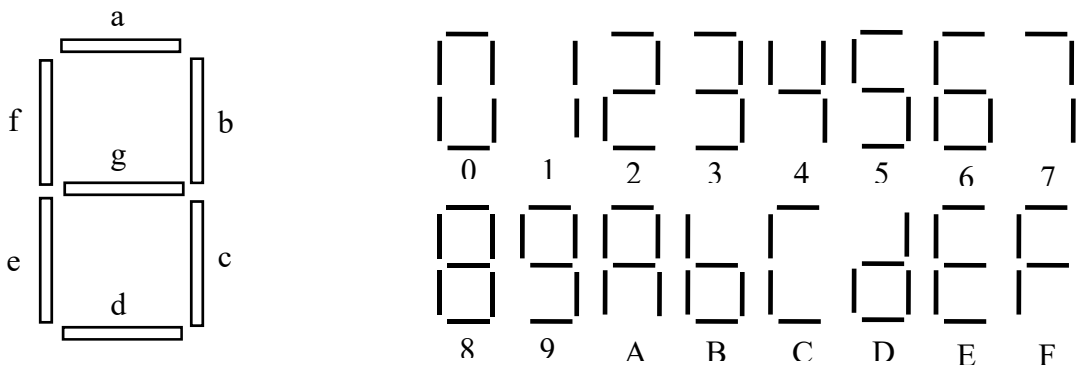
Laboratorul 3 - VHDL

Temă: Să se implementeze *software* și să se realizeze fizic un decodificator binar - 7 segmente, cu utilizarea completă a tuturor codurilor binare de intrare. Se va ține cont că elementul de afișaj este de tipul anod comun (anodurile tuturor elementelor de afișare, a LED-urilor, sunt legate în comun, deci pentru aprinderea unui segment va trebui pus un potențial de 0L pe pinul asociat).

Un afișaj 7 segmente este prezentat mai jos, o dată cu valorile care se doresc afișate:



a. identificarea **b.** corespondența între valoarea afișată și codul hexazecimal

Figura 1. Element de afișaj de tipul digit

Pentru obținerea ecuațiilor între ieșire și intrare ne vom folosi de tabelul de mai jos care a fost scris ținând cont de **Figura 1 (b).** Acest tabel va fi utilizat în realizarea diagramelor VK pentru fiecare segment în parte.

Cod Hexa	Intrări				Ieșiri spre afișaj						
	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
A	1	0	1	0	1	1	1	0	1	1	1
B	1	0	1	1	0	0	1	1	1	1	1
C	1	1	0	0	1	0	0	1	1	1	0
D	1	1	0	1	0	1	1	1	1	0	1
E	1	1	1	0	1	0	0	1	1	1	1
F	1	1	1	1	1	0	0	0	1	1	1

Codul implementării unui decodificator pe 4-biți, BCD : 7 segmente

Codul de mai jos este implementarea în VHDL a circuit combinațional (a decodificatorului) BCD => 7 Segmente. În implementarea acestui circuit decodificator s-a ținut cont doar de codurile 0-9.

```
Entity bcdto7segment_dataflow Is
port(
                x       : in STD_LOGIC_VECTOR(3 downto 0);
                an      : out STD_LOGIC_VECTOR(3 downto 0);
                seg     : out STD_LOGIC_VECTOR(6 downto 0)
        );
end bcdto7segment_dataflow;

Architecture behavior of bcdto7segment_dataflow Is

        Signal m_int : STD_LOGIC;
        Signal u1_o : STD_LOGIC;

begin
        an(0) <= '0';
        an(1) <= '1';
        an(2) <= '1';
        an(3) <= '1';
```

```
	seg(0) <= (x(2) and not(x(1)) and not(x(0))) or (not(x(3)) and not(x(2)) and not(x(1)) and
x(0));
	seg(1) <= (x(2) and not(x(1)) and x(0)) or (x(2) and x(1) and not(x(0)));
	seg(2) <= (not(x(2)) and x(1) and not(x(0)));
	seg(3) <= (x(2) and not(x(1)) and not(x(0))) or (x(2) and x(1) and x(0)) or (not(x(3)) and
not(x(2)) and not(x(1)) and x(0));
	seg(4) <= x(0) or (x(2) and not(x(1)));
	seg(5) <= (not(x(3)) and not(x(2)) and x(0)) or (x(1) and x(0)) or (not(x(3)) and not(x(2))
and x(1));
	seg(6) <= (not(x(3)) and not(x(2)) and not(x(1))) or (x(2) and x(1) and x(0));

end behavior;
```

Modalități de reprezentare a numerelor

Limbajul de modelare hardware VHDL permite reprezentarea numerelor în mai multe baze de numerotație.

VHDL-ul definește în principal 2 tipuri de date: de tip bit (**STD_LOGIC**) și de tip vectoriale (**STD_LOGIC_VECTOR** mai mulți biți grupați împreună). Un semnal poate avea una din următoarele 4 valori de bază:

1. 0: cu semnificația de nivel logic 0 sau fals;
2. 1: cu semnificația de nivel logic 1 sau adevărat (true);
3. x: necunoscut;
4. z: înaltă impedanță;

Limbaul VHDL nu este "case sensitive". Există trei tipuri de date pe care o constantă le poate lua:

1. intreg,
2. real, și
3. de tip string.

Numerele întregi pot fi scrise în următoarele două moduri:

1. simplu zecimal sau
2. într-un format în care se precizează baza de reprezentare (octală, hexazecimală, etc.).

O valoare întreagă în format simplu zecimal este reprezentată ca o secvență d digiți precedată de semn "+" sau "-". Semnul este opțional, putând fi introdus doar dacă este cazul (de ex. în cazul numerelor negative). De exemplu: "15", "+15" sau "-32".

Pentru exemplul anterior:

1. "15" este reprezentat în binar prin "01111", în cazul unei reprezentări binare pe 5 biți;
2. În timp ce "-32" este reprezentat în binar prin "100000" dacă reprezentarea este pe 6-biți.

Un număr reprezentat în format bază are, în VHDL, următoarea sintaxă de descriere:

<bază> <valore >

În reprezentarea anterioară baza poate lua una din următoarele valori o sau O (pentru octal), b sau B (pentru binar) și x sau X (pentru hexazecimal). Valoare reprezintă orice secvență de caractere validă bazei respective. Valorile binare se definesc între ghilimele fără a specifica baza. Valoarea trebuie să fie fără semn. De exemplu:

1. o“37” reprezentare în octal pe 5 biți;
1. “1111” reprezentare pe 4 biți binară;
2. x“AA” reprezentare hexazecimală pe 8-biți.

Dacă variabila care va fi inițializată are mai mulți biți decât valoarea care i se atribuie, atunci trebuie concatenați biți suplimentari pentru egalarea numărului de biți a celor 2 reprezentări. De exemplu:

```
--un vector pe 8 bi•i se declar•
signal counter : std_logic_vector(7 downto 0);

--mai jos exemplu atunci când se dore•te atât declararea cât •i ini•ializarea
signal counter2: std_logic_vector(7 downto 0) := ''01101001'';

...

-- Hexadecimal ''A'' este ''1010'', semnalul counter este reprezentat pe 8 bi•i
-- deci 4 bi•i de valoare zero trebuie concatena•i
counter <= ''0000'' & x''A'';
```

Implementați în VHDL un program ierarhic care să utilizeze modulul BCD:7 segmente și în care o valoare pe 4 biți stocată ca semnal să fie afișată pe elementul de afișare. Codul implementării este prezentat mai jos:

```
entity up_level is
   Port (  an_up   : out STD_LOGIC_VECTOR (3 downto 0);
           seg_up  : out STD_LOGIC_VECTOR (6 downto 0));
end up_level;

architecture Behavioral of up_level is

        Component bcdto7segment_dataflow
            port (
                x   : in  STD_LOGIC_VECTOR(3 downto 0);
                an  : out STD_LOGIC_VECTOR(3 downto 0);
                seg : out STD_LOGIC_VECTOR(6 downto 0)
            );
        End Component;

        Signal x_int : STD_LOGIC_VECTOR (3 downto 0) := "0101";

begin
```

```
        U1:  bcdto7segment_dataflow PORT MAP (
                        x => x_int,
                        an  => an_up,
                        seg => seg_up
                        );
end Behavioral;
```