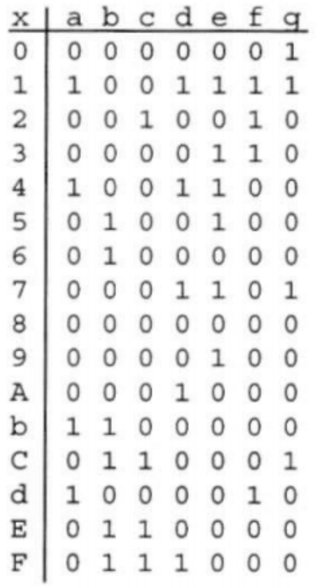
sw3	sw2	sw1	sw0	„a"	led0	Led1	„a"=led0=led1?
0	0	0	0				
0	0	0	1				
0	0	1	0				
0	0	1	1				
0	1	0	0				
0	1	0	1				
0	1	1	0				
0	1	1	1				
1	0	0	0				
1	0	0	1				
1	0	1	0				
1	0	1	1				
1	1	0	0				
1	1	0	1				
1	1	1	0				
1	1	1	1				

- Note your experience in the Laboratory's Report Questionnaire

Lab3_3assignment:

7 segment display

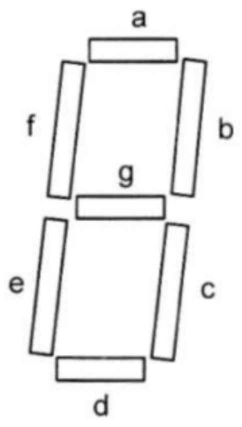




x	a	b	c	d	e	f	g
0	0	0	0	0	0	0	1
1	1	0	0	1	1	1	1
2	0	0	1	0	0	1	0
3	0	0	0	0	1	1	0
4	1	0	0	1	1	0	0
5	0	1	0	0	1	0	0
6	0	1	0	0	0	0	0
7	0	0	0	1	1	0	1
8	0	0	0	0	0	0	0
9	0	0	0	0	1	0	0
A	0	0	0	1	0	0	0
b	1	1	0	0	0	0	0
C	0	1	1	0	0	0	1
d	1	0	0	0	0	1	0
E	0	1	1	0	0	0	0
F	0	1	1	1	0	0	0

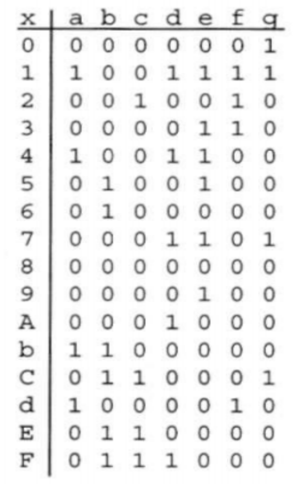
1 = off

0 = on



- The segments and dots are driven individually
- The segments are active low

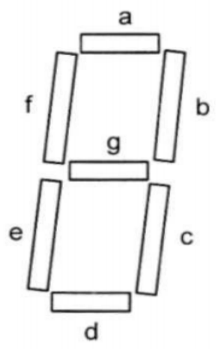
Driving the 7 segment display

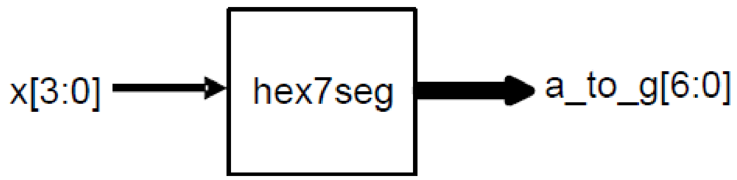


x	a	b	c	d	e	f	g
0	0	0	0	0	0	0	1
1	1	0	0	1	1	1	1
2	0	0	1	0	0	1	0
3	0	0	0	0	1	1	0
4	1	0	0	1	1	0	0
5	0	1	0	0	1	0	0
6	0	1	0	0	0	0	0
7	0	0	0	1	1	0	1
8	0	0	0	0	0	0	0
9	0	0	0	0	1	0	0
A	0	0	0	1	0	0	0
b	1	1	0	0	0	0	0
C	0	1	1	0	0	0	1
d	1	0	0	0	0	1	0
E	0	1	1	0	0	0	0
F	0	1	1	1	0	0	0

1 = off

0 = on





```
module hex7seg ( input [3:0] x,
                 output reg [6:0] a_to_g );
always @(*)
case(x)
0: a_to_g = 7'b0000001;
1: a_to_g = 7'b1001111;
2: a_to_g = 7'b0010010;
3: a_to_g = 7'b0000110;
4: a_to_g = 7'b1001100;
5: a_to_g = 7'b0100100;
6: a_to_g = 7'b0100000;
7: a_to_g = 7'b0001111;
8: a_to_g = 7'b0000000;
9: a_to_g = 7'b0000100;
'hA: a_to_g = 7'b0001000;
'hb: a_to_g = 7'b1100000;
'hC: a_to_g = 7'b0110001;
'hd: a_to_g = 7'b1000010;
'hE: a_to_g = 7'b0110000;
'hF: a_to_g = 7'b0111000;
default: a_to_g = 7'b0000001; // 0
endcase
endmodule
```

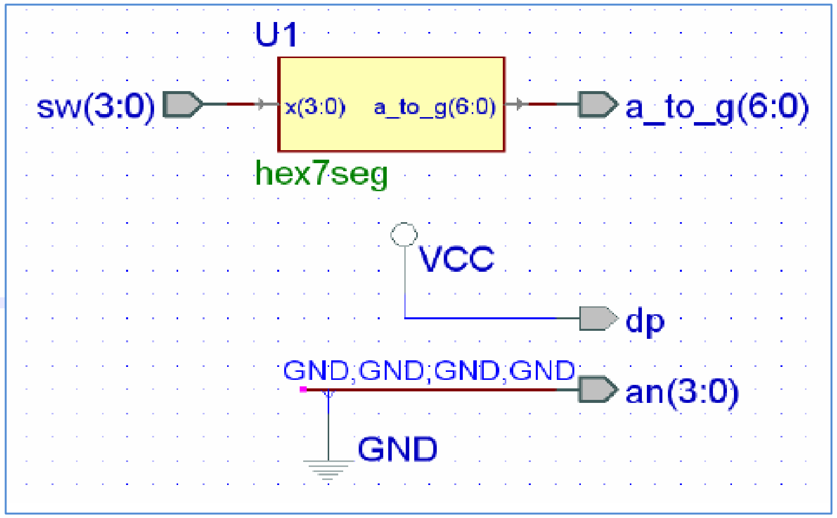
- Create a new project
- Add a new source Verilog file
  - Inputs: x[3:0]
  - Outputs: a_to_g[6:0]
- Add a new Verilog file, which will be the „top module”
  - Inputs: x[3:0]
  - Outputs: a_to_g[6:0]
- The file could be downloaded from Digital design laboratory website.

Driving the 7 segment display

```
module hex7seg_top (input [3:0] sw, output [6:0] a_to_g ,
output [3:0] an, output dp);

assign an = 4'b0000; // all digits on
assign dp = 1; // dp off

hex7seg D4 (.x(sw),.a_to_g(a_to_g));
endmodule
```



- Add and adapt the Nexysx.UCF file
  - Inputs are: ***sw[3:0]***
  - Outputs are: ***a_to_g [6:0], an(3:0), dp.***

- Generate the configuration file and generálás, download to board, test the project
- Using switches sw0, sw1, sw2 and sw3 make all possible combinations and check the correctness of the corresponding states of the 7 segment display
- Change the code to so that only one display will display the hexadecimal number.
- Note your experience in the Laboratory's Report Questionnaire
  - How did you managed to display the numbers only on a single digit?
  - Which signal level did you used to torn on a digit on Nexys 4 board?
  - What are the active states for the signals used to drive the segments
  - What is the "dp" signal driving?