LABORATOR 2

Proiectarea circuitelor de codificare/decodificare
Proiectarea circuitelor cu ieşiri three-state
Proiectarea multiplexoarelor/demultiplexoarelor

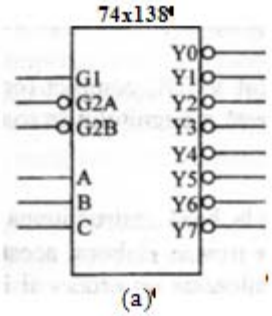
1 Proiectarea circuitelor de codificare/decodificare

A. Decodificatoare

Primul model pe care îl propunem este cel al unui decodificator binar 3-la-8, de tip 74x138. Schema bloc şi tabelul de adevăr al acestui circuit digital sunt prezentate în Figura .1. în notația folosită, intrările şi ieşirile care au un cerculeț indică faptul că sunt active L.
Aşadar, 74x138 are trei intrări de validare: G1 - activă H, G2A şi G2B - active L, trei intrări de selecție: A, B, C. Circuitul are opt ieşiri active L.
Din tabelul de adevăr se pot obține ecuațiile logice (minimizate) pentru cele opt ieşiri. De exemplu:

$$\overline{Y5} = \underbrace{G1 \cdot \overline{G2A\_L} \cdot \overline{G2B\_L}}_{\text{validare}} \cdot \underbrace{C \cdot \overline{B} \cdot A}_{\text{selecție}}$$

Primul model al circuitului 74x138 care trebuie dezvoltat urmăreşte direct tabelul de adevăr. Declarația de entitate şi arhitectura trebuie să se elaboreze completând fragmentul din Figura 2. Intrările de adresă, `A (2 downto 0)` şi ieşirile decodificate, active L, `Y_L (0 to 7)`, sunt declarate ca vectori; intrările de validare sunt declarate ca semnale individuale.
În corpul arhitecturii se declară un semnal intern, `Y_L_i,` care este o copie a ieşirii. Valoarea care trebuie să se obțină la ieşire se va obține mai întâi la `Y_L_i,` după care se va asigna ieşirii `Y_L.` Descrierea comportării decodificatorului are la bază o singură instrucțiune `select` care enumera cele opt cazuri posibile şi asignează şablonul corespunzător de ieşire semnalului intern `Y_L_i.` Această valoare se transferă la ieşirea `Y_L` numai dacă toate intrările de validare sunt active.



Intrări						Ieşiri							
G1	*G2A*	*G2B*	*C*	*B*	*A*	*Y7*	*Y6*	*Y5*	*Y4*	*Y3*	*Y2*	*Y1*	*Y0*
0	–	–	–	–	–	1	1	1	1	1	1	1	1
–	1	–	–	–	–	1	1	1	1	1	1	1	1
–	–	1	–	–	–	1	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1	0
1	0	0	0	0	1	1	1	1	1	1	1	0	1
1	0	0	0	1	0	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	0	1	1	1
1	0	0	1	0	0	1	1	1	0	1	1	1	1
1	0	0	1	0	1	1	1	0	1	1	1	1	1
1	0	0	1	1	0	1	0	1	1	1	1	1	1
1	0	0	1	1	1	0	1	1	1	1	1	1	1

(b)

Fig. .1 Decodificatorul 74x138. a) schema bloc; b) tabelul de adevăr.

```
entity circuit_74xl38
  port  (Gl, ..........       — intrări de validare
    A:  in                        (2  downto  0);  --  intrări  de  selecţie
    Y_L: out .... (0 to 7));      — ieşiri decodate
end entity circuit_74xl38;

architecture varianta_l of circuit_74xl38 is
  signal Y_L_i:    .......... — semnal copie a ieşirii
begin
  with A select Y_L_i <=
"01111111" when ........ , — opt cazuri posibile

     .......... '
     ......  when others;

Y_L <= Y_L_i when  (Gl and not ... )  = '1' else  __;
end architecture varianta 1;
```

Fig. 2 Modelul VHDL incomplet al decodificatorului 74x138 (instrucţiunea **select).**

Completaţi programul din Figura 2, compilaţi-1 şi simulaţi funcţionarea circuitului. Utilizaţi facilitatea *Language Templates* din ISE Xilinx.

Elaboraţi un raport scris care să conţină modelul VHDL complet (comentat) şi formele de undă care demonstrează funcţionarea corectă a circuitului în toate cazurile posibile.

A doua variantă de model pentru decodificatorul 74x138 este o arhitectură cu adevărat comportamentală. Modelul complet este dat în Figura 3

1.Completaţi şi compilaţi programul din Figura 3 şi simulaţi funcţionarea circuitului.

2. Elaboraţi un raport scris care să conţină modelul VHDL complet (comentat) şi formele de undă care demonstrează funcţionarea corectă a circuitului în toate cazurile posibile.

3. De ce condiţia din instrucţiunea `if` este (G1 and G2 and G3) = '1'? Este greşită sau nu? De ce?

4. Care este rolul funcției CONV INTEGER?

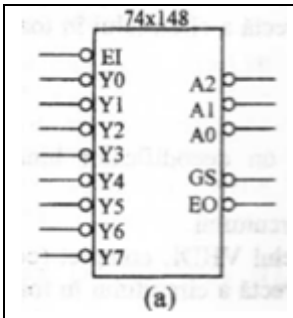
```
architecture varianta_3 of circuit_74x138 is begin
process  (A,  Gl,  G2,  G3)
  variable  i:  integer  range  0  to  7;  begin  Y  <=
    "00000000";
    if  (Gl and G2 and G3) =        then for i in
      0 to 7 loop
  if i = CONV_INTEGER  (A)  then Y(i)  <=  '1'; end if; end loop; end
  if; end process;

end architecture varianta 3;
```

Fig. 3 Modelul comportamental complet al decodificatorului 74x138

B Codificatoare

Circuitul propus pentru modelare şi simulare este codificatorul de priorități 74x148. Schema bloc şi tabelul de adevăr pentru acest circuit sunt prezentate în Figura 4. Intrările şi ieşirile sunt active L. Circuitul are o intrare de validare, EI, opt intrări de date, YO - Y7, trei ieşiri de date, A2, Al, AO, o ieşire ***group select*** GS care se activează atunci când dispozitivul este validat şi se aplică una sau mai multe cereri la intrare, şi o ieşire ***enable output*** EO care se foloseşte pentru cascadarea codificatoarelor (se conectează la intrarea EI a dispozitivului următor). Funcționarea este următoarea: La ieşire se obține codul binar (complementat) al intrării active (zero logic). Dacă sunt active mai multe cereri la intrare, la ieşire se obține codul binar (complementat) al intrării cu prioritatea cea mai mare.



Intrări									Ieşiri				
EI	*I0*	*I1*	*I2*	*I3*	*I4*	*I5*	*I6*	*I7*	*A2*	*A1*	*A0*	*GS*	*EO*
1	–	–	–	–	–	–	–	–	1	1	1	1	1
0	–	–	–	–	–	–	–	0	0	0	0	0	1
0	–		–	–	–	–	0	1	0	0	1	0	1
0	–		–	–	–	0	1	1	0	1	0	0	1
0	–	–	–	–	0	1	1	1	0	1	1	0	1
0	–	–	–	0	0	1	1	1	1	0	0	0	1
0	–	–	0	1	1	1	1	1	1	0	1	0	1
0	–	0	1	0	0	1	1	1	1	1	0	0	1
0	0	1	1	0	1	1	1	0	1	1	1	0	1
0	1	1	1	1	0	1	0	1	1	1	1	1	0

Fig. 4 Codificatorul de priorități 74x148. a) schema bloc; b) tabelul de adevăr

Prima variantă de model VHDL se poate obține folosind construcția `IF-THEN-ELSE`, care este accesibilă numai în interiorul proceselor. Declarația de entitate şi arhitectura trebuie să se elaboreze completând fragmentul din Figura 5.

Programul foloseşte o buclă `FOR` pentru a căuta o intrare activă, începând cu intrarea cu prioritatea cea mai mare. Se realizează şi conversia explicită a nivelurilor active la începutul şi la sfârşitul programului. Se foloseşte funcția `CONV_STD_LOGIC_VECTOR (j`

, n) pentru a realiza conversia de la un întreg, j, la un STD_LOGIC_VECTOR cu dimensiunea specificată, n. Programul se poate modifica uşor pentru a schimba ordinea priorităților sau pentru a modifica numărul de intrări.

```
    entity circuit_74xl38 is
        Port  ( EI_L :  in    STD_LOGIC;
                I_L :  in    STD_LOGIC_VECTOR  (7 downto 0);
                EO_L,  GS_L :  out    STDJLOGIC;
                A_L  :    out       STD_LOGIC_VECTOR    (2  downto  0));  end
    circuit_74xl38;

    architecture Behavioral of circuit_74xl38 is
      signal EI: ...— versiuni active H ale intrărilor
      signal I: ...
      signal EO, GS:  ....— si ale ieşirilor
      signal A:   ..
process  (EI_L,  I_L,  EI,  EO,  GS,  I, A) variable j:
integer range 7 downto 0; begin
   EI <= .... — conversia intrării
   I <= .... -- conversia intrărilor;
   — valori inițiale
   EO <= ....;
   GS <= ---------------;
   A <= ----------------;

   if   (EI)  = ’0’  then EO <= ' 0', ;

   else for j in 7 downto 0 loop

         if I(j)  = '1'  then
              GS <=....; EO <= ....; — prima linie din tabelul de adevăr

         A <= CONV_STD_LOGIC_VECTOR  (.".'.....);
         exit;
         end if;
         end loop;
   end if;
   EO_L <= ...; — conversia ieşirii
   GS_L <= ....; — conversia iesirii
   A_L <= ....; — conversia ieşirilor
end process;
```

Fig. 5 Modelul VHDL incomplet al codificatorului de priorități 74x148.

1. Completați programul din Figura 5 compilați-1 şi simulați funcționara circuitului. Utilizați facilitatea *Language Templates* din ISE Xilinx.
2. Elaborați un raport scris care să conțină modelul VHDL complet (comentat) şi formele de undă care demonstrează funcționarea corectă a circuitului în toate cazurile posibile.

2. Proiectarea circuitelor cu ieşiri three-state

VHDL nu are tipuri şi operatori pentru ieşiri cu trei stări. Are totuşi primitive care pot fi folosite pentru a forma semnale şi sisteme cu ieşiri three-state; pachetul IEEE 1164 foloseşte aceste primitive. **IEEE.STD_L0GIC_1164** defineşte '**Z'** ca una dintre cele nouă valori posibile ale semnalelor; această valoare este folosită pentru starea de înaltă impedanţă. Se poate asigna această valoare oricărui semnai **STD_LOGIC** iar definiţiile funcţiilor standard logic se aplică şi pentru intrări cu valoarea '**Z'**(în general, o intrare '**Z'** va determina o ieşire **'U').**

Prima variantă a circuitului propus în acest paragraf conţine patru drivere (buffer three-state de opt biţi fiecare. Modelul va fi dezvoltat în patru procese prin care se selectează unul din cele patru bus-uri de opt biţi **A, B, C** sau **D** pentru a se transmite la ieşirea **X** (Figura 6). Declaraţia de entitate şi arhitectura trebuie să se elaboreze completând fragmentul din Figura 7.

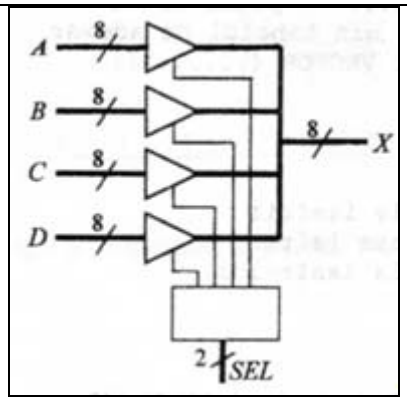


Fig. 6 Selecţia unui bus de 8 biţi dintr-un grup de patru

```
entity circuit_3_state is
     port  (
           G_L:       ; — semnal global de enable
           SEL:  …. intrări de selecţie
           A,B,C, D: 4 bus-uri de 8 biţi
           X:  ….); - bus de ieşire cu 3 stări;
       end entity;
      architecture behavior of circuit_3_state is begin
          process   (G_L,  SEL, A, B, ) begin
             if G_L =    and    SEL =  then X <= A;
             elsif G_L =    and SEL =   then X <=B;
                ……..
              else   X <=  (others => "ZZZZZZZZ") ;
             end if;
          end process;
          end architecture;
```

Fig. 7 Programul VHDL incomplet pentru un circuit care conţine patru Buffere de tip three-state de opt biţi.

1. Completați programul din Figura 8, compilați-1 şi simulați funcționarea circuitului. Utilizați facilitatea *Language Templates* din ISE Xilinx.
2. Elaborați un raport scris care să conțină modelul VHDL complet (comentat) şi formele de undă care demonstrează funcționarea corectă a circuitului în toate cazurile posibile.

3. Multiplexoare

Un multiplexor este un dispozitiv digital care conectează una dintre cele ***n*** surse de intrare la unica ieşire, în funcție de valoarea semnalului de selecție. Schema bloc generală a unui multiplexor este prezentată în Figura 8. Sunt ***n*** surse de date, fiecare de câte ***b*** biți; există o ieşire de ***b*** biți. Variantele comerciale tipice au ***n*** = 1,2,4,8,16 iar ***b*** = 1,2,4,8. Există ***s*** intrări cu ajutorul cărora se selectează una dintre cele ***n*** surse, aşa încât ***s*** = $[\log_2 n]$. În anumite variante există şi o intrare generală de validare **EN.**

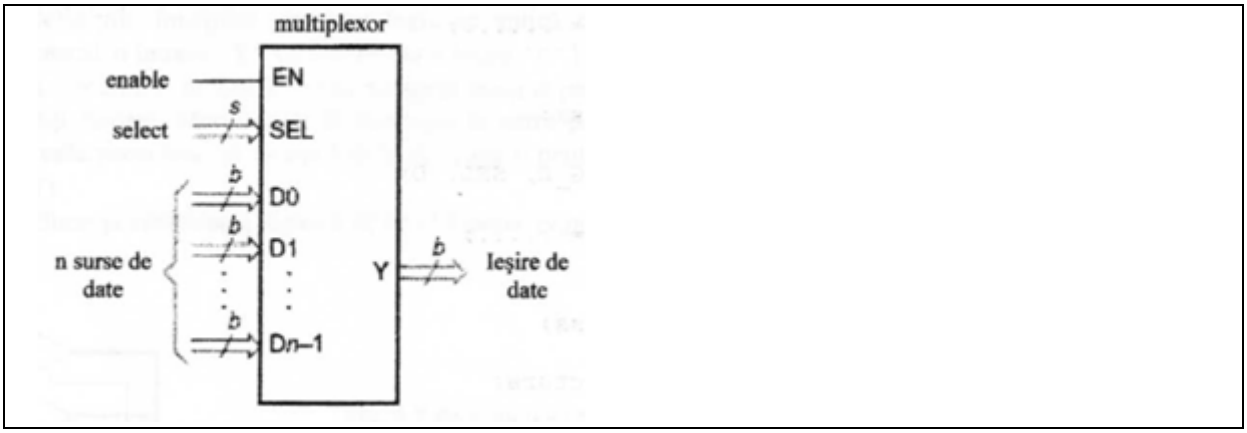


Fig. 8 Schema bloc a unui multiplexor.

Primul model care va trebui elaborat este acela al unui multiplexor cu patru intrări, de câte opt biți fiecare. **Modelul, de tip flux de date (**sau *data flow***)**, se bazează pe o instrucțiune **select** şi este prezentat în Figura 9.

1. Completați programul din Figura 9, compilați-1 şi simulați funcționarea circuitului. Utilizați facilitatea *Language Templates* din ISE Xilinx.
2. Elaborați un raport scris care să conțină modelul VHDL complet (comentat) şi formele de undă (adnotate) care demonstrează funcționarea corectă a circuitului în toate cazurile posibile.

```
entity mux_4tol
port  (
  S:.....; – intrare de selecție,  0 - 3  ==> A - D
  A, B, C,  D:   .. ; – patru intrări
                            (bus) de date de cate 8 biți
```

```
      Y:    ... – ieşirea  (bus)  8 biţi
      ) ;
  end entity mux_4tol;
■
  architecture mux_4tol of mux_4tol is begin
    with S select Y <=
      A when ......, – cele 4 variante posibile
      B when  .....
      C when  ....
      D when  .....
       (others => 'U') when others;       formează vector de 8 biţi
                                        cu componente 'U'
end architecture mux_4tol;
```

Fîg. 9 Modelul VHDL incomplet al unui multiplexor 4:1 de opt biți (instrucțiunea **select)**

A doua variantă poate fi un model comportamental care foloseşte instrucțiunea **case.** Utilizați facilitatea *Language Templates* din ISE Xilinx pentru a descrie o astfel de variantă de multiplexor