



**Universitatea Tehnică “Gheorghe Asachi” din Iași**



**FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE**

# **ELECTRONICĂ DIGITALĂ**

## **proiect**

**Tema: Comparator**

Studenți: Buțu Alexandra-Gabriela

Chelea Diana-Maria

Spiridon Bianca

Grupa : 1207B

Coordonator:

Asist. Drd. Marius Obreja

**2023**

## 1. Specificațiile proiectului:

### COMPARATOR

Să se implementeze în FPGA prin descriere în limbaj VHDL, două comparatoare de câte 2 vectori de 4 biți care să furnizeze la ieșire rezultatul mai mic, egal sau mai mare; unul dintre vectorii de intrare va fi comun la cele două comparatoare.

Implementarea proiectului va fi făcută printr-o descriere comportamentală

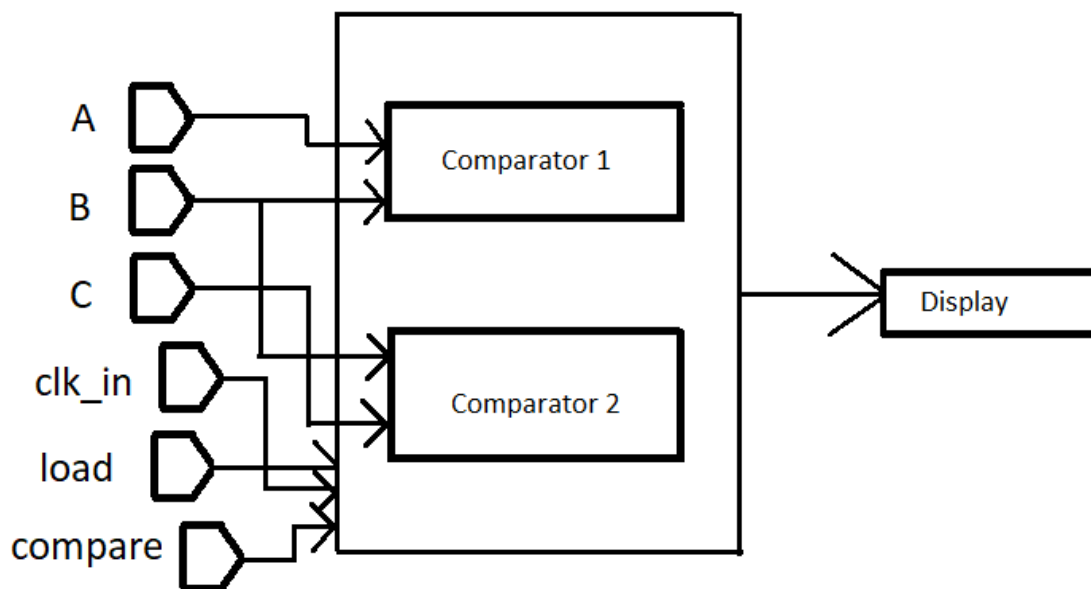


Fig. 1 schema bloc a modului COMPARATOR

Fișierul bitstream rezultat în urma procesului de implementare va fi verificat utilizând placa de dezvoltare BASYS3.

## 2. Modulul COMPARATOR

Modulul COMPARATOR are urmatoarele functionalitati:

1. Modulul are ca sursa de clock extern butonul din centru (U18)
2. Functioneaza pe frontul pozitiv al clock-ului
3. Contine 3 numere pe 4 biti , introduse cu ajutorul a 12 switch-uri(Primul numar este introdus de la stanga la dreapta , incepand cu switch-ul 1 pana la 4, al doilea este introdus de la switch-ul 6 la switchul 9, al treilea este introdus de la switch-ul 11 pana la 14)
4. Odata introduse numerele, pentru aprinderea led-urilor se va activa switch-ul 15.
5. Pentru compararea numerelor , se va pune switch-ul 15 pe 0, si se va pune switch-ul 16 pe 1
6. Rezultatul compararii numerelor va fi afisat utilizand unul din displayurile modulului astfel :

❖ In partea de sus a display-ului va aparea rezultatul compararii primelor doua numere sub 3 forme:

- Primul numar mai mare decat al doilea (se aprinde portiunea stanga sus )



- Numerele sunt egale (se aprinde portiunea de sus din mijloc )



- Primul numar mai mic decat al doilea (se aprinde portiunea dreapta sus)



❖ In partea de jos a display-ului va aparea rezultatul compararii ultimelor doua numere sub 3 forme:

- Al doilea numar mai mare decat al treilea (se aprinde portiunea stanga jos)



- Numerele sunt egale (se aprinde portiunea de jos din mijloc)



- Al doilea numar mai mic decat al treilea (se aprinde portiunea dreapta jos )



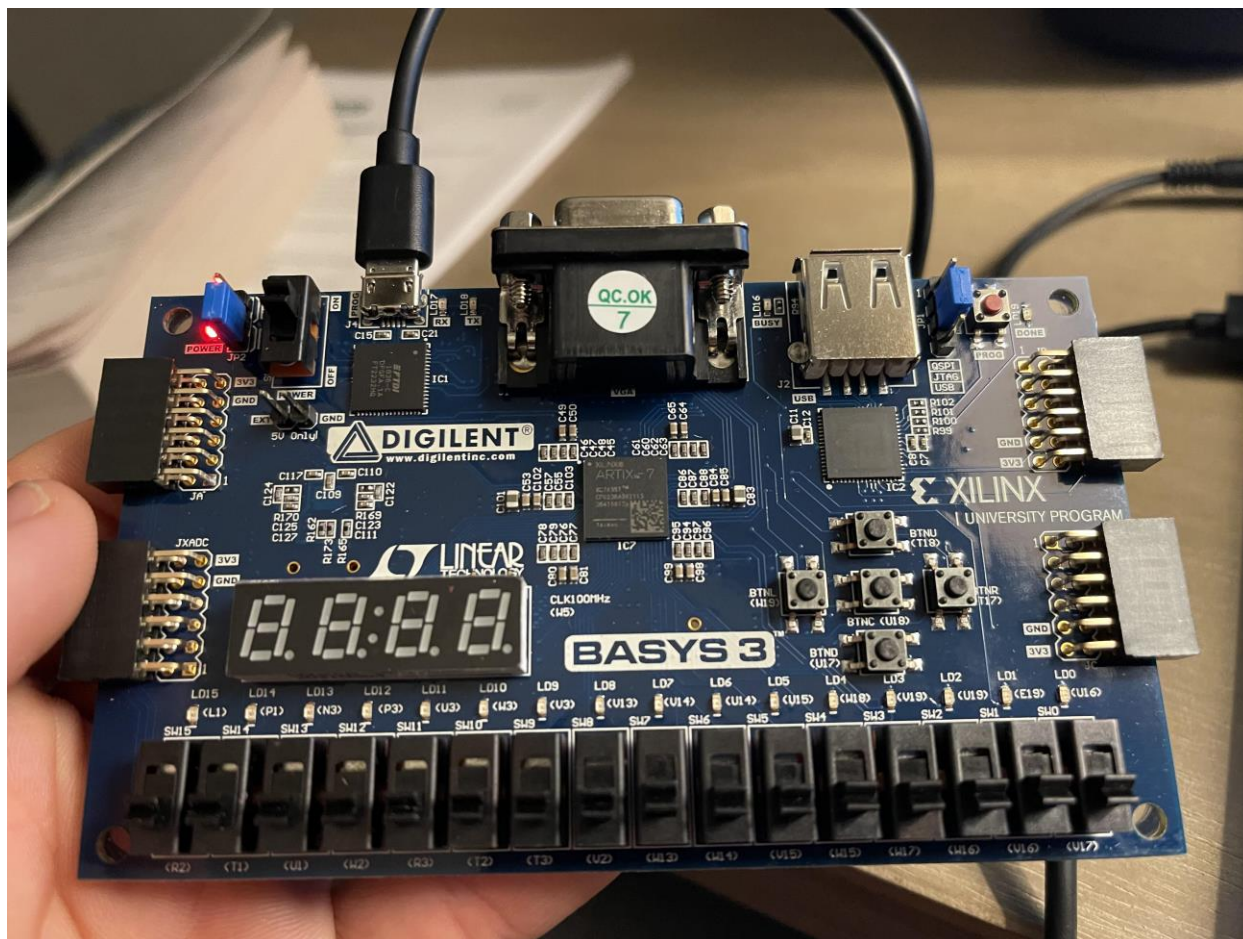
### 3. Metoda de implementare

Pentru implementarea acestui modul s-au folosit programul de sinteza Vivado si limbajul VHDL. Implementarea proiectului a fost facuta printr-o descriere comportamentala. S-a proiectat entitatea comparator . Aceasta are un “proces” care compara, prin operatii pe biti, numerele incarcate anterior, si furnizeaza rezultatul compararii printr-un segment din display.

Fisierul bitstream creat de programul Vivado a fost testat cu ajutorul placii BASYS 3 Artix-7 xc7a35tcbg236-1.

### 4. Descrierea (scurtă) a sistemului de dezvoltare BASYS 3

Placa de dezvoltare BASYS 3 este un circuit de dezvoltare complet si ready-to-use bazat pe ultimele Artix-7 Field Programmable Gate Array(FPGA) produse de Xilinx. Cu o mare capacitate de FPGA si cu o colectie de porturi USB, VGA si altele, placa de dezvoltare BASYS 3 permite proiectarea unor design-uri variate, atat circuite introductorii combinationale, cat si circuite secventiale complexe ca procesoarele si controllerele embedded.



## 5. Editarea fișierului VHDL

-----proiect\_2.vhd(TOP MODULE)-----

Entitatea comparator:


```

1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity comparator is
5  Port (
6  clk_in: std_logic;
7  X: in std_logic_vector(3 downto 0);
8  Y: in std_logic_vector(3 downto 0);
9  Z: in std_logic_vector(3 downto 0);
10
11  X_out: out std_logic_vector(3 downto 0);
12  Y_out: out std_logic_vector(3 downto 0);
13  Z_out: out std_logic_vector(3 downto 0);
14
15  load: in std_logic;
16  compare: in std_logic;
17
18  an: out STD_LOGIC_VECTOR(3 downto 0);
19  seg: out STD_LOGIC_VECTOR(6 downto 0)
20
21  );
22  end comparator;
23

```

Buffere pentru semnalele de intrare si iesire si vectorii pentru calcule:

```

24  architecture Behavioral of comparator is
25
26   --Buffer semnal intrare
27
28   signal X1: std_logic_vector(3 downto 0);
29   signal Y1: std_logic_vector(3 downto 0);
30   signal Z1: std_logic_vector(3 downto 0);
31   --Buffer semnal iesire
32
33   signal X1_out: std_logic_vector(3 downto 0);
34   signal Y1_out: std_logic_vector(3 downto 0);
35   signal Z1_out: std_logic_vector(3 downto 0);
36
37   --Vectorii pentru calculul diferentei
38   signal U1 :std_logic_vector(3 downto 0);
39   signal U2 :std_logic_vector(3 downto 0);
40
41   --Vectorii pentru transporturile rezultate in urma scaderii
42   signal C1 :std_logic_vector(4 downto 0);
43   signal C2 :std_logic_vector(4 downto 0);
44
45   begin
46
47     X1<=X;           --initializarea bufferelor
48     X_out<=X1_out;
49     Y1<=Y;
50     Y_out<=Y1_out;
51     Z1<=Z;
52     Z_out<=Z1_out;

```

Ledul folosit:

```

53
54   an(0) <= '0';
55   an(1) <= '1';
56   an(2) <= '1';
57   an(3) <= '1';
58

```

Rezultatele compararii :

```
58 |
59 | --Rezultatele compararii
60 |
61 | seg(5) <= not(not(C1(4)) and (U1(0) or U1(1) or U1(2) or U1(3)) ); --mai mare
62 | seg(0) <= not(not(C1(4)) and not(U1(0) or U1(1) or U1(2) or U1(3)) ); --egal
63 | seg(1) <= not(C1(4)); --mai mic
64 |
65 | seg(4) <= not(not(C2(4)) and (U2(0) or U2(1) or U2(2) or U2(3)) ); --mai mare
66 | seg(3) <= not(not(C2(4)) and not(U2(0) or U2(1) or U2(2) or U2(3)) ); --egal
67 | seg(2) <= not(C2(4)); --mai mic
68 |
69 | seg(6) <= '1';
```

Cat timp clk si load sunt pe 1 , se incarca inputurile numerelor in leduri

```
70 | process(clk_in, load, compare) begin
71 |     if rising_edge(clk_in) then
72 |         --In cazul in care load este pe '1' , incarcam inputul din switchuri in leduri
73 |         if(load = '1') then
74 |             X_out <= X;
75 |             Y_out <= Y;
76 |             Z_out <= Z;
77 |
78 |         else
79 |
```



Cand load-ul este pe 0 , si clk si compare sunt pe 1 se incepe compararea numerelor:

```
else

--In caz contrar, ne folosim de valoarea output-ului din led-uri pentru a compara numerele
if(compare = '1') then
--Comparator 1:

C1(0) <= '0';
C2(0) <= '0';

for I in 0 to 3 loop
U1(I) <= ( not ( C1(I) ) and not ( X1_out(I) ) and Y1_out(I) ) or (not ( C1(I) )and X1_out(I)
and not ( Y1_out(I) )) or (not( Y1_out(I) ) and not ( X1_out(I) ) and C1(I)) or (C1(I) and X1_out(I) and Y1_out(I));
C1(I +1)<= (not (X1_out(I)) and Y1_out(I)) or (C1(I) and ((not ( X1_out(I) )) or Y1_out(I)));
```

## Comparator 2:

```
--Comparator 2:

for I in 0 to 3 loop
U2(I) <= ( not ( C2(I) ) and not ( Y1_out(I) ) and Z1_out(I) ) or (not ( C2(I) )and Y1_out(I) and not ( Z1_out(I) ))
or (not( Z1_out(I) ) and not ( Y1_out(I) ) and C2(I)) or (C2(I) and Y1_out(I) and Z1_out(I));
C2(I +1)<= (not (Y1_out(I)) and Z1_out(I)) or (C2(I) and ((not ( Y1_out(I) )) or Z1_out(I)));
```

## 6. Editarea fișierului de constrângeri

Switch-uri:

```
14 set_property PACKAGE_PIN R2 [get_ports {X[3]}]
15     set_property IOSTANDARD LVCMOS33 [get_ports {X[3]}]
16 set_property PACKAGE_PIN T1 [get_ports {X[2]}]
17     set_property IOSTANDARD LVCMOS33 [get_ports {X[2]}]
18 set_property PACKAGE_PIN U1 [get_ports {X[1]}]
19     set_property IOSTANDARD LVCMOS33 [get_ports {X[1]}]
20 set_property PACKAGE_PIN W2 [get_ports {X[0]}]
21     set_property IOSTANDARD LVCMOS33 [get_ports {X[0]}]
22
23 set_property PACKAGE_PIN V16 [get_ports {load}]
24     set_property IOSTANDARD LVCMOS33 [get_ports {load}]
25 set_property PACKAGE_PIN V17 [get_ports {compare}]
26     set_property IOSTANDARD LVCMOS33 [get_ports {compare}]
27
28 #set_property PACKAGE_PIN W15 [get_ports {swt[4]}]
29     #set_property IOSTANDARD LVCMOS33 [get_ports {swt[4]}]
30 set_property PACKAGE_PIN T2 [get_ports {Y[3]}]
31     set_property IOSTANDARD LVCMOS33 [get_ports {Y[3]}]
32 set_property PACKAGE_PIN T3 [get_ports {Y[2]}]
33     set_property IOSTANDARD LVCMOS33 [get_ports {Y[2]}]
34 set_property PACKAGE_PIN V2 [get_ports {Y[1]}]
35     set_property IOSTANDARD LVCMOS33 [get_ports {Y[1]}]
36 set_property PACKAGE_PIN W13 [get_ports {Y[0]}]
37     set_property IOSTANDARD LVCMOS33 [get_ports {Y[0]}]
38 #set_property PACKAGE_PIN T3 [get_ports {sw[9]}]
39     #set_property IOSTANDARD LVCMOS33 [get_ports {sw[9]}]
40 set_property PACKAGE_PIN V15 [get_ports {Z[3]}]
41     set_property IOSTANDARD LVCMOS33 [get_ports {Z[3]}]
42 set_property PACKAGE_PIN W15 [get_ports {Z[2]}]
43     set_property IOSTANDARD LVCMOS33 [get_ports {Z[2]}]
44 set_property PACKAGE_PIN W17 [get_ports {Z[1]}]
45     set_property IOSTANDARD LVCMOS33 [get_ports {Z[1]}]
46 set_property PACKAGE_PIN W16 [get_ports {Z[0]}]
47     set_property IOSTANDARD LVCMOS33 [get_ports {Z[0]}]
```

## LED-uri:

```
53
54
55  ## LEDs
56  set_property PACKAGE_PIN L1 [get_ports {X_out[3]}]
57      set_property IOSTANDARD LVCMOS33 [get_ports {X_out[3]}]
58  #set_property PACKAGE_PIN U16 [get_ports {Z_out[0]}]
59      #set_property IOSTANDARD LVCMOS33 [get_ports {Z_out[0]}]
60  set_property PACKAGE_PIN P1 [get_ports {X_out[2]}]
61      set_property IOSTANDARD LVCMOS33 [get_ports {X_out[2]}]
62  set_property PACKAGE_PIN N3 [get_ports {X_out[1]}]
63      set_property IOSTANDARD LVCMOS33 [get_ports {X_out[1]}]
64  set_property PACKAGE_PIN P3 [get_ports {X_out[0]}]
65      set_property IOSTANDARD LVCMOS33 [get_ports {X_out[0]}]
66  #set_property PACKAGE_PIN W18 [get_ports {led[4]}]
67      #set_property IOSTANDARD LVCMOS33 [get_ports {led[4]}]
68  set_property PACKAGE_PIN W3 [get_ports {Y_out[3]}]
69      set_property IOSTANDARD LVCMOS33 [get_ports {Y_out[3]}]
70  set_property PACKAGE_PIN V3 [get_ports {Y_out[2]}]
71      set_property IOSTANDARD LVCMOS33 [get_ports {Y_out[2]}]
72  set_property PACKAGE_PIN V13 [get_ports {Y_out[1]}]
73      set_property IOSTANDARD LVCMOS33 [get_ports {Y_out[1]}]
74  set_property PACKAGE_PIN V14 [get_ports {Y_out[0]}]
75      set_property IOSTANDARD LVCMOS33 [get_ports {Y_out[0]}]
76  #set_property PACKAGE_PIN V3 [get_ports {led[9]}]
77      #set_property IOSTANDARD LVCMOS33 [get_ports {led[9]}]
78  set_property PACKAGE_PIN U15 [get_ports {Z_out[3]}]
79      set_property IOSTANDARD LVCMOS33 [get_ports {Z_out[3]}]
80  set_property PACKAGE_PIN W18 [get_ports {Z_out[2]}]
81      set_property IOSTANDARD LVCMOS33 [get_ports {Z_out[2]}]
82  set_property PACKAGE_PIN V19 [get_ports {Z_out[1]}]
83      set_property IOSTANDARD LVCMOS33 [get_ports {Z_out[1]}]
84  set_property PACKAGE_PIN U19 [get_ports {Z_out[0]}]
85      set_property IOSTANDARD LVCMOS33 [get_ports {Z_out[0]}]
86  #set_property PACKAGE_PIN P1 [get_ports {led[14]}]
```

<

7-segment display:

```
90 |
91 | #7 segment display
92 | set_property PACKAGE_PIN W7 [get_ports {seg[0]}]
93 |     set_property IOSTANDARD LVCMOS33 [get_ports {seg[0]}]
94 | set_property PACKAGE_PIN W6 [get_ports {seg[1]}]
95 |     set_property IOSTANDARD LVCMOS33 [get_ports {seg[1]}]
96 | set_property PACKAGE_PIN U8 [get_ports {seg[2]}]
97 |     set_property IOSTANDARD LVCMOS33 [get_ports {seg[2]}]
98 | set_property PACKAGE_PIN V8 [get_ports {seg[3]}]
99 |     set_property IOSTANDARD LVCMOS33 [get_ports {seg[3]}]
100 | set_property PACKAGE_PIN U5 [get_ports {seg[4]}]
101 |     set_property IOSTANDARD LVCMOS33 [get_ports {seg[4]}]
102 | set_property PACKAGE_PIN V5 [get_ports {seg[5]}]
103 |     set_property IOSTANDARD LVCMOS33 [get_ports {seg[5]}]
104 | set_property PACKAGE_PIN U7 [get_ports {seg[6]}]
105 |     set_property IOSTANDARD LVCMOS33 [get_ports {seg[6]}]
106 |
107 | #set_property PACKAGE_PIN V7 [get_ports dp]
108 |     #set_property IOSTANDARD LVCMOS33 [get_ports dp]
109 |
110 | set_property PACKAGE_PIN U2 [get_ports {an[0]}]
111 |     set_property IOSTANDARD LVCMOS33 [get_ports {an[0]}]
112 | set_property PACKAGE_PIN U4 [get_ports {an[1]}]
113 |     set_property IOSTANDARD LVCMOS33 [get_ports {an[1]}]
114 | set_property PACKAGE_PIN V4 [get_ports {an[2]}]
115 |     set_property IOSTANDARD LVCMOS33 [get_ports {an[2]}]
116 | set_property PACKAGE_PIN W4 [get_ports {an[3]}]
117 |     set_property IOSTANDARD LVCMOS33 [get_ports {an[3]}]
118 |
```

Butoane externe (pentru clock):

```
118 |
119 | ##Buttons
120 | set_property PACKAGE_PIN U18 [get_ports {clk_in}]
121 | set_property IOSTANDARD LVCMOS33 [get_ports {clk_in}]
122 | set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets {clk_in}]
```

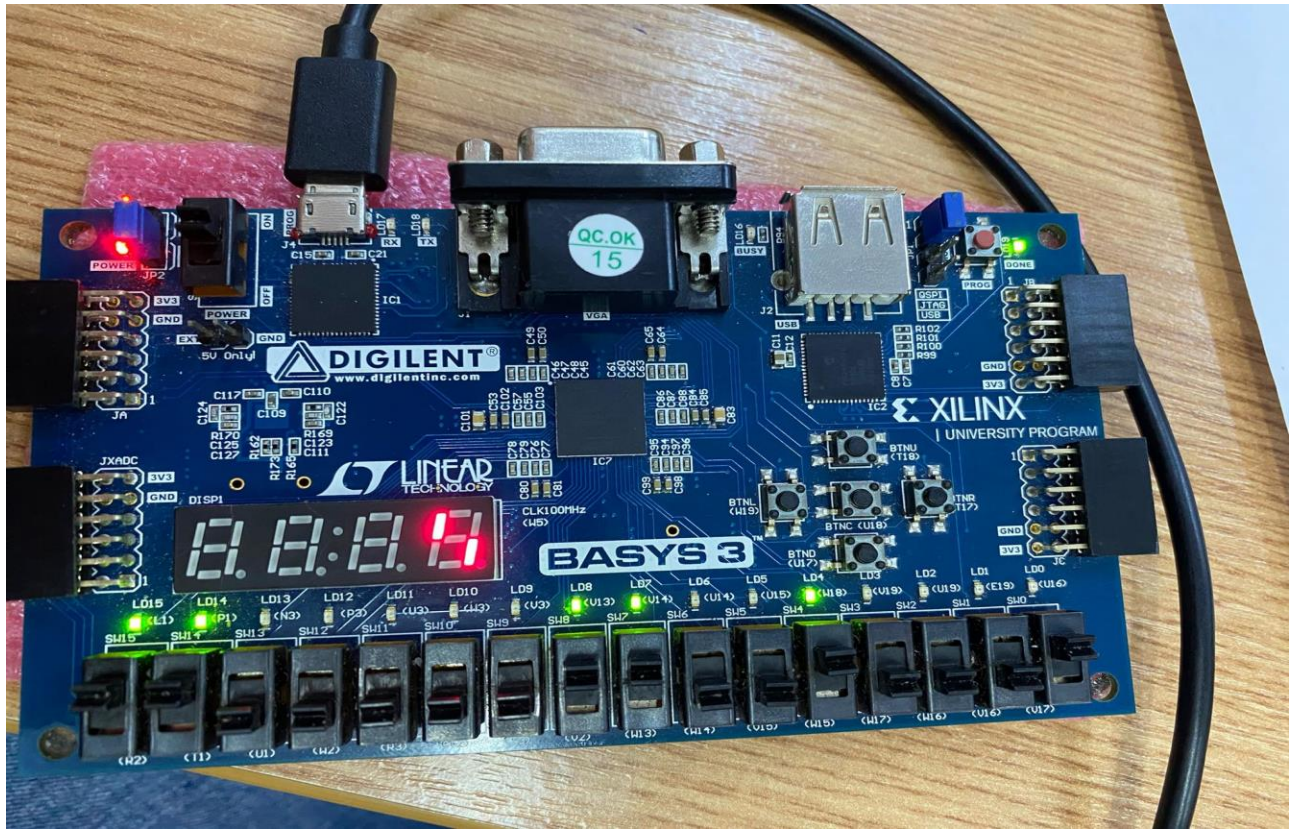
## **7. Descrierea pașilor de sinteză și testarea circuitului rezultat**

1. S-a creat un proiect nou în programul Vivado
2. S-a implementat modulul “MODUL\_COMPARATOR” printr-o descriere comportamentală.
3. S-a editat fișierul de constrângeri în vederea realizării legăturilor între switch-uri și intrări, butonului din mijloc(U18) și clock, switch(V16) și load, switch(V17) și activarea comparatorului, segmentele display-ului și ieșirile modulului .
4. S-a realizat analiza RTL(Register Transfer Level)
5. S-a sintetizat modulul(pentru se vedea design-ul sintetizat)
6. S-a lansat implementarea proiectului care a avut ca efect final generarea fișierului bitstream
7. S-a programat placa de dezvoltare BASYS 3 cu fișierul bitstream și s-a testat funcționarea corespunzătoare a modulului implementat

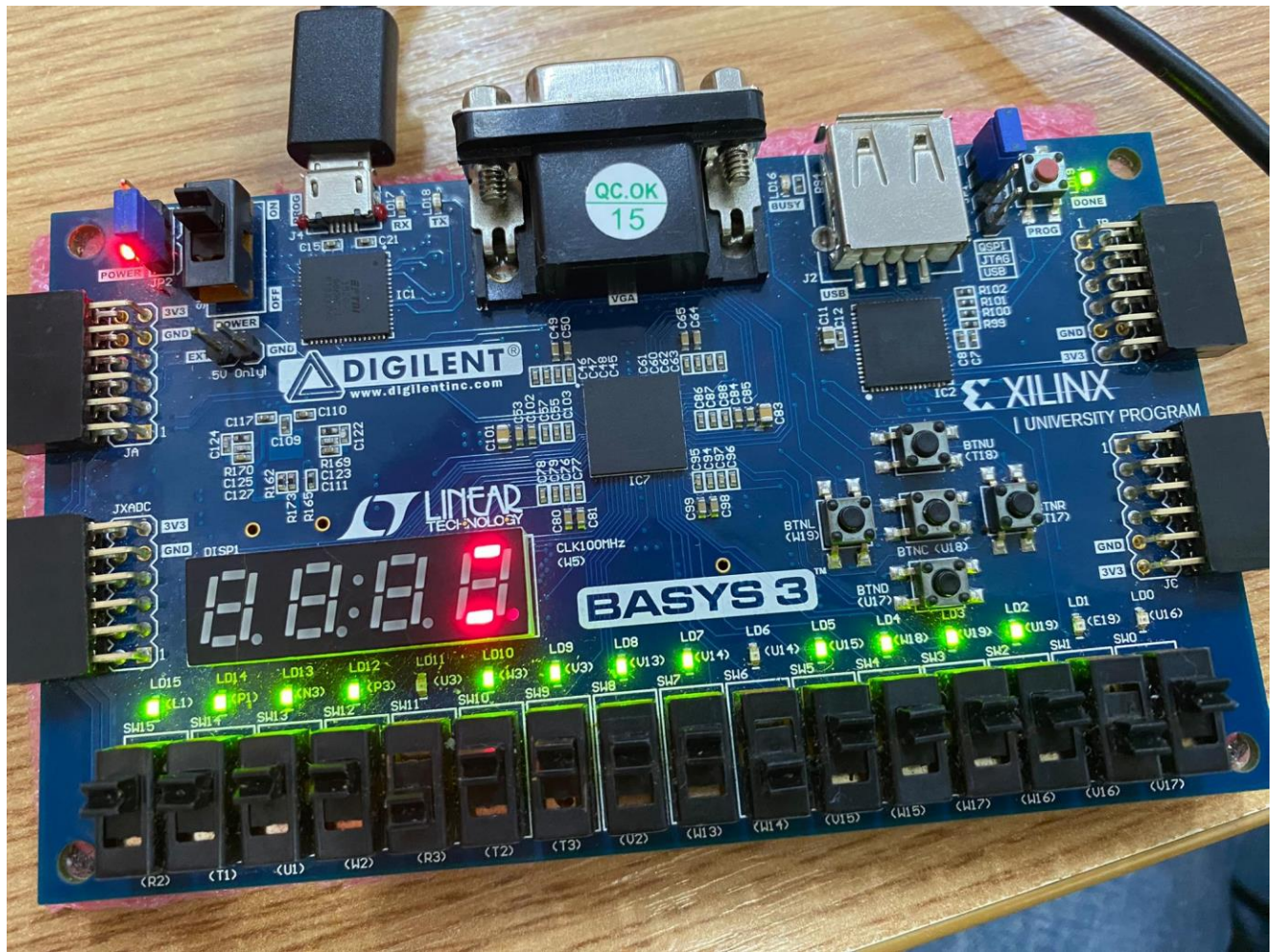


## 8. Fotografii cu functionarea moduluiui:

Cazul cand  $nr1 > nr2$  si  $nr2 < nr3$

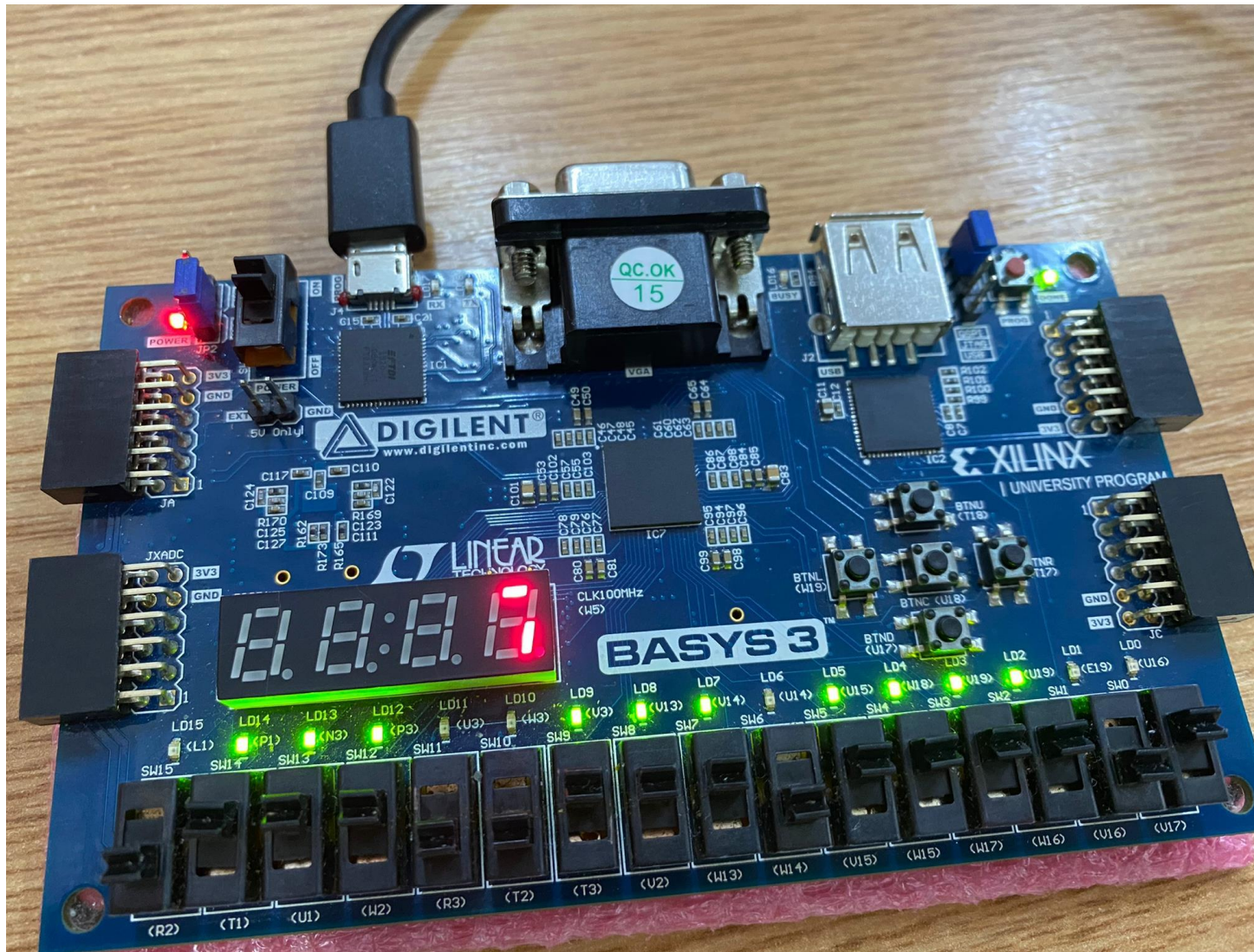


Cazul cand  $nr1 = nr2$  si  $nr2 = nr3$  deci  $nr1 = nr2 = nr3$



Cazul cand  $nr1=nr2$  si  $nr2 < nr3$





## 9. Concluzii

In concluzie, s-a implementat ca proiect un comparator. Acesta compara , prin operatii pe biti, trei numere , doua cate doua, incarcate prin intermediul switch-urilor, iar rezultatul este furnizat la final printr-un segment din display(cel mai din dreapta). Astfel, relatia (= / < / >) dintre primul si al doilea numar poate fi observata in partea superioara, iar cea dintre al doilea si al treilea numar in partea inferioara al segmentului.

### Bibliografie:



1. VHDL Reference Manual,

<http://www.ics.uci.edu/~jmoorkan/vhdlref/Synario%20VHDL%20Manual.pdf>

2. BASYS 3 Reference Manual, <https://reference.digilentinc.com/reference/programmable-logic/basys-3/reference-manual>