

# Sistem de iluminare variabilă

Proiect realizat de Halasz Lorand Daniel

Facultatea de Automatică și Calculatoare

Anul I

Seria B, Grupa 30216

Profesor Coordonator  
Conf. Dr. Ing. Lucia Văcariu

# Cuprins

1. Introducere	3
1.1 Specificația proiectului	3
1.2 Considerații teoretice	4
2. Schema bloc a proiectului	6
3. Etapele de proiectare	8
3.1 Lista componentelor utilizare	8
3.2 Cod Componente (Explicații)	9
4. Pașii de proiectare	22
5. Manual de utilizare	27
6. Justificarea soluției alese	33
7. Posibilități de dezvoltare ulterioară	33

# 1. Introducere

## 1.1 Specificația proiectului

Să se proiecteze un **sistem de iluminare variabilă** cu ledurile de pe plăcile cu FPGA. Sistemul va avea mai multe moduri de funcționare:

- Mod manual – valoarea intensității luminoase a ledurilor se furnizează de pe încrerupătoare (8 biți);
- Mod test – intensitatea luminoasă a ledurilor variază de la valoarea minimă la valoarea maximă într-un interval de timp specific fiecărui led (led0 – 1 secundă, led1 – 2 secunde, ... led7 – 8 secunde), forma de undă aproximată ferăstrău;
- Mod automat – intensitatea luminoasă a ledurilor variază de la valoarea minimă la valoarea maxima și înapoi la minim într-un interval de timp măsurat in secunde. Acest interval de timp este furnizat ca o intrare a sistemului, forma de undă aproximată triunghi.

Pentru varierea intensității luminoase se va folosi tehnica PWM (pulse width modulation).

Proiectul va fi realizat de 1 student. Proiectul este realizat pe placa Basys 3.

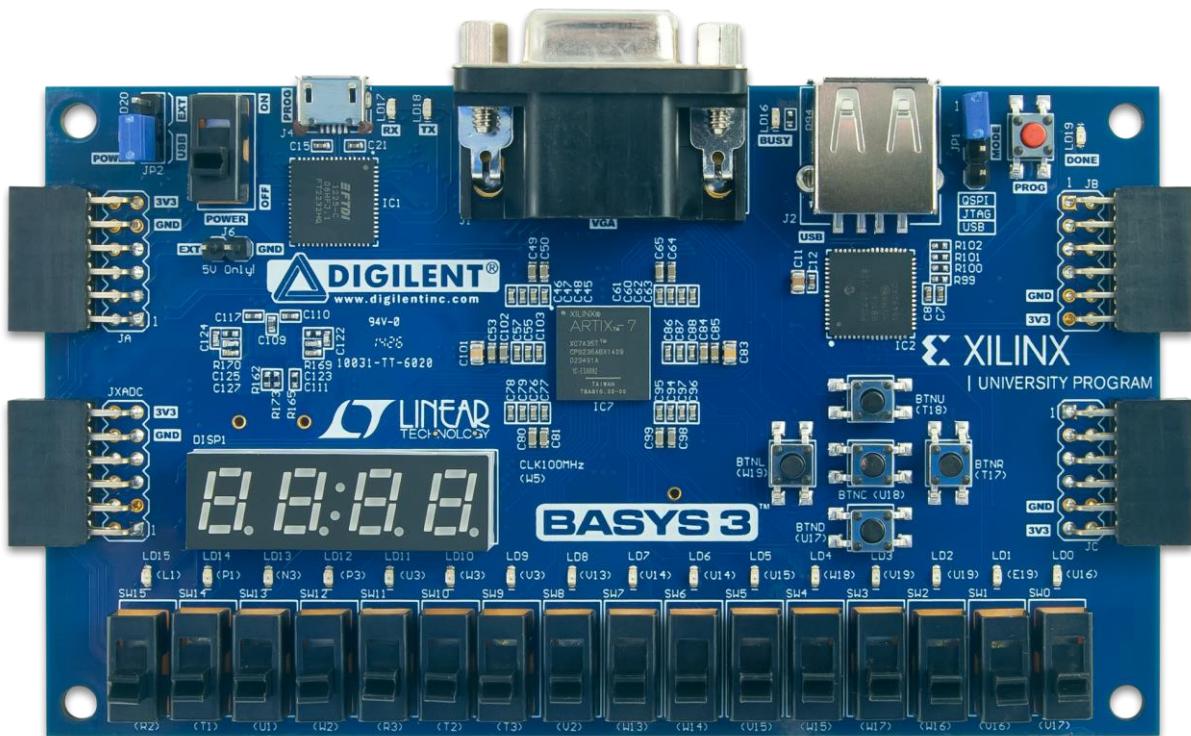


Fig. 1. Placa Basys 3

## 1.2 Considerații teoretice

**PWM (Pulse Width Modulation)** este o tehnică folosită pentru a varia în mod controlat tensiunea dată unui dispozitiv electronic. Această metodă schimbă foarte rapid tensiunea oferită dispozitivului respectiv din ON (1 logic) în OFF (0 logic) și invers. Perioada de timp corespunzătoare valorii ON dintr-un ciclu ON-OFF se numește factor de umplere (duty cycle)

și reprezintă, în medie, ce tensiune va primi dispozitivul electronic. Astfel, se pot controla circuitele analogice din domeniul digital. Astfel, se pot controla circuitele analogice din domeniul digital. Practic, asta înseamnă că un LED acționat astfel se va putea aprinde / stinge gradual.

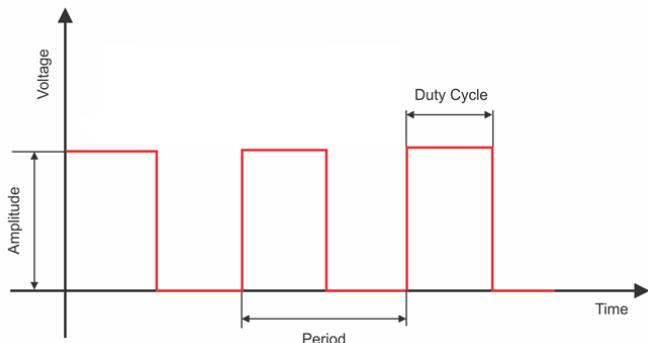


Fig.2. Forma de undă

Factorul de umplere (Duty cycle) se exprimă în procente și reprezintă cât la sută din perioada unui semnal acesta va fi pe nivelul ON (1 logic). În figura 3 se pot observa semnalele PWM cu factori de umplere diferenți. Astfel, se poate deduce foarte ușor formula pentru a obține valoarea factorului de umplere (D):  $D = \frac{t_{on}}{t_{on}+t_{off}} * 100 = \frac{\text{pulse\_width}}{\text{period}} * 100$

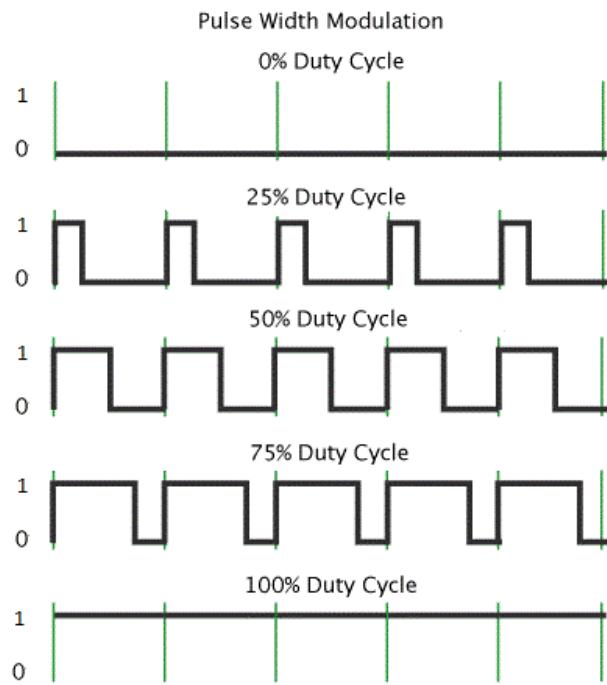


Fig. 3. Semnal PWM cu diferenți factori de umplere

## BCD - 7 segmente

Pentru afişarea informației binare în zecimal se utilizează afișoare care au 7 leduri grupate ca în figura 4 și un punct zecimal. Cu ajutorul acestor segmente se pot scrie cifrele zecimale ca în figura 5.

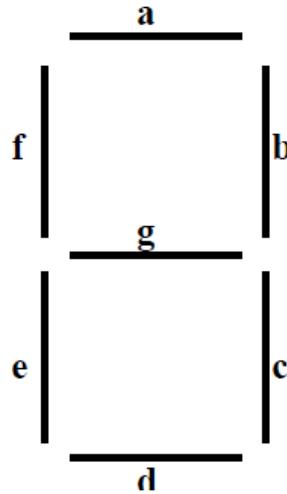


Fig. 4. Afișor 7 segmente

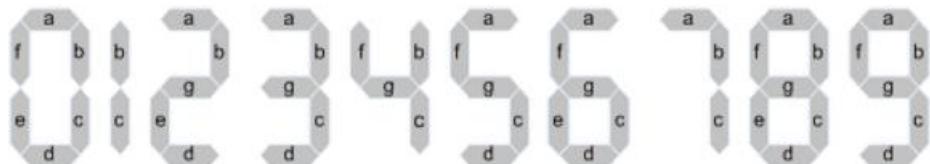


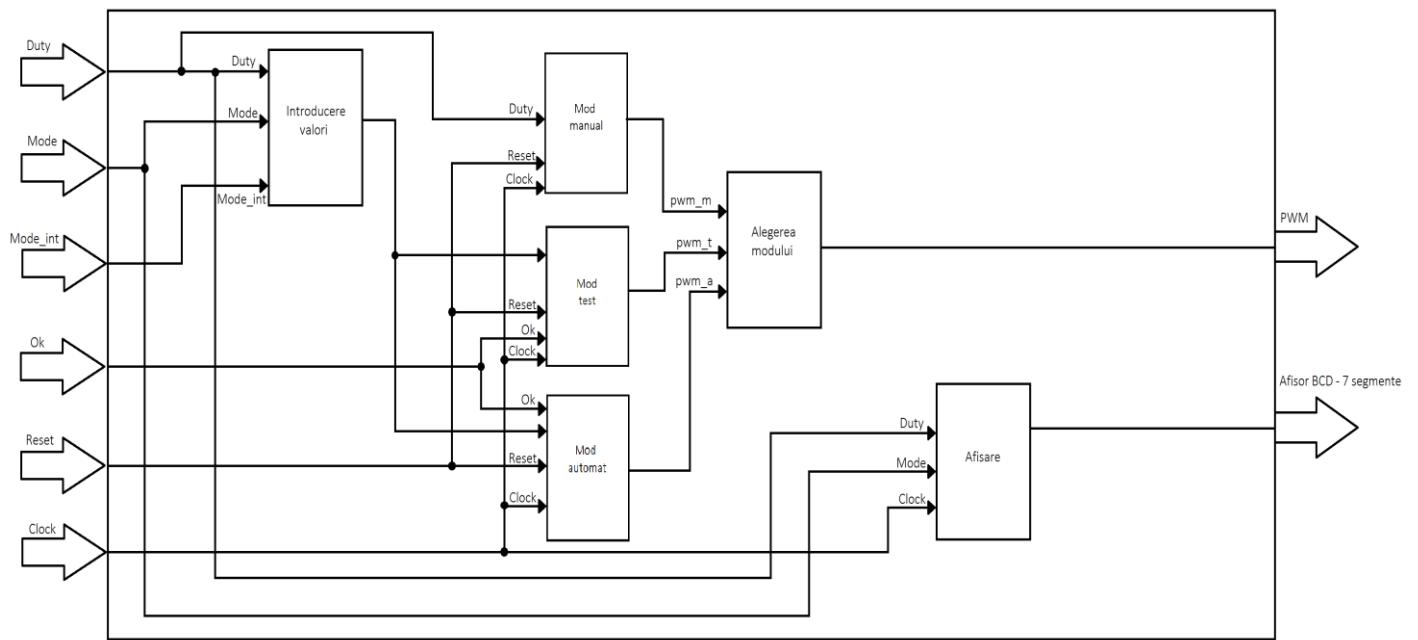
Fig. 5. Combinațiile binare pe 4 biți afișate pe 7 segmente

Afișoarele pot fi construite cu anod sau cu catod comun. Fiecare segment este comandat separat de către o intrare a afișorului. Valoarea logică (0 sau 1 logic) pe care o aplicăm pe cele 7 segmente și pe punctul zecimal, le va aprinde sau stinge, în funcție de tipul afișorului.

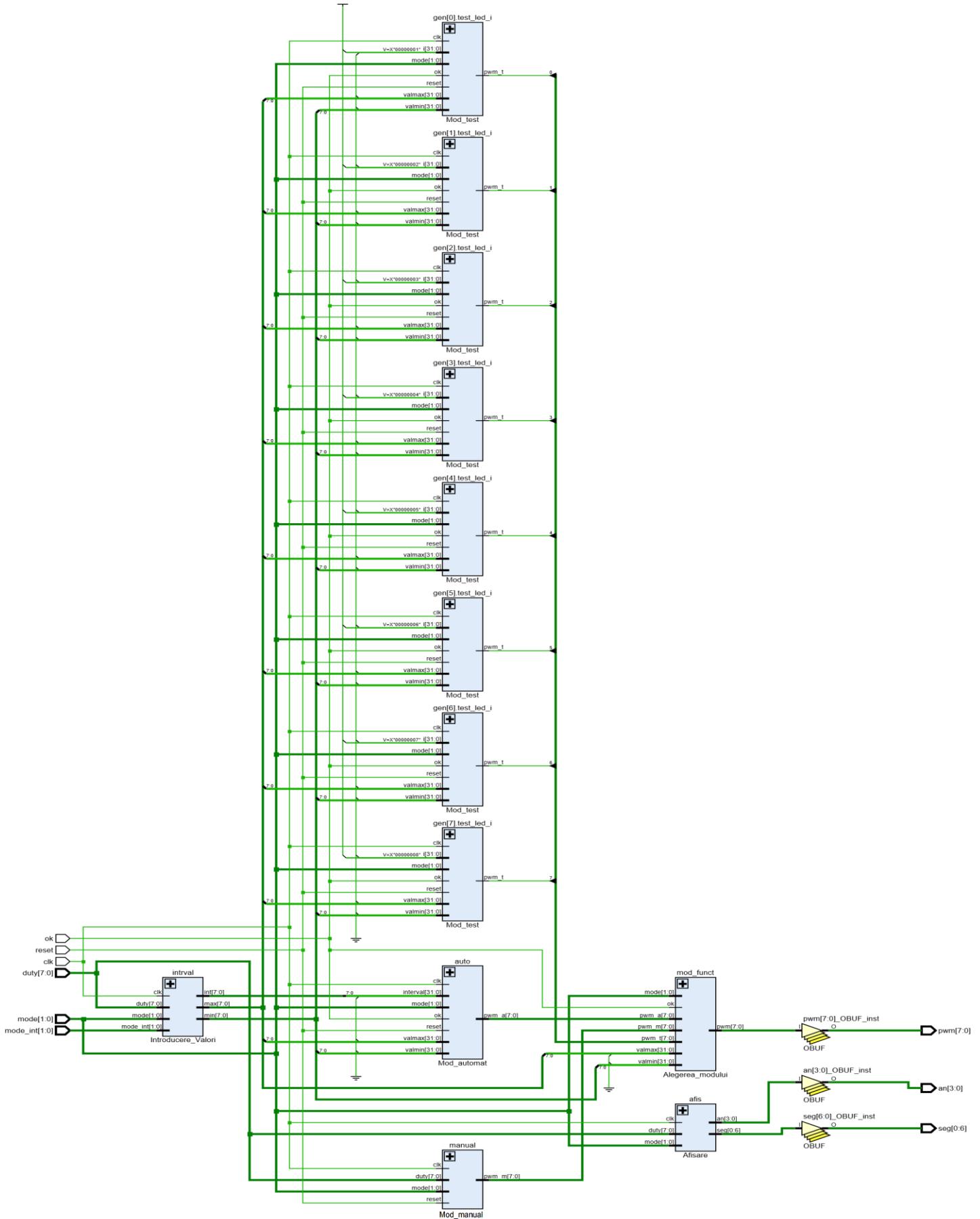
Display	D	C	B	A	a	b	c	d	e	f	g
0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	1	1	0	0	1	1	1	1
2	0	0	1	0	0	0	1	0	0	1	0
3	0	0	1	1	0	0	0	0	1	1	0
4	0	1	0	0	1	0	0	1	1	0	0
5	0	1	0	1	0	1	0	0	1	0	0
6	0	1	1	0	0	1	0	0	0	0	0
7	0	1	1	1	0	0	0	1	1	1	1
8	1	0	0	0	0	0	0	0	0	0	0
9	1	0	0	1	0	0	0	1	0	0	0

## 2. Schema bloc a proiectului

---



- Schemă generată de programul Vivado



### **3. Etapele de proiectare**

---

#### **3.1 Lista componentelor utilizate**

##### **I. Afisare**

Componenta realizează afişarea valorilor introduse de utilizator și a modului selectat de acesta. Ea este una dintre cele mai importante componente care ajuta la formarea interfeței cu exteriorul.

##### **II. Introducere\_valori**

Această componentă procesează datele introduse de utilizator.

##### **III. Mod\_manual**

Componenta realizează modul de funcționare manual al sistemului – valoarea intensității luminoase a ledurilor se furnizează de pe întrerupătoare (8 biți).

##### **IV. Mod\_test**

Această componentă realizează modul test pentru un singur led, urmând ca în programul principal să asigure funcționarea tuturor ledurilor. Modul test constă prin faptul că intensitatea luminoasă a ledurilor variază de la valoarea minimă la valoarea maximă într-un interval de timp specific fiecărui led (led0 – 1 secundă, led1 – 2 secunde, ... led7 – 8 secunde).

##### **V. Mod\_automat**

Această componentă înglobează modul de funcționare automat – intensitatea luminoasă a ledurilor variază de la valoarea minimă la valoarea maxima și înapoi la minim într-un interval de timp măsurat în secunde.

##### **VI. Alegerea\_modului**

O altă componentă importantă este alegerea\_modului deoarece aceasta permite selectarea modului de funcționare în funcție de nevoia utilizatorului.

##### **VII. Main**

Main-ul are rolul de a lega toate componentele și de a asigura o bună funcționare a acestora.

## 3.2 Cod Componente (Explicații)

### I. Afisare

Procesul nr\_afis implementează un numărător care numără în bucla 0 – 3. Cu ajutorul acestuia, se pot afişa pe toate cele 4 afişoare cifre diferite. Pentru o afişare mai uşoară se face conversia valorii din STD\_LOGIC\_VECTOR în INTEGER și se împarte numărul pe cifre. Afişarea se va realiza, în procesul afis, în următorul fel:

- Dacă count\_afis este egal cu 0, se activează ultimul anod și se afişează cifra unităților;
- Dacă count\_afis este egal cu 1, se activează numai anodul 3 și se reprezintă cifra zecilor;
- Dacă count\_afis este egal cu 2, se activează doar al doilea anod și se reprezintă cifra sutelor;
- Dacă count\_afis este egal cu 3, se activează primul anod pe care apare modul selectat.
- Acest lucru este posibil deoarece procesul se realizează la o frecvență de 20.000 Hz, frecvență la care ochiul percep ca și când toate cifrele se afişează în același timp.

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3 use IEEE.NUMERIC_STD.ALL;
4
5 entity Afisare is
6 port(
7     clk : in STD_LOGIC;                      -- Clock-ul de 100 MHz
8     mode : in STD_LOGIC_VECTOR (1 downto 0);  -- Modul de functionare: manual, test sau automat
9     duty : in STD_LOGIC_VECTOR (7 downto 0);   -- Vector de 8 biti care reprezinta intrarea sistemului si este data de 8 switch-uri
10    an : out STD_LOGIC_VECTOR (3 downto 0);    -- Anodurile afisorului
11    seg : out STD_LOGIC_VECTOR (0 to 6);        -- Cele 7 segmente ale afisorului
12 );
13 end Afisare;
14
15 architecture BCD of Afisare is
16     signal count_afis: INTEGER range 0 to 3;
17 begin
18
19     -- Numara în bucla 0 - 3, pentru a putea afisa cifre diferite pe toate cele 4 afisoare în același timp
20     nr_afis: process(clk)
21         variable counter : INTEGER range 0 to 49999;
22     begin
23         if rising_edge(clk) then
24             if counter = 49999 then
25                 counter := 0;
26                 count_afis <= count_afis + 1;
27             else
28                 counter := counter + 1;
29             end if;
30         end if;
31     end process nr_afis;
32 
```

```

33 afis: process(count_afis, duty, mode)
34     variable induty: INTEGER range 0 to 260;
35     variable uc1, uc2, uc3 : INTEGER range 0 to 9;
36     variable mcontrol: INTEGER range 0 to 3;
37 begin
38     -- Se face conversia datelor duty si mode din STD_LOGIC_VECTOR in INTEGER
39     induty := TO_INTEGER(unsigned(duty));
40     mcontrol := TO_INTEGER(unsigned(mode));
41     -- Se calculeaza cele trei cifre ale valorii introduse de pe switch-uri
42     uc1 := induty rem 10;           -- uc1 reprezinta ultima cifra a numarului
43     uc2 := (induty/10) rem 10;      -- uc2 este a doua cifra a numarului
44     uc3 := (induty/100) rem 10;     -- uc3 este prima cifra a numarului
45     -- Se afiseaza cele trei cifre
46     if count_afis = 0 then
47         an <= "1110";
48         case uc1 is
49             when 0 => seg <= "0000001"; -- 0
50             when 1 => seg <= "1001111"; -- 1
51             when 2 => seg <= "0010010"; -- 2
52             when 3 => seg <= "0000110"; -- 3
53             when 4 => seg <= "1001100"; -- 4
54             when 5 => seg <= "0100100"; -- 5
55             when 6 => seg <= "0100000"; -- 6
56             when 7 => seg <= "0001111"; -- 7
57             when 8 => seg <= "0000000"; -- 8
58             when 9 => seg <= "0000100"; -- 9
59             when others => seg <= "1000000";
60         end case;
61     elsif count_afis = 1 then
62         an <= "1101";
63         case uc2 is
64             when 0 => seg <= "0000001"; -- 0
65             when 1 => seg <= "1001111"; -- 1
66             when 2 => seg <= "0010010"; -- 2
67             when 3 => seg <= "0000110"; -- 3
68             when 4 => seg <= "1001100"; -- 4
69             when 5 => seg <= "0100100"; -- 5
70             when 6 => seg <= "0100000"; -- 6
71             when 7 => seg <= "0001111"; -- 7
72             when 8 => seg <= "0000000"; -- 8
73             when 9 => seg <= "0000100"; -- 9
74             when others => seg <= "1000000";
75         end case;
76     elsif count_afis = 2 then
77         an <= "1011";
78         case uc3 is
79             when 0 => seg <= "0000001"; -- 0
80             when 1 => seg <= "1001111"; -- 1
81             when 2 => seg <= "0010010"; -- 2
82             when others => seg <= "1000000";
83         end case;
84     -- Se afiseaza pe primul afisor modul de functionare: manual - 1, test - 2 sau automat - 3
85     elsif count_afis = 3 then
86         an <= "0111";
87         case mcontrol is
88             when 0 => seg <= "0000001"; -- 0
89             when 1 => seg <= "1001111"; -- 1
90             when 2 => seg <= "0010010"; -- 2
91             when 3 => seg <= "0000110"; -- 3
92         end case;
93     end if;
94     -- Daca nu este selectat niciun mod atunci ramane doar primul afisor aprins si se indica valoarea 0, adica nefunctional
95     if mcontrol = 0 then
96         an <= "0111";
97         seg <= "0000001";
98     end if;
99 end process afis;
100
101 end BCD;

```

## II. Introducere\_valori

Componența procesează datele introduse de utilizator. Dacă semnalul mode\_int este egal cu "01" atunci valoarea de pe switch-uri va fi stocată în semnalul min (valoarea minimă). Dacă mode\_int are valoarea egală cu "10" atunci valoarea de pe întrerupătoare se salvează în semnalul max (valoarea maximă), iar în cazul în care semnalul este egal cu "11" valoarea se va stoca în semnalul int (intervalul de timp).

```
1library IEEE;
2use IEEE.STD_LOGIC_1164.ALL;
3
4entity Introducere_Valori is
5    port(
6        clk : in STD_LOGIC;                                -- Clock
7        mode : in STD_LOGIC_VECTOR (1 downto 0);          -- Modul de functionare: manual, test sau automat
8        mode_int : in STD_LOGIC_VECTOR (1 downto 0);        -- Mod pentru introducerea datelor: valmin, valmax, interval
9        duty : in STD_LOGIC_VECTOR (7 downto 0);           -- Vector de 8 biti care reprezinta intrarea sistemului si este data de 8 switch-uri
10       min, max, int : out STD_LOGIC_VECTOR (7 downto 0)  -- Valoarea minima, valoarea maxima si intervalul de timp - vectori de biti
11    );
12end Introducere_Valori;
13
14architecture Introducere_Valori of Introducere_Valori is
15begin
16
17    -- In functie de mode_int se stocheaza datele introduse de pe switch-uri in min, max si int
18    process(mode_int, duty)
19    begin
20        if mode_int = "01" then                         -- Daca mode_int = "01" atunci intrarea de 8 biti se stocheaza in min
21            min <= duty;
22        elsif mode_int = "10" then                      -- Daca mode_int = "10" atunci intrarea de 8 biti se stocheaza in max
23            max <= duty;
24        elsif mode_int = "11" then                      -- Daca mode_int = "11" atunci intrarea de 8 biti se stocheaza in int
25            int <= duty;
26        end if;
27    end process;
28
29end Introducere_valori;
```

### III. Mod\_manual

Pentru rezolvarea modului manual avem nevoie de două procese. Primul implementează un numărător care este incrementat periodic și care este resetat la sfârșitul fiecărei perioade a PWM-ului (numără în bucla 0 – 255). Cel de-al doilea proces compară valoarea numărătorului cu valoarea de referință introdusă ca intrare. Cât timp valoarea numărătorului este mai mică decât intrarea sistemului, ieșirea primește valoarea 1 logic, iar în caz contrar, acesteia îi este asignată valoarea 0 logic.

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3 use IEEE.STD_LOGIC_UNSIGNED.all;
4
5 entity Mod_manual is
6   port(
7     clk : in STD_LOGIC;                      -- Clock
8     mode : in STD_LOGIC_VECTOR (1 downto 0);    -- Modul de functionare: manual, test sau automat
9     duty : in STD_LOGIC_VECTOR (7 downto 0);      -- Vector de 8 biti care reprezinta intrarea sistemului si este data de 8 switch-uri
10    reset: in STD_LOGIC;                      -- Semnal de reset
11    pwm_m : out STD_LOGIC_VECTOR (7 downto 0)    -- Iesirea pentru modul manual
12  );
13 end Mod_manual;
14
15 architecture Mod_manual of Mod_manual is
16   signal count_m: STD_LOGIC_VECTOR (7 downto 0) := "00000000";
17 begin
18   -- Numara de la 0 la 255, 255 = "11111111"
19   cnt_man: process(clk, mode)
20   begin
21     if mode = "01" then
22       if rising_edge(clk) then
23         if count_m = "11111111" then          -- Daca count_m a ajuns la valoarea maxima, "11111111",
24           count_m <= "00000000";            -- atunci count_m se reseteaza
25         else                                -- altfel continua sa numere
26           count_m <= count_m + 1;
27         end if;
28       end if;
29     end if;
30   end process cnt_man;
31
32   -- se formeaza forma de unda
33   manual: process(mode, count_m, duty, reset)
34   begin
35     if mode = "01" then
36       if count_m < duty then              -- Daca valoarea numaratorului este mai mica decat intrarea de date de pe
37         pwm_m <= "11111111";            -- cele 8 switch-uri, atunci iesirea primeste "11111111", iar in caz contrar
38       else                                -- primeste "00000000"
39         pwm_m <= "00000000";
40       end if;
41     end if;
42     if reset = '1' then                  -- Daca semnalul de reset este activ, atunci, ledurile vor ramane stinse
43       pwm_m <= "00000000";
44     end if;
45   end process;
46 end Mod_manual;
```

## IV. Mod\_test

Pentru realizarea modului test avem nevoie de alte două procese, în plus față de modul manual. Primul dintre ele este un divizor de frecvență, care divizează clock-ul de 100 MHz utilizând formula:  $50.000.000 * i / (\text{valmax} - \text{valmin})$ . Placa Basys 3 generează un semnal de clock cu frecvență de 100MHz, ceea ce înseamnă că într-un interval de 1 secundă valoarea clock-ului se schimbă de un număr de 100.000.000 de ori. Pentru a-l diviza este nevoie de un numărător care se incrementează pe front crescător. Așadar, pentru a obține un clock cu frecvență de 1 Hz, este nevoie ca, doar în momentul în care se înregistrează un număr de 50.000.000 de schimbări de tact pe front crescător, noul clock să își schimbe valoarea. Având clock-ul de 1 Hz, pentru a obține intervalul de timp specific fiecarui led, mai este necesar să înmulțim valoarea de 50.000.000 cu intervalul de timp dorit (semnalul *i*, în cazul nostru), iar apoi să o împărțim la diferența dintre valoarea maximă și valoarea minimă (diferența reprezentând numărul de stări care trebuie numărate). Cel de-al doilea proces reprezintă un numărător, având semnalul de tact divizat, care numără de la valoarea minima la cea maximă și care se resetează în valoarea minimă. O altă diferență între modul manual și modul test constă în faptul că, la modul test, valoarea primului numărător nu mai este comparată cu intrarea sistemului, ci cu valoarea numărătorului care se incrementează de la valoarea minimă la cea maximă.

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3 use IEEE.STD_LOGIC_UNSIGNED.ALL;
4 use IEEE.NUMERIC_STD.ALL;
5
6 entity Mod_test is
7   port(
8     clk : in STD_LOGIC;                      -- Clock de 100 MHz
9     mode : in STD_LOGIC_VECTOR (1 downto 0);  -- Modul de functionare: manual, test sau automat
10    ok : in STD_LOGIC;                        -- Semnal care îi permite sistemului să funcționeze doar după introducerea valorilor min, max și int
11    reset: in STD_LOGIC;                     -- Semnal de reset
12    i: in INTEGER;                          -- Semnal care reprezintă un indice - componenta realizează modul test pentru ledul cu indicele i
13    valmin, valmax: INTEGER;                -- Valoarea minima și valoarea maxima - valori întregi
14    pwm_t : out STD_LOGIC                  -- Iesirea pentru modul test a unui singur led
15  );
16 end Mod_test;
17
18 architecture Mod_test of Mod_test is
19   signal count_t: INTEGER range 0 to 256;  -- Numara în bucla 0 - 255
20   signal clkdiv_t: STD_LOGIC;               -- Clock divizat
21   signal count: INTEGER;                   -- Număratore care se folosește pentru a diviza frecvența clock-ului
22   signal duty_t: INTEGER range 0 to 256;   -- Numărătoare care contine diferența dintre valoarea maxima și valoarea minima
23   signal dif: INTEGER range 0 to 256;
24 begin
25   dif <= valmax - valmin;                 -- Numărătoare care contine diferența dintre valoarea maxima și valoarea minima
26   -- Numără în bucla 0 - 255
27   cnt_t: process(clk, ok, mode)
28   begin
29     if mode = "10" and ok = '1' then
30       if rising_edge(clk) then
31         if count_t = 255 then
32           count_t <= 0;
33         else
34           count_t <= count_t + 1;
35         end if;
36       end if;
37     end if;
38   end process;
39
```

```

40 -- Divizeaza clock-ul pentru ledul i cu ajutorul formulei 50.000.000 * i / (valmax - valmin)
41 div_test: process(clk, ok, mode, reset)
42 begin
43     if mode = "10" and ok = '1' then
44         if rising_edge(clk) then
45             if count = (50_000_000 * i / dif) then -- i reprezinta indicele ledului si implicit numarul de secunde
46                 count <= 0;                         -- in care acesta trebuie sa ajunga de la valmin la valmax
47                 clkdiv_t <= not clkdiv_t;
48             else
49                 count <= count + 1;
50             end if;
51         end if;
52     end if;
53     if reset = '1' then
54         count <= 0;
55         clkdiv_t <= '0';
56     end if;
57 end process;
58 -- Numara de la valoarea minima la valoarea maxima cu un clock divizat pentru a parcurge toate
59 -- aceste valori in intervalul de timp dorit (led0 - 1 secunda, led1 - 2 secunde, ... led7 - 8 secunde)
60 duty: process(clkdiv_t, ok, mode, reset)
61 begin
62     if mode = "10" and ok = '1' then
63         if rising_edge(clkdiv_t) then
64             if duty_t = valmax then           -- Cand numaratorul ajunge la valoarea maxima (valmax)
65                 duty_t <= valmin;          -- acesta se reseteaza in valoarea minima (valmin)
66             else
67                 duty_t <= duty_t + 1;
68             end if;
69         end if;
70     end if;
71     if reset = '1' then
72         duty_t <= 0;
73     end if;
74 end process;
75
76 -- se formeaza forma de unda
77 test_led: process (count_t, duty_t, ok, mode, reset)
78 begin
79     if mode = "10" and ok = '1' then
80         if count_t < duty_t then        -- Daca valoarea numaratorului este mai mica decat intrarea de date de pe
81             pwm_t <= '1';              -- cele 8 switch-uri, atunci iesirea, pentru ledul i, primeste '1', iar in
82         else                           -- caz contrar primeste '0'
83             pwm_t <= '0';
84         end if;
85     end if;
86     if reset = '1' then               -- Daca semnalul de reset este activ, atunci, ledurile vor ramane stinse
87         pwm_t <= '0';
88     end if;
89 end process;
90
91 end Mod_test;

```

## V. Mod\_automat

Realizarea modului de funcționare automat este identică cu cea a modului test, cu excepția a 3 lucruri:

- Intervalul de timp nu mai este specific fiecărui led ci este identic tuturor ledurilor, fiind introdus ca o intrare a sistemului;
- Avem nevoie de un numărător bidirectional în detrimentul celui direct, care să numere de la valoarea minimă, la valoarea maximă și înapoi la cea minimă, având un semnal de tact divizat;
- Mai avem nevoie de folosirea unui proces care ajută la formarea numărătorului bidirectional, în următorul mod: când numărătorul ajunge la valoarea maximă semnalul k primește valoarea 1 logic, iar când numărătorul ajunge la valoarea minima semnalul k trece înapoi în 0 logic. În funcție de valoarea lui k numărătorul numără direct sau invers.

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4 entity Mod_automat is
5   port(
6     clk : in STD_LOGIC;          -- Semnalul de clock
7     mode : in STD_LOGIC_VECTOR (1 downto 0); -- Modul de functionare: manual, test sau automat
8     ok : in STD_LOGIC;           -- Semnal care ii permite sistemului sa functioneze doar dupa introducerea valorilor min, max si int
9     reset: in STD_LOGIC;         -- Semnal de reset
10    valmin, valmax, interval: INTEGER;      -- Valoarea minima, valoarea maxima si intervalul - valori intregi
11    pwm_a : out STD_LOGIC_VECTOR (7 downto 0) -- Iesirea pentru modul test
12  );
13 end Mod_automat;
14
15 architecture Mod_automat of Mod_automat is
16   signal counter : INTEGER range 0 to 256;
17   signal counter_a : INTEGER;
18   signal clkdiv_a : STD_LOGIC;
19   signal duty_a : INTEGER range 0 to 256;
20   signal k : STD_LOGIC;
21   signal dif: INTEGER range 0 to 256;
22 begin
23   dif <= valmax - valmin;
24   -- numara in bucla 0 - 255
25   cnt_a: process (clk, ok, mode, reset)
26 begin
27     if ok = '1' and mode = "11" then
28       if rising_edge(clk) then
29         if counter = 255 then
30           counter <= 0;
31         else
32           counter <= counter + 1;
33         end if;
34       end if;
35     end if;
36     if reset = '1' then
37       counter <= 0;
38     end if;
39   end process;
40 
```

```

41 -- se utilizeaza formula 25.000.000 * intervalul_de_timp / (valoarea_maxima - valoarea_minima)
42 -- 25.000.000 = 50.000.000 / 2 - ledul trebuie sa ajunga de la valoarea minima la valoarea
43 -- maxima si inapoi la cea minima in intervalul de timp dat, deci trebuie ca in jumitate
44 -- din interval sa ajunga la maxim, iar in cealalta jumate sa ajunga inapoi la minim
45 div_a: process (ok, mode, clk, reset)
46 begin
47     if rising_edge(clk) and ok = '1' and mode = "11" then
48         if counter_a = ((25_000_000 / dif) * interval) then
49             counter_a <= 0;
50             clkdiv_a <= not clkdiv_a;
51         else
52             counter_a <= counter_a + 1;
53         end if;
54     end if;
55     if reset = '1' then
56         counter_a <= 0;
57         clkdiv_a <= '0';
58     end if;
59 end process;
60
61 -- se stabileste valoarea lui k
62 val_k: process(duty_a, valmax, valmin)
63 begin
64     if duty_a >= valmax then
65         k <= '1';
66     elsif duty_a <= valmin then
67         k <= '0';
68     end if;
69 end process;
70
71 -- numarator bidirectional in functie de k, avand clock-ul divizat
72 duty: process(ok, clkdiv_a, mode, reset)
73 begin
74     if rising_edge(clkdiv_a) and ok = '1' and mode = "11" then
75         if k = '0' then
76             duty_a <= duty_a + 1;
77         else
78             duty_a <= duty_a - 1;
79         end if;
80     end if;
81     if reset = '1' then
82         duty_a <= 0;
83     end if;
84 end process;
85

```

```

86      -- se formeaza forma de unda
87      auto: process (ok, clk, mode)
88      begin
89          if rising_edge(clk) and ok = '1' and mode = "11" then
90              if reset = '0' then
91                  if counter <= duty_a then
92                      pwm_a <= "11111111";
93                  else
94                      pwm_a <= "00000000";
95                  end if;
96              else
97                  pwm_a <= "00000000";
98              end if;
99          end if;
100     end process;
101
102 end Mod_automat;

```

## VI. Alegerea\_modului

Componenta selectează modul de funcționare al sistemului în felul următor:

- Dacă mode este egal cu "01" atunci se selectează modul manual;
- Dacă mode este egal cu "10" atunci se selectează modul test;
- Dacă mode este egal cu "11" atunci se selectează modul automat.

În cazul în care valoarea maximă este mai mică decât valoarea minimă sistemul nu va funcționa în modurile respective.

---

```

1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4 entity Alegerea_modului is
5     port(
6         mode : in STD_LOGIC_VECTOR (1 downto 0);           -- Modul de functionare: manual, test sau automat
7         ok : in STD_LOGIC;                                -- Semnal care îi permite sistemului să funcționeze doar după introducerea valorilor min, max și int
8         pwm_m, pwm_t, pwm_a: in STD_LOGIC_VECTOR (7 downto 0); -- Iesirile pentru fiecare mod: manual, test și automat
9         valmin, valmax: in INTEGER;                      -- Valoarea minima și valoarea maxima - întregi
10        pwm : out STD_LOGIC_VECTOR (7 downto 0);          -- Iesirea sistemului
11    );
12 end Alegerea_modului;
13
14 architecture Alegerea_modului of Alegerea_modului is
15 begin
16     -- se face selectia in functie de intrarea mode
17     process (mode, pwm_m, pwm_t, pwm_a, valmax, valmin, ok)
18     begin
19         if (valmax - valmin) >= 0 and (mode = "10" or mode = "11") then
20             if mode = "10" and ok = '1' then
21                 pwm <= pwm_t;
22             elsif mode = "11" and ok = '1' then
23                 pwm <= pwm_a;
24             else
25                 pwm <= "00000000";
26             end if;
27         elsif mode = "01" then
28             pwm <= pwm_m;
29         else
30             pwm <= "00000000";
31         end if;
32     end process;
33
34 end Alegerea_modului;

```

## VII. Main

Main-ul reprezintă programul principal care face legătura între toate componentele, ceea ce face din el una dintre cele mai importante părți ale întregului program.

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3 use IEEE.STD_LOGIC_UNSIGNED.all;
4 use IEEE.NUMERIC_STD.ALL;
5
6 entity main is
7     port(
8         clk : in STD_LOGIC;                                -- Clock de 100 MHz
9         mode : in STD_LOGIC_VECTOR (1 downto 0);          -- Modul de funcționare: manual, test sau automat
10        duty : in STD_LOGIC_VECTOR (7 downto 0);          -- Vector de 8 biti care reprezinta intrarea sistemului si este data de 8 switch-uri
11        mode_int : in STD_LOGIC_VECTOR (1 downto 0);       -- Mod pentru introducerea datelor: valmin, valmax, interval
12        ok : in STD_LOGIC;                                 -- Semnal care ii permite sistemului sa functioneze doar dupa introducerea valorilor min, max si int
13        reset: in STD_LOGIC;                             -- Semnal de reset
14        an : out STD_LOGIC_VECTOR (3 downto 0);           -- anodurile afisorului
15        seg : out STD_LOGIC_VECTOR (0 to 6);              -- Cele 7 segmente ale afisorului
16        pwm : out STD_LOGIC_VECTOR (7 downto 0);          -- Iesirea sistemului
17    );
18 end main;
19
20 architecture main of main is
21
22 signal min, max, int: STD_LOGIC_VECTOR (7 downto 0);
23 signal pwm_m, pwm_t, pwm_a: STD_LOGIC_VECTOR (7 downto 0);
24 signal valmin, valmax, interval: INTEGER range 0 to 256;
25 -- Componenta care afiseaza valoarea introdusa de pe switch-uri si modul ales
26 component Afisare is
27     port(
28         clk : in STD_LOGIC;
29         mode: in STD_LOGIC_VECTOR (1 downto 0);
30         duty : in STD_LOGIC_VECTOR (7 downto 0);
31         an: out STD_LOGIC_VECTOR (3 downto 0);
32         seg: out STD_LOGIC_VECTOR (0 to 6)
33     );
34 end component Afisare;
35 -- Componenta care se ocupa de procesarea datelor de intrare
36 component Introducere_Valori is
37     port(
38         clk : in STD_LOGIC;
39         mode : in STD_LOGIC_VECTOR (1 downto 0);
40         mode_int : in STD_LOGIC_VECTOR (1 downto 0);
41         duty : in STD_LOGIC_VECTOR (7 downto 0);
42         min, max, int : out STD_LOGIC_VECTOR (7 downto 0)
43     );
44 end component Introducere_Valori;
45 -- Componenta care realizeaza modul de funcționare manual
46 component Mod_manual is
47     port(
48         clk : in STD_LOGIC;
49         mode : in STD_LOGIC_VECTOR (1 downto 0);
50         duty : in STD_LOGIC_VECTOR (7 downto 0);
51         reset: in STD_LOGIC;
52         pwm_m : out STD_LOGIC_VECTOR (7 downto 0)
53     );
54 end component Mod_manual;
55 -- Componenta care se ocupa de modul test
56 component Mod_test is
57     port(
58         clk : in STD_LOGIC;
59         mode : in STD_LOGIC_VECTOR (1 downto 0);
60         ok : in STD_LOGIC;
61         reset: in STD_LOGIC;
62         i: in INTEGER;
63         valmin, valmax: integer;
64         pwm_t : out STD_LOGIC
65     );
66 end component Mod_test;
```

```

67 -- Componenta care realizeaza modul automat
68 component Mod_automat is
69     port(
70         clk : in STD_LOGIC;
71         mode : in STD_LOGIC_VECTOR (1 downto 0);
72         ok : in STD_LOGIC;
73         reset: in STD_LOGIC;
74         valmin, valmax, interval: INTEGER;
75         pwm_a : out STD_LOGIC_VECTOR (7 downto 0)
76     );
77 end component Mod_automat;
78 -- Componenta care se ocupa cu alegerea modului
79 component Alegerea_modului is
80     port(
81         mode : in STD_LOGIC_VECTOR (1 downto 0);
82         ok : in STD_LOGIC;
83         pwm_m, pwm_t, pwm_a: in STD_LOGIC_VECTOR (7 downto 0);
84         valmin, valmax: in INTEGER;
85         pwm : out STD_LOGIC_VECTOR (7 downto 0)
86     );
87 end component Alegerea_modului;
88 begin
89     -- Conversia datelor(valoare minima, valoare maxima si interval de timp) din STD_LOGIC_VECTOR in INTEGER
90     valmin <= TO_INTEGER(unsigned(min));
91     valmax <= TO_INTEGER(unsigned(max));
92     interval <= TO_INTEGER(unsigned(int));
93     -- Componenta pentru afisarea intrarii sistemului si a modului selectat
94     afis: Afisare port map (clk, mode, duty, an, seg);
95     -- Componenta pentru stabilirea valorii minime, maxime si a intervalului
96     intrval: Introducere_Valori port map (clk, mode, mode_int, duty, min, max, int);
97     -- Modul manual
98     manual: Mod_manual port map (clk, mode, duty, reset, pwm_m);
99     -- Generate pentru modul test (pentru fiecare led in parte)
100    gen: for i in 0 to 7 generate
101        test_led_i: Mod_test port map(clk, mode, ok, reset, i+1, valmin, valmax, pwm_t(i));
102    end generate;
103    -- Modul automat
104    auto: Mod_automat port map (clk, mode, ok, reset, valmin, valmax, interval, pwm_a);
105    -- Afisarea pe leduri in functie de mod
106    mod_funct: Alegerea_modului port map (mode, ok, pwm_m, pwm_t, pwm_a, valmin, valmax, pwm);
107 end main;

```

## VIII. Fișierul de constrângeri

În acest fișier sunt realizate legăturile dintre semnalele pentru intrările și ieșirile sistemului și pinii placii FPGA.

```
1 ## Clock signal
2 set_property PACKAGE_PIN W5 [get_ports clk]
3     set_property IOSTANDARD LVCMOS33 [get_ports clk]
4
5 ## Switches
6 set_property PACKAGE_PIN V17 [get_ports duty[0]]
7     set_property IOSTANDARD LVCMOS33 [get_ports duty[0]]
8 set_property PACKAGE_PIN V16 [get_ports duty[1]]
9     set_property IOSTANDARD LVCMOS33 [get_ports duty[1]]
10 set_property PACKAGE_PIN W16 [get_ports duty[2]]
11     set_property IOSTANDARD LVCMOS33 [get_ports duty[2]]
12 set_property PACKAGE_PIN W17 [get_ports duty[3]]
13     set_property IOSTANDARD LVCMOS33 [get_ports duty[3]]
14 set_property PACKAGE_PIN W15 [get_ports duty[4]]
15     set_property IOSTANDARD LVCMOS33 [get_ports duty[4]]
16 set_property PACKAGE_PIN V15 [get_ports duty[5]]
17     set_property IOSTANDARD LVCMOS33 [get_ports duty[5]]
18 set_property PACKAGE_PIN W14 [get_ports duty[6]]
19     set_property IOSTANDARD LVCMOS33 [get_ports duty[6]]
20 set_property PACKAGE_PIN W13 [get_ports duty[7]]
21     set_property IOSTANDARD LVCMOS33 [get_ports duty[7]]
22 set_property PACKAGE_PIN T2 [get_ports reset]
23     set_property IOSTANDARD LVCMOS33 [get_ports reset]
24 set_property PACKAGE_PIN R3 [get_ports ok]
25     set_property IOSTANDARD LVCMOS33 [get_ports ok]
26 set_property PACKAGE_PIN W2 [get_ports mode_int[0]]
27     set_property IOSTANDARD LVCMOS33 [get_ports mode_int[0]]
28 set_property PACKAGE_PIN U1 [get_ports mode_int[1]]
29     set_property IOSTANDARD LVCMOS33 [get_ports mode_int[1]]
30 set_property PACKAGE_PIN T1 [get_ports mode[0]]
31     set_property IOSTANDARD LVCMOS33 [get_ports mode[0]]
32 set_property PACKAGE_PIN R2 [get_ports mode[1]]
33     set_property IOSTANDARD LVCMOS33 [get_ports mode[1]]
34
35 ## LEDs
36 set_property PACKAGE_PIN U16 [get_ports pwm[0]]
37     set_property IOSTANDARD LVCMOS33 [get_ports pwm[0]]
38 set_property PACKAGE_PIN E19 [get_ports pwm[1]]
39     set_property IOSTANDARD LVCMOS33 [get_ports pwm[1]]
40 set_property PACKAGE_PIN U19 [get_ports pwm[2]]
41     set_property IOSTANDARD LVCMOS33 [get_ports pwm[2]]
42 set_property PACKAGE_PIN V19 [get_ports pwm[3]]
43     set_property IOSTANDARD LVCMOS33 [get_ports pwm[3]]
44 set_property PACKAGE_PIN W18 [get_ports pwm[4]]
45     set_property IOSTANDARD LVCMOS33 [get_ports pwm[4]]
46 set_property PACKAGE_PIN U15 [get_ports pwm[5]]
47     set_property IOSTANDARD LVCMOS33 [get_ports pwm[5]]
48 set_property PACKAGE_PIN U14 [get_ports pwm[6]]
49     set_property IOSTANDARD LVCMOS33 [get_ports pwm[6]]
50 set_property PACKAGE_PIN V14 [get_ports pwm[7]]
51     set_property IOSTANDARD LVCMOS33 [get_ports pwm[7]]
```

52

```
53 ##7 segment display
54 set_property PACKAGE_PIN W7 [get_ports seg[0]]
55     set_property IOSTANDARD LVC MOS33 [get_ports seg[0]]
56 set_property PACKAGE_PIN W6 [get_ports seg[1]]
57     set_property IOSTANDARD LVC MOS33 [get_ports seg[1]]
58 set_property PACKAGE_PIN U8 [get_ports seg[2]]
59     set_property IOSTANDARD LVC MOS33 [get_ports seg[2]]
60 set_property PACKAGE_PIN V8 [get_ports seg[3]]
61     set_property IOSTANDARD LVC MOS33 [get_ports seg[3]]
62 set_property PACKAGE_PIN U5 [get_ports seg[4]]
63     set_property IOSTANDARD LVC MOS33 [get_ports seg[4]]
64 set_property PACKAGE_PIN V5 [get_ports seg[5]]
65     set_property IOSTANDARD LVC MOS33 [get_ports seg[5]]
66 set_property PACKAGE_PIN U7 [get_ports seg[6]]
67     set_property IOSTANDARD LVC MOS33 [get_ports seg[6]]
68
69 set_property PACKAGE_PIN U2 [get_ports an[0]]
70     set_property IOSTANDARD LVC MOS33 [get_ports an[0]]
71 set_property PACKAGE_PIN U4 [get_ports an[1]]
72     set_property IOSTANDARD LVC MOS33 [get_ports an[1]]
73 set_property PACKAGE_PIN V4 [get_ports an[2]]
74     set_property IOSTANDARD LVC MOS33 [get_ports an[2]]
75 set_property PACKAGE_PIN W4 [get_ports an[3]]
76     set_property IOSTANDARD LVC MOS33 [get_ports an[3]]
```

## 4. Pașii de proiectare

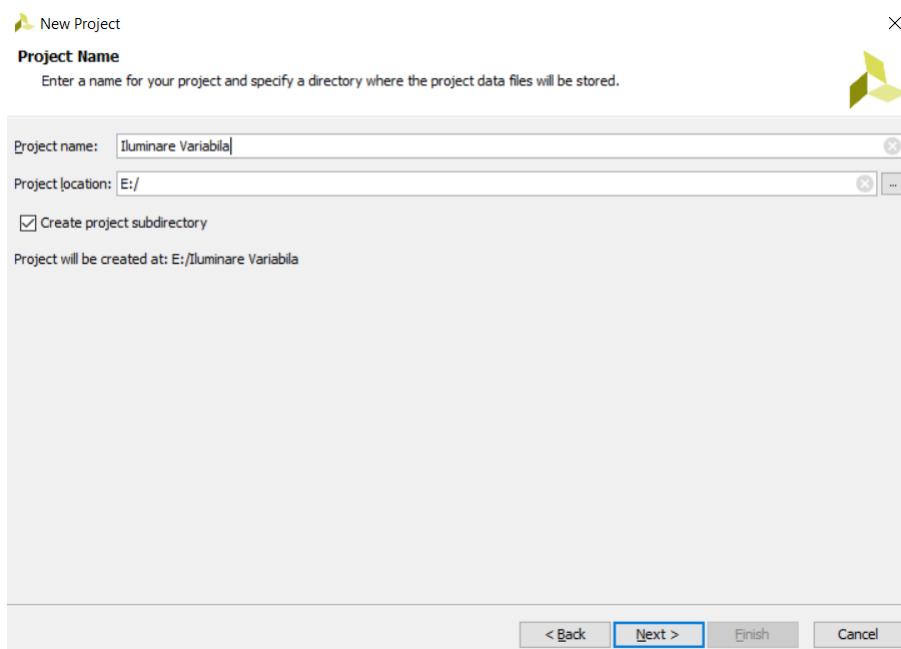
Proiectul dat presupune utilizarea plăcii Basys 3 și implicit a utilitarului Vivado Design Suite dezvoltat de Xilinx.

Pentru încărcarea proiectului pe placă FPGA este necesara respectarea următorilor pași:

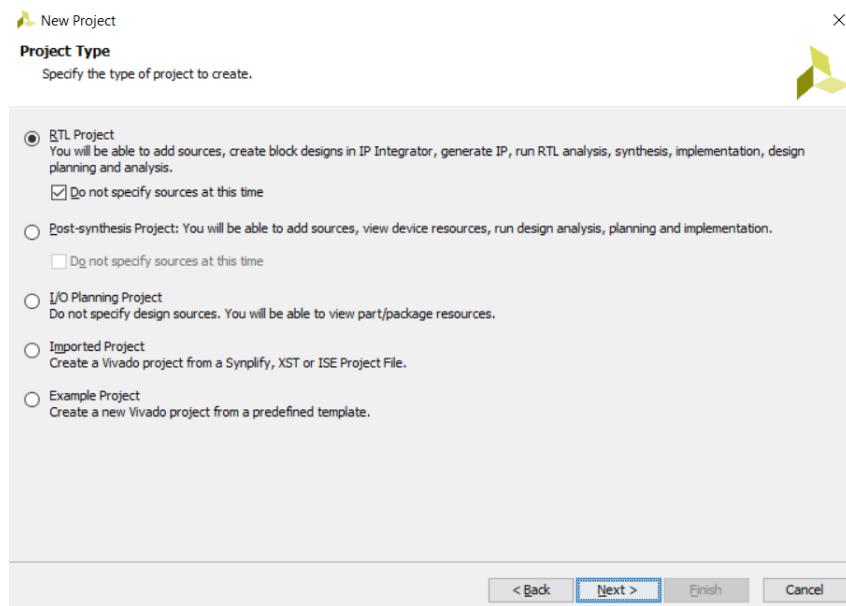
1. Crearea unui proiect nou
  - o Se deschide programul Vivado Design Suite
  - o Click pe **Create New Project**



2. Specificarea unui nume



3. Specificarea tipului de proiect creat
- RTL Project

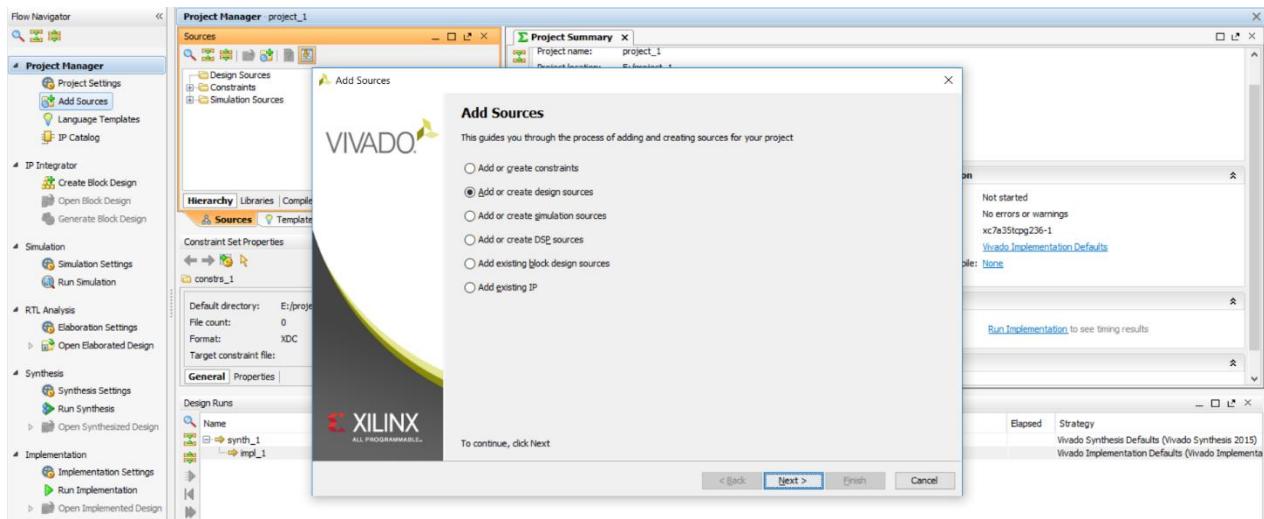


4. Selectarea plăcii Basys 3, având particularitățile precizate în imaginea de mai jos
- Xc7a35tcpg236-1

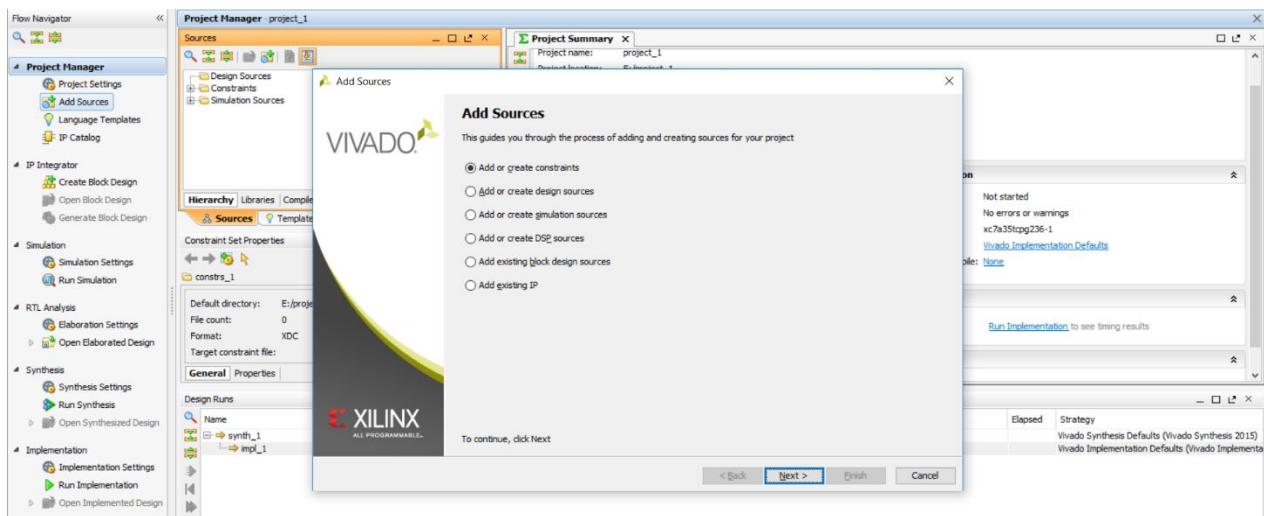
Part	I/O Pin Count	Block RAMs	DSPs	FlipFlops	GTPE2 Transceivers	Gb Transceivers	Available IOBs	LUT Elements
xc7a15tcpg236-1	236	25	45	20800	2	2	106	10400
<b>xc7a35tcpg236-1</b>	<b>236</b>	<b>50</b>	<b>90</b>	<b>41600</b>	<b>2</b>	<b>2</b>	<b>106</b>	<b>20800</b>
xc7a50tcpg236-1	236	75	120	65200	2	2	106	32600

5. Se creează fișierele sursă necesare (componentele proiectului)

- Click pe **Add Sources**, iar mai apoi se selectează opțiunea **Add or create design sources**.



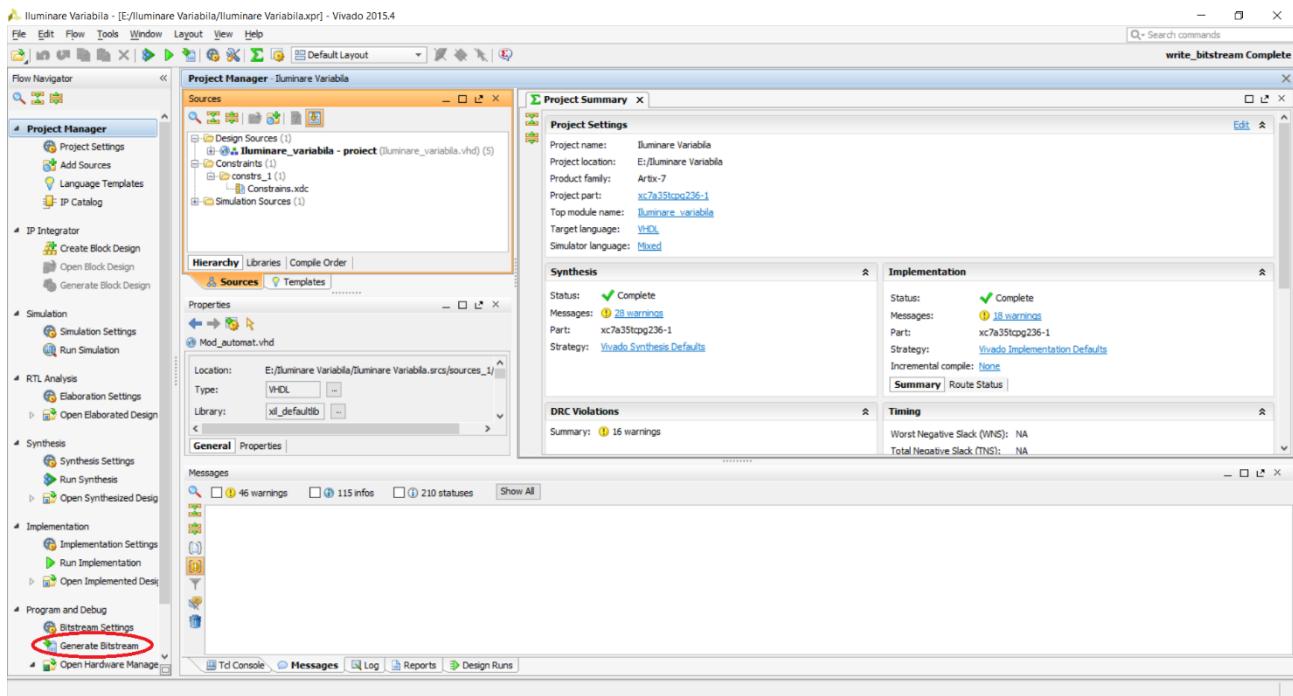
6. Se creează un fișier de constrângeri cu extensia **.xdc** pentru a putea realiza asignarea pinilor pentru intrările și ieșirile sistemului



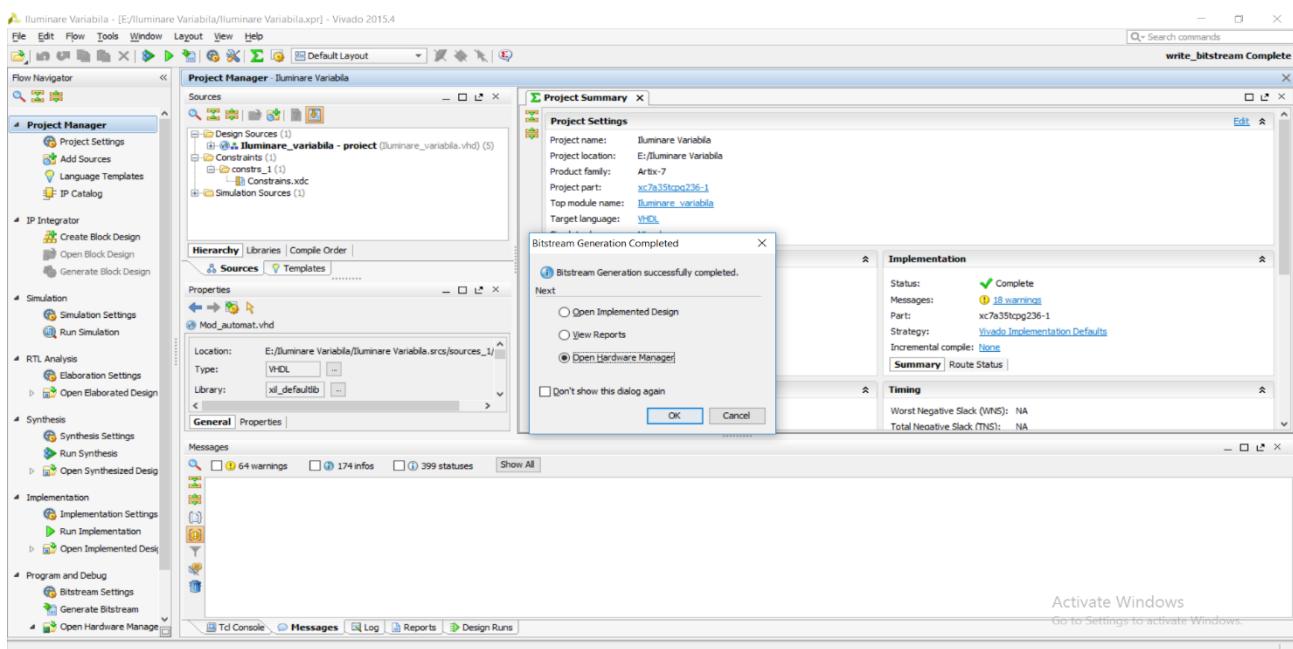
- Click pe **Add Sources**, după care se selectează opțiunea **Add or create constraints**.

## 7. Se generează fișierul cu extensia .bit

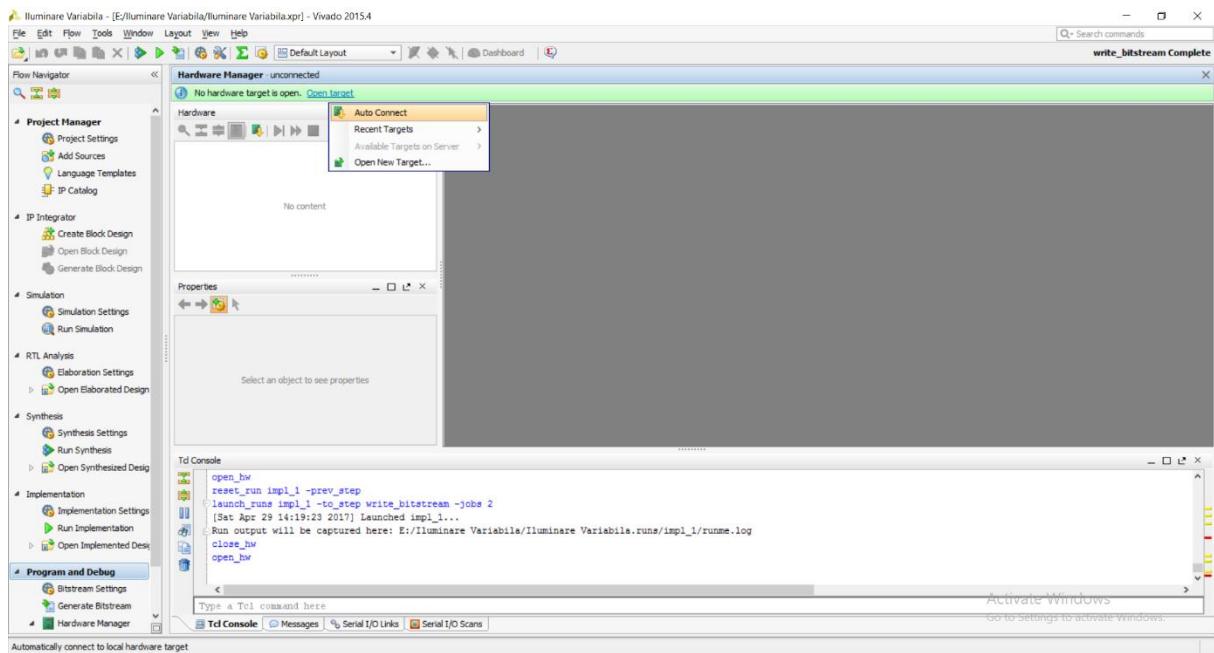
- După crearea tuturor componentelor necesare și scrierea codului VHDL se selectează **Generate Bitstream**



## 8. După crearea fișierului cu extensia .bit se selectează opțiunea Open Hardware Manager

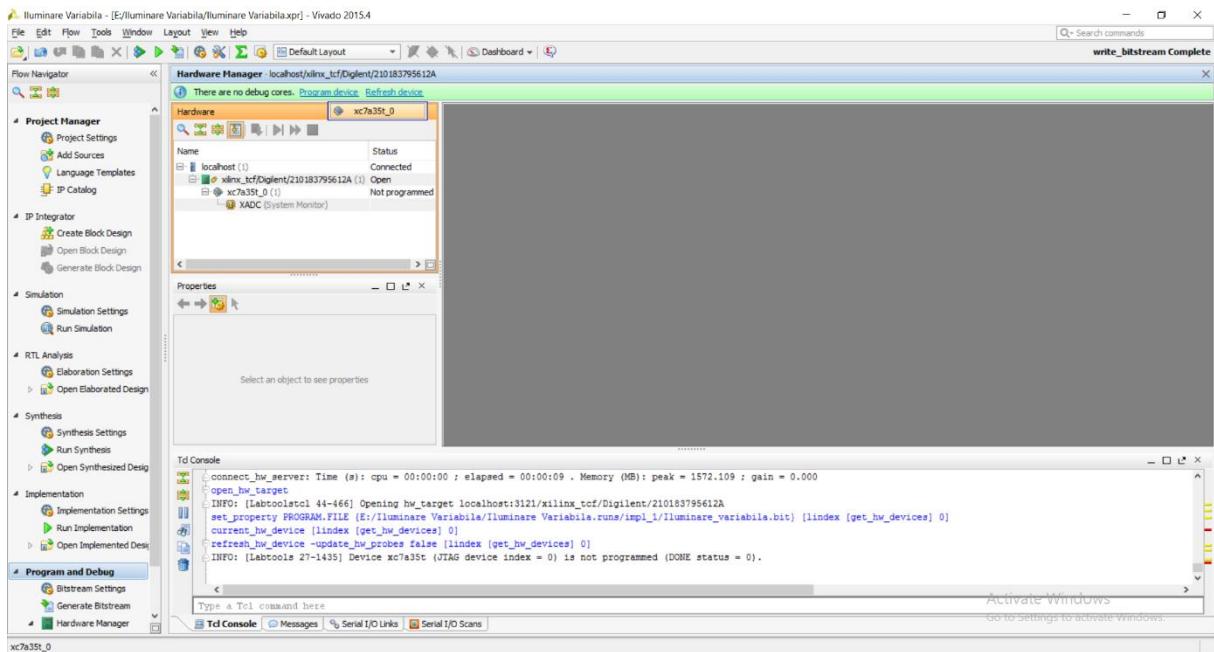


## 9. Click pe Open target, iar după aceea pe Auto Connect

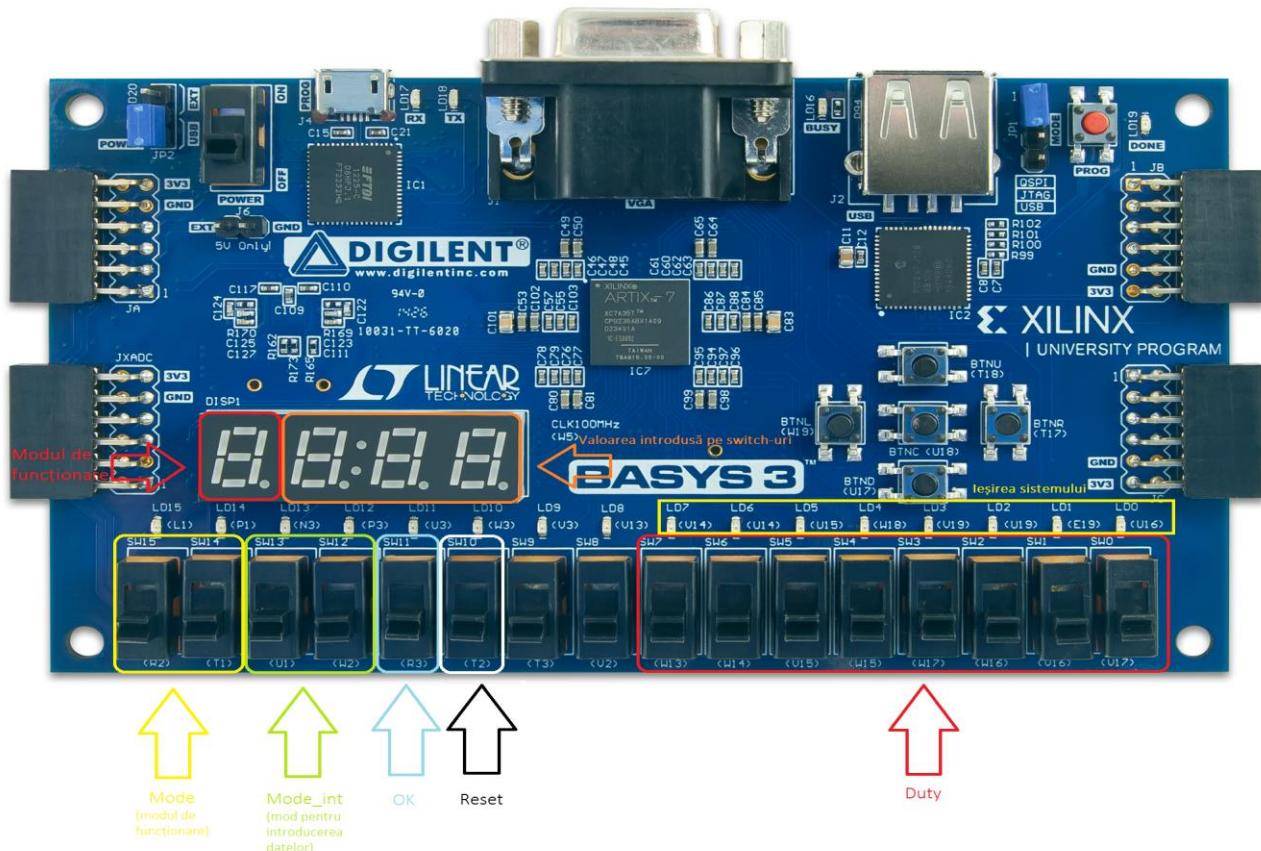


## 10. Ultimul pas care trebuie făcut este programarea plăcii

- Click pe **Program Device**, iar apoi pe **xc7a35t\_0**

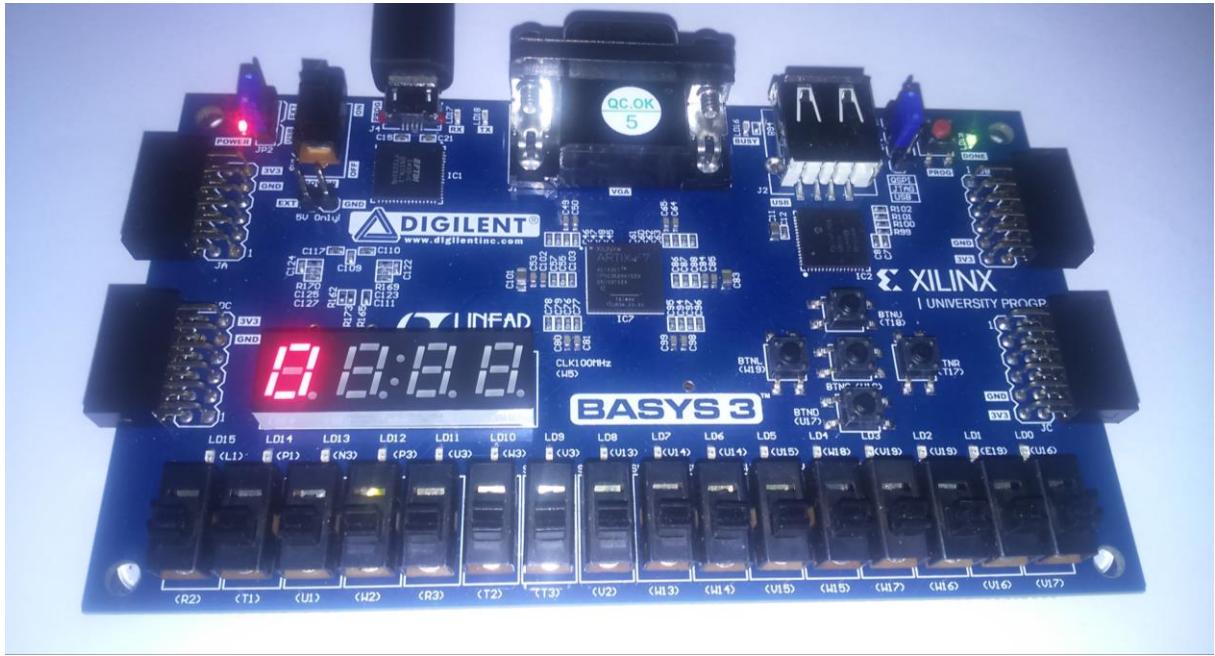


## 5. Manual de utilizare



- De pe primele 8 switch-uri (SW0 – SW7) se introduce valoarea pentru intensitate, dacă este selectat modul manual, sau valoarea pentru minim, maxim și/sau interval, dacă este selectat modul test sau automat. Această valoare va apărea pe primele 3 afișoare.
- SW10 se folosește pentru resetarea plăcuței (este activ pe 1). Este necesară resetarea sistemului de fiecare dată când se dorește modificarea valorilor.
- SW11 reprezintă un ok, care trebuie activat, pentru modul test și automat, doar după ce au fost introduse toate valorile necesare (minimul, maximul și/sau intervalul de timp) pentru a porni sistemul.
- SW12 și SW13 reprezintă un mod pentru introducerea datelor. Ele se utilizează astfel:
  - Pentru valoarea minimă sw12 trebuie să fie setat în starea 1, iar sw13 pe 0
  - Pentru valoarea maximă sw12 trebuie să fie setat în starea 0, iar sw13 pe 1
  - Pentru a introduce intervalul de timp ambele switch-uri trebuie să fie activate
- Ultimele două switch-uri (sw15, sw14) se folosesc pentru selectarea modului, în felul următor:
  - Pentru modul manual sw15 trece în starea 0, iar sw14 în 1
  - Pentru modul test sw15 trece în starea 1, iar sw14 în 0
  - Pentru modul automat ambele switch-uri trebuie să treacă în starea 1

După programarea plăcii, când nu este selectat niciun mod, pe afișorul din stânga apare valoarea 0, iar celelalte afișoare sunt inactive.

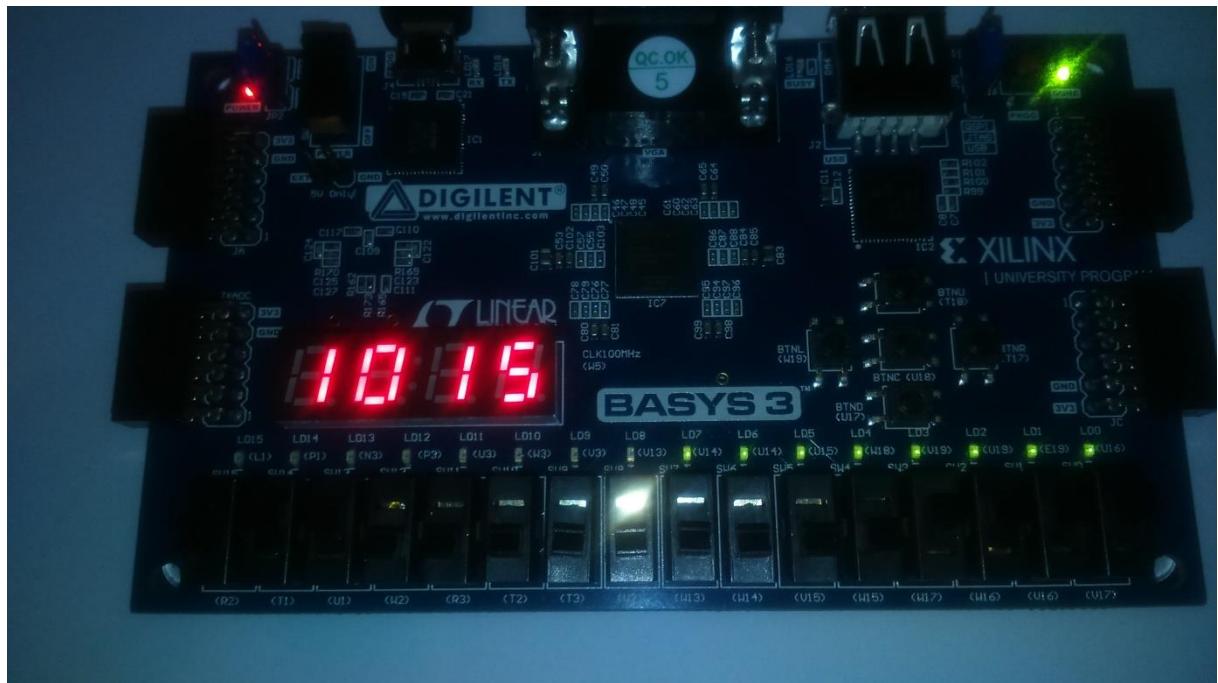


În momentul în care se alege un mod, acesta va apărea pe primul afișor, iar pe celelalte trei afișoare se va indica valoarea introdusă de pe cele 8 switch-uri.

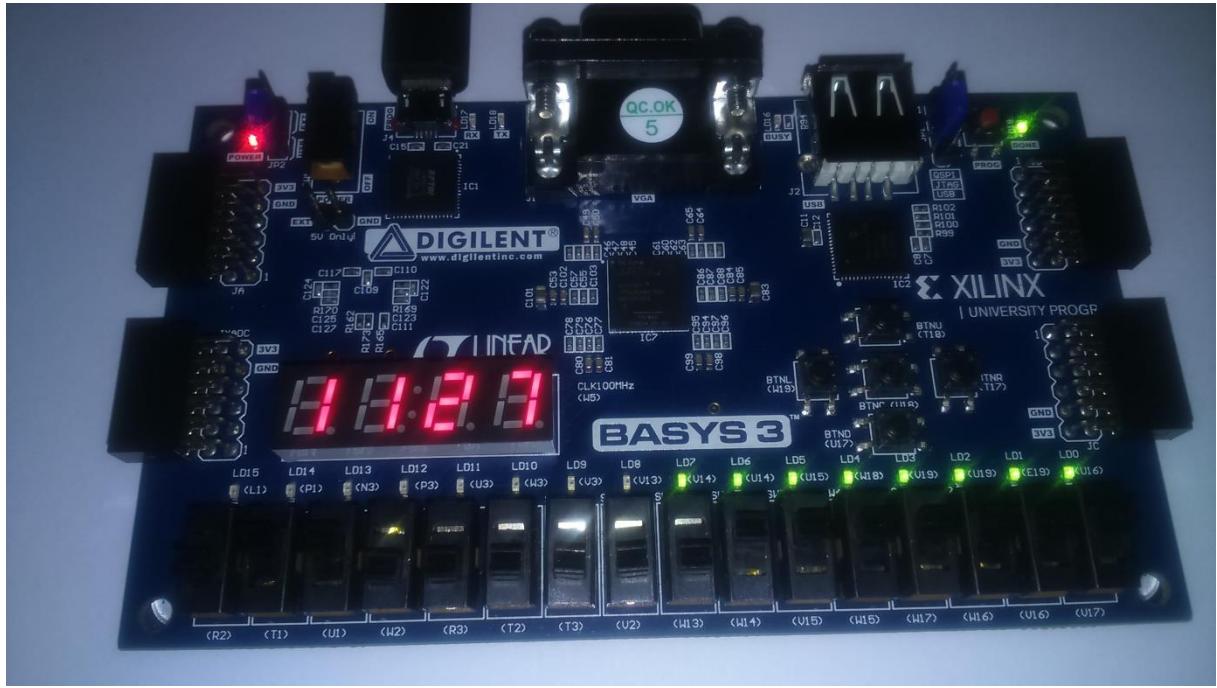
Dacă selectarea modului manual, trebuie să introducem intensitatea luminoasă.

Exemple de funcționare:

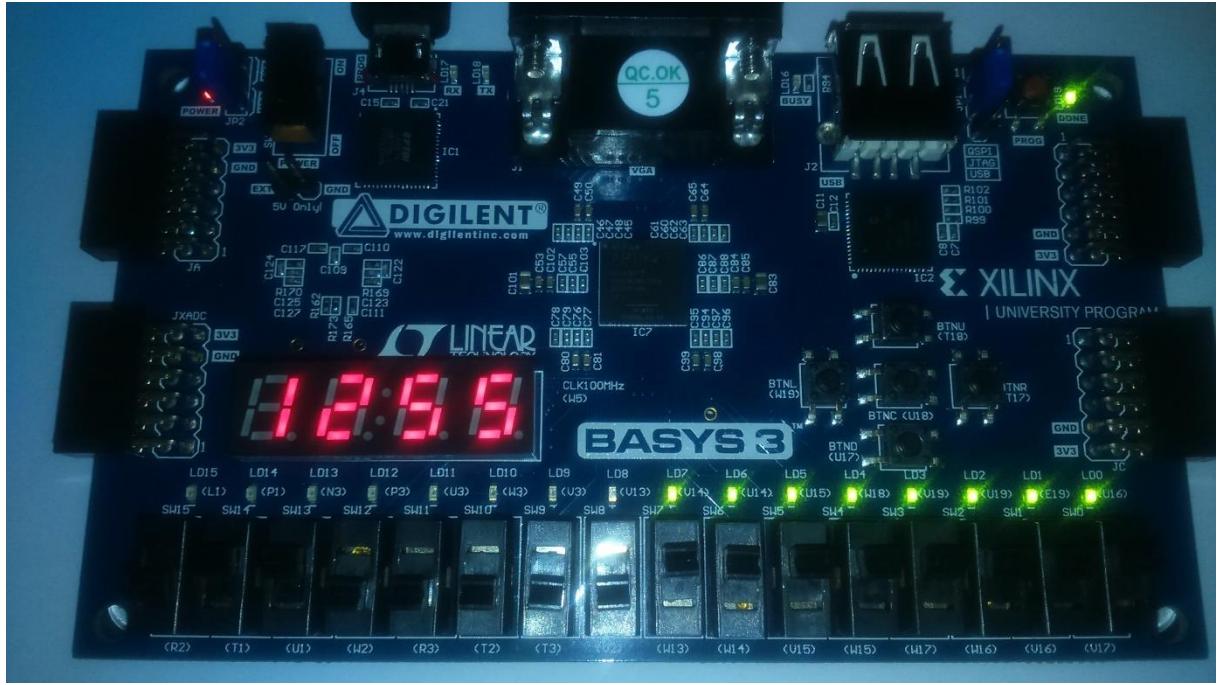
1. Duty = 15, ledurile sunt aprinse la o intensitatea destul de mică



2. Duty = 127

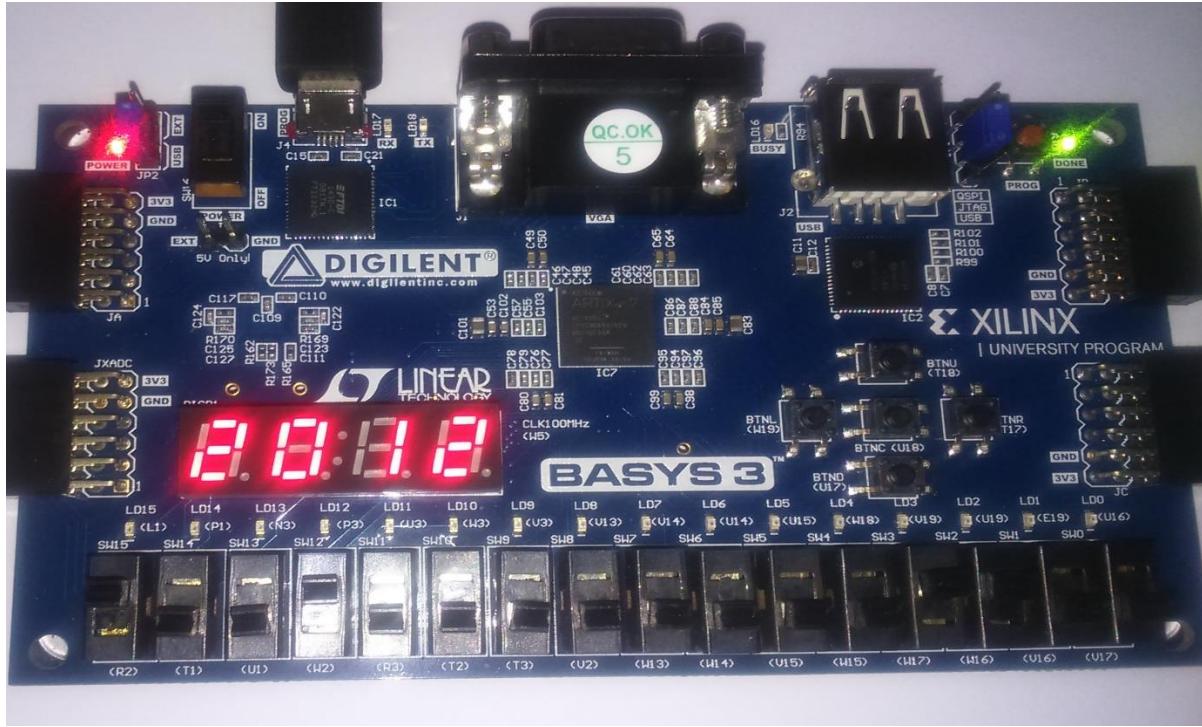


3. Duty = 255 – ledurile sunt aprinse la intensitate maximă

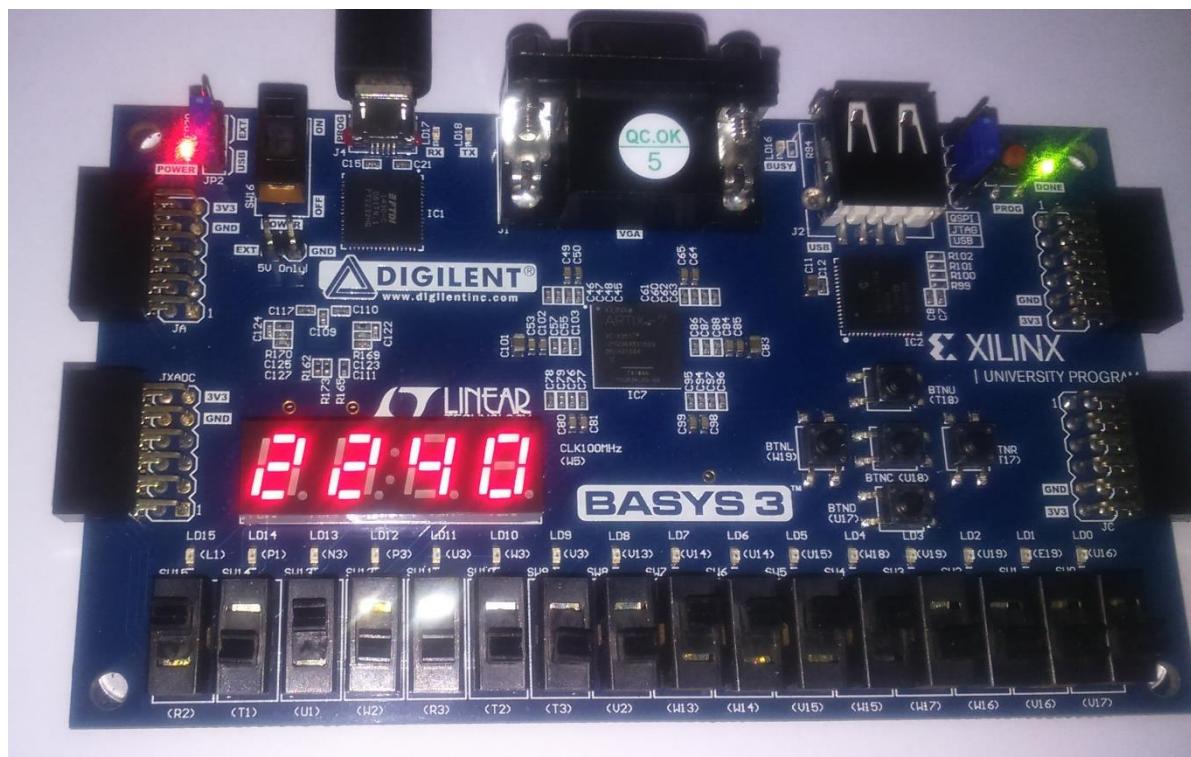


Dacă se selectează modul test, trebuie să introducem valoarea minimă și valoarea maximă a intensității. Acest lucru se realizează în următorul fel:

- Pentru a introduce valoarea minimă, sw12 trece în starea 1 logic, iar sw13 rămâne în starea 0 logic

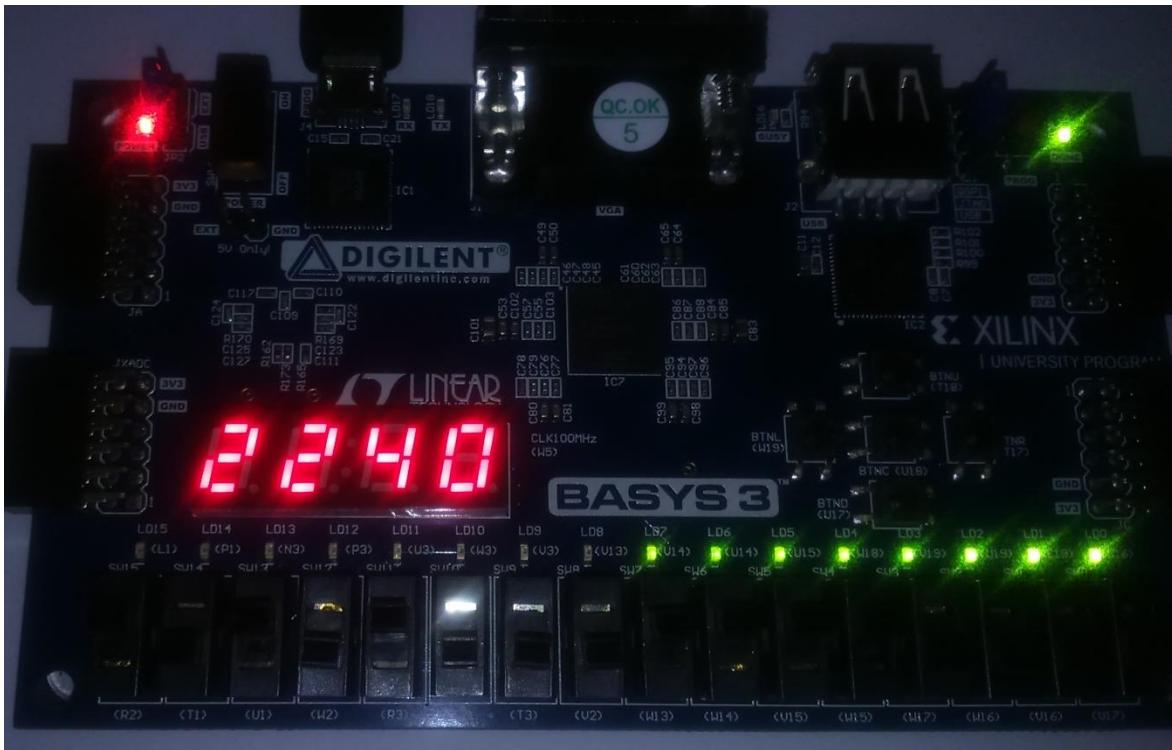


- Pentru a introduce valoarea maximă sw12 se trece în starea 0 logic, iar sw13 trece în 1 logic

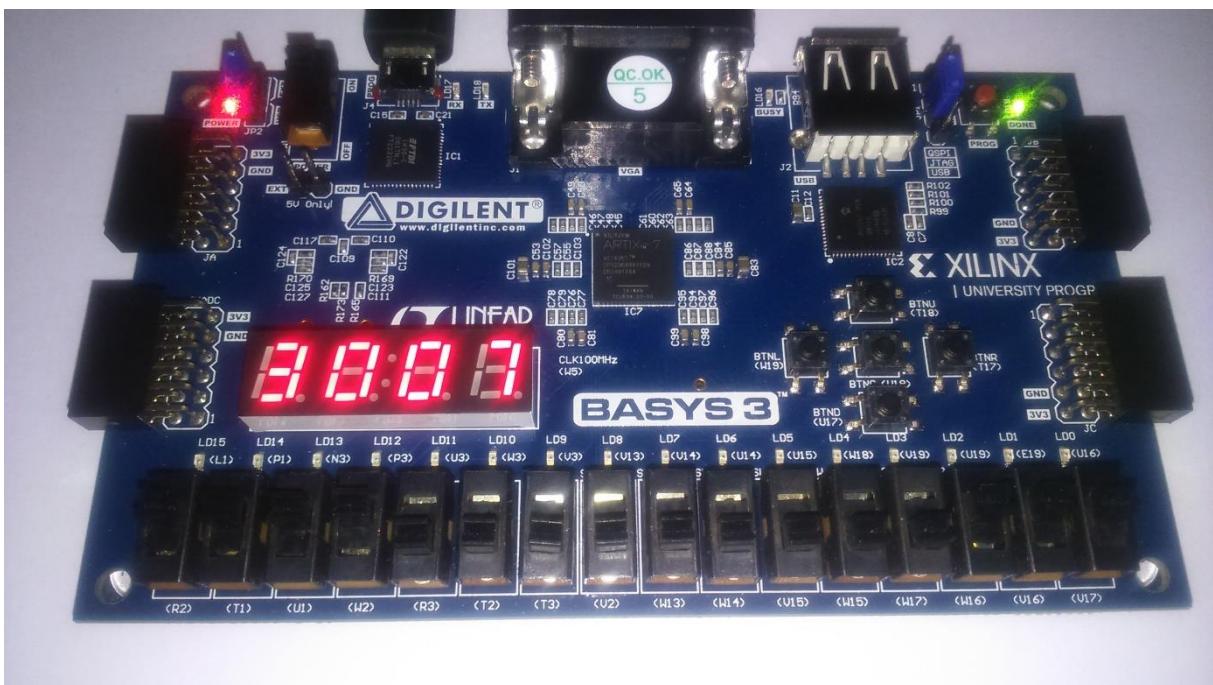


După introducerea valorilor, pentru a porni sistemul este nevoie ca sw11 (ok) să își schimbe starea în 1 logic. Înainte de a se modifica valorile (minim sau maxim) sistemul trebuie resetat.

Exemple de funcționare:

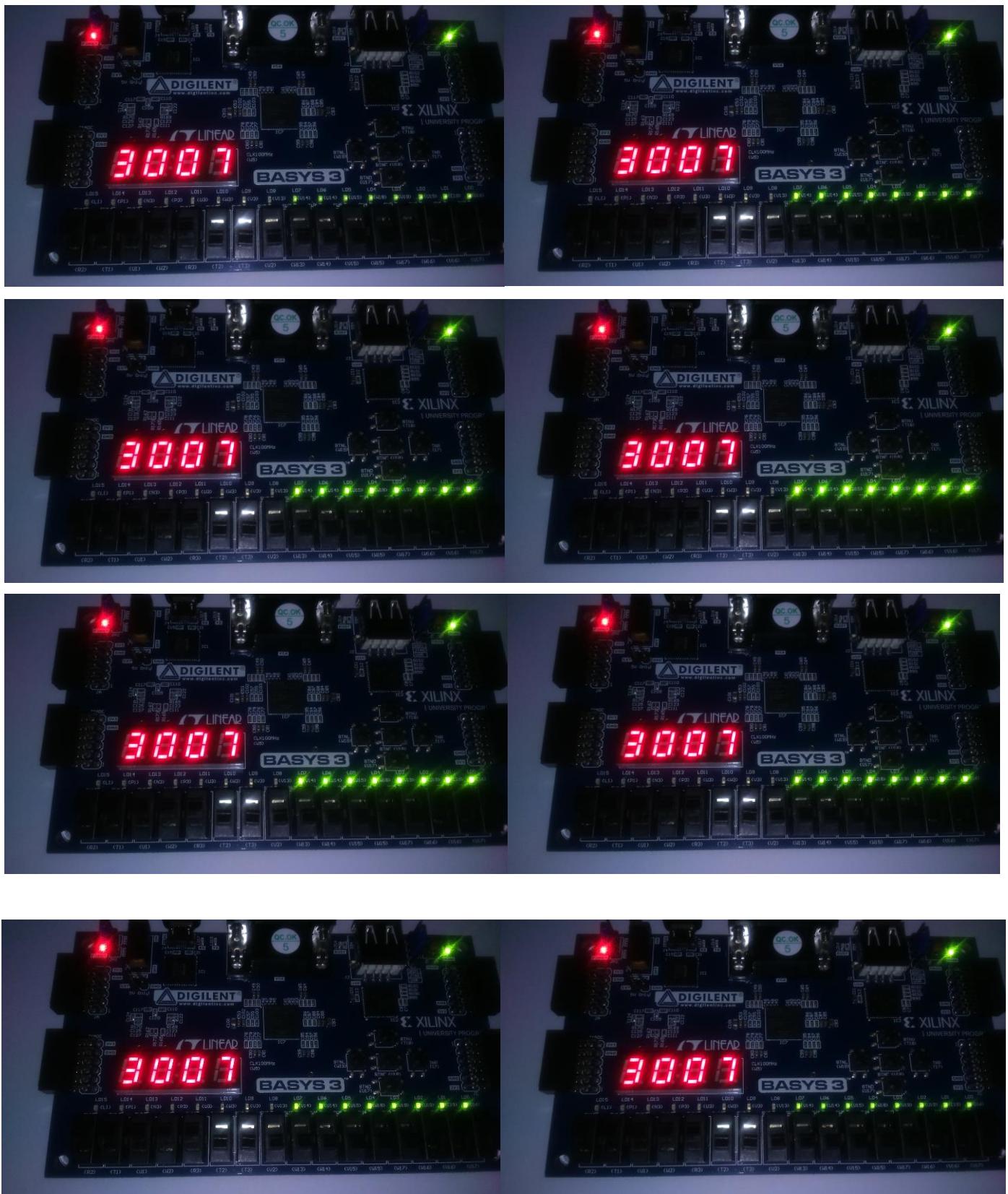


Dacă se selectează modul automat, valoarea minimă și valoarea maximă se introduc la fel ca și în cazul modului test, iar pentru a introduce intervalul de timp este necesar ca sw12 și sw13 sa fie în starea 1:



Pentru a porni sistemul este necesar ca sw11 sa fie în starea 1 logic.

Exemple de funcționare:



## 6. Justificarea soluției alese

---

La realizarea acestui proiect am optat pentru obținerea unui cod cât mai ușor de înțeles de către o altă persoană. În acest scop am utilizat nume semnificative atât pentru intrările și ieșirile sistemului, cât și pentru alte semnale și variabilele din interiorul programului. De asemenea, utilizarea comentariilor pentru a explica anumite secvențe din cod îl face mai ușor de înțeles.

Împărțirea codului în componente face ca acesta să poată fi îmbunătățit, modificat și reutilizat mult mai ușor. De asemenea, am preferat scrierea unui cod indentat corespunzător.

Toate aceste lucruri fac ca soluția aleasă să fie eficientă pentru proiect.

## 7. Posibilități de dezvoltare ulterioară

---

Proiectul dat poate fi dezvoltat prin adăugarea de noi funcționalități ca:

1. Posibilitatea de a alege pentru fiecare led câte un mod de funcționare diferit (de exemplu: led0 – mod manual, led1 – mod test, led2 – mod manual, ..., led7 – mod automat)
2. Posibilitatea de a utiliza sistemul de iluminare variabilă pentru o reclamă luminoasă
3. Adăugarea unui nou mod de funcționare care să regleze intensitatea unui afișor
4. Mărirea domeniului de valori pentru a se vedea mai exact diferențele de intensitatea
5. Posibilitatea de a introduce datele de la o tastatură