Proiectare de tip flux de date

În limbajul VHDL există un şir de instrucţiuni ***concurente*** care permit descrierea circuitului din punctul de vedere al fluxului de date şi al prelucrării acestuia în circuit.

1. Instrucţiunea de atribuire concurentă directă de semnal
(*signal-name<=expression*).

Tipul expresiei trebuie să fie compatibil cu cel al semnalului.

Pentru a exemplifica atribuirea concurentă de semnal arhitectura detectorului de numere prime este rescrisă pentru fluxul de date.

```
architecture   prime2_arch of prime is
        signal T1, T2, T3, T4  :  STD_LOGIC;
                begin
                        T1  <=  not N(3) and N(0) ;
                        T2  <= not N(3) and not  N(2) and N(1) ;
                        T3  <= not N(2) and N(1) and N(0) ;
                        T4  <= N(2) and not N(1) and N(0) ;
                        F    <= T1 or T2 or T3 or T4;
                end prime2_arch;
```

Spre deosebire de descrierea anterioară (descriere structurală) porţile şi conexiunile dintre ele nu mai apar explicit, ci se folosesc operatori *VHDL* definiţi implicit in biblioteca IEEE (*and, or, not*). Se observă că operatorul *not* are prioritatea cea mai mare, astfel că nu este necesar să se utilizeze paranteze pentru expresii de genul *"not N(2)"*.

2. Instrucţiunea de atribuire concurentă condiţională de semnal when - else.

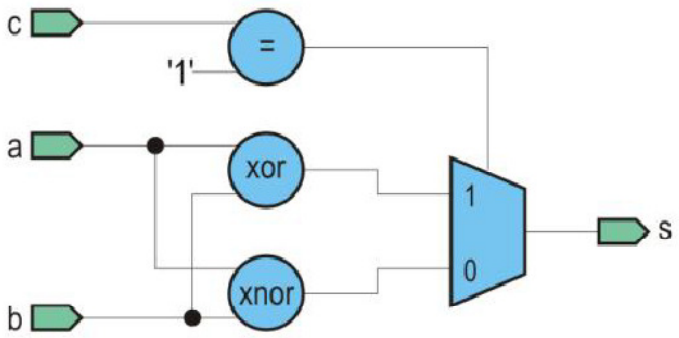
Nume-semnal <= *expresie* ***when*** *espresie-booleană* ***else***
. . .
expresie ***when*** *espresie-booleană* ***else***
expresie;

In acest caz o expresie booleană combină termeni booleeni individuali folosind operatori impliciţi (*and, or, not*). Termenii booleeni sunt de obicei variabile sau rezultate ale unor comparaţii făcute cu ajutorul operatorilor relaţionali: = , /= (*inegaliate*), > , >= , < , <= (*mai mic sau egal*)

Instrucţiunea de atribuire condiţională este implementată printr-un multiplexor care selectează una din expresiile sursă.

Circuitul rezultat pentru următoarea instrucţiune:

```
s <= a xor b when c = '1' else
not (a xor b);
```



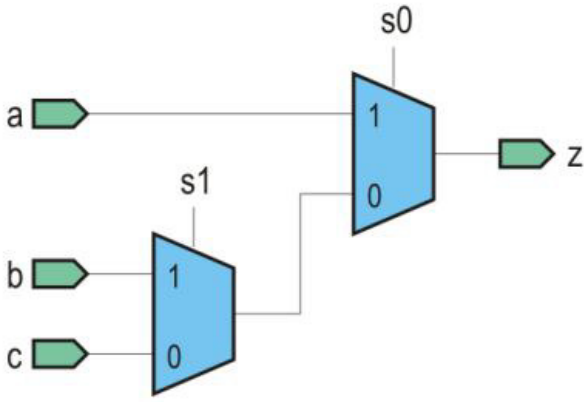
Circuitul este cel generat iniţial de utilitarul de sinteză, dar operatorul de egalitate va fi minimizat ulterior la o simplă conexiune, astfel încât semnalul c să controleze multiplexorul în mod direct.

Exemplul anterior reprezintă forma cea mai simplă a unei instrucţiuni de atribuire condiţională, în care există o singură condiţie testată. Un alt exemplu în care există două condiţii testate este următorul:

```
z <= a when s0 = '1' else
b when s1 = '1' else
c;
```

Condiţiile sunt evaluate în ordinea în care sunt scrise, fiind selectată pentru atribuire prima expresie a cărei condiţie este adevărată. Aceasta echivalează din punct de vedere hardware cu o serie de multiplexoare cu două căi, prima condiţie controlând multiplexorul cel mai apropiat de ieşire.

Circuitul rezultat pentru acest exemplu:



Pentru acest circuit, condiţiile au fost deja optimizate, astfel încât semnalele s0 şi s1 controlează multiplexoarele în mod direct. Din acest circuit se poate

observa că atunci când ***s0*** este '1', este selectat semnalul ***a*** indiferent de valoarea semnalului ***s1***. Dacă ***s0*** este '0', atunci ***s1*** selectează între intrările ***b*** şi ***c.***

Dacă există un număr mare de ramuri ale instrucţiunii, la implementare rezultă un şir lung de multiplexoare. De acest aspect trebuie să se ţină cont la proiectare: cu cât o expresie sursă apare mai târziu în lista de selecţie, cu atât semnalele din această expresie vor traversa mai multe multiplexoare în circuitul rezultat la sinteză.

Exemplu. Arhitectura detectorului de numere prime in care se foloseşte atribuirea concurentă condiţională de semnal.

```
architecture   prime3_arch of prime is
signal T1, T2, T3, T4  :  STD_LOGIC;
    begin
T1  <=  '1' when N(3)='0' and N(0)='1' else '0' ;
T2  <=  '1' when N(3)='0' and N(2)='0' and N(1)='1' else '0' ;
T3  <=  '1' when N(2)='0' and N(1)='1' and N(0)='1' else '0' ;
T4  <=  '1' when N(2)='1' and N(1)='0' and N(0)='1' else '0' ;
F    <= T1 or T2 or T3 or T4;
 end prime3_arch;
```

Comparaţia unui bit de tip *std_logic* cum ar fi *N(3)* se face in funcţie de caracterele literale *'1'* sau *'0'*, iar rezultatul returnat este de tip boolean.

Rezultatul acestor comparaţii se combină intr-o expresie booleană plasată intre cuvintele cheie *when, else.* Clauza *else* este obligatorie deoarece setul de condiţii dintr-o expresie trebuie să acopere toate combinaţiile posibile.

3. ***Instrucţiunea de atribuire concurentă selectivă de semnal:***

```
with  expresie  select
nume-semnal <= valoare-semnal  when  opţiuni,
                        valoare-semnal  when  opţiuni,
                  . . .
                        valoare-semnal  when  opţiuni;
```

In acest tip de instrucţiune, se evaluează expresia dată, iar cand una dintre valori se potriveşte cu una dintre opţiuni (*choices*) atunci identificatorului *nume-semnal* i se va atribui valoarea corespunzătoare *valoare-semnal.* Opţiunea corespunzătoare fiecărei clauze *when* poate să fie o singură valoare sau o listă de valori separate intre ele prin bara verticală (|). Cuvantul cheie *others* poate fi

folosit impreună cu ultima clauză *when* pentru a acoperii toate valorile pe care le poate lua expresia de evaluare.

Această instrucţiune este echivalentă funcţional cu instrucţiunea secvenţială ***case.***

Exemplu. Arhitectura detectorului de numere prime in care se foloseşte atribuirea concurentă selectivă de semnal. Toate opţiunile pentru care *F* ia valoarea '*1*' ar fi putut fi scrise folosind o singură clauză *when,* dar pentru o inţelegere mai bună s-au folosit mai multe clauze.

```
architecture   prime4_arch of prime is
                begin
                      with  N select
                            F  <=  `1`  when "0001" |  "0010" | "0011",
                                    `1`  when  | "0101"  | "0111" ,
                                   `1`  when "1011"  | "1101"  ,
                                   `0`  when others ;
                end prime4_arch;
```

Exemple.

1. Poarta SAU EXCLUSIV cu două intrări. În arhitectură se utilizează o instrucţiune de atribuire condiţională.

```
library ieee;
use ieee.std_logic_1164.all;
entity xor2 is
   port (a, b: in std_logic;
        x: out std_logic);
end xor2;
architecture arh1_xor2 of xor2 is
  begin
     x <= '0' when a = b else
           '1';
  end arh1_xor2;
```

2. Definirea porţii SAU EXCLUSIV cu două intrări, în cadrul arhitecturii se utilizează o instrucţiune de atribuire selectivă.

```
library ieee;
use ieee.std_logic_1164.all;
  entity xor2 is
     port (a, b: in std_logic;
```

```
            x: out std_logic);
  end xor2;
architecture arh_xor2 of xor2 is
    signal tmp: std_logic_vector (1 downto 0);
      begin
        tmp <= a & b;
        with tmp select
        x <= '0' when "00",
        '1' when "01",
        '1' when "10",
        '0' when "11";
end arh_xor2;
```