# Lucrare de laborator nr. 8
# Modelarea data-flow
# şi modelarea structurală în VHDL

## 1. Scopul lucrării

Însuşirea principiilor privind modelarea data-flow şi modelarea structurală în VHDL. Însuşirea noţiunilor privind instrucţiunea de atribuire pentru semnale, specificarea întârzierilor, întârzierea Δ, declararea unei componente, instrucţiunea de instanţiere de componente. Sunt exemplificate modelarea data-flow a unui bistabil D şi modelarea structurală a unui latch pe 8 biţi. Pentru fiecare model sunt implementate arhitecturi de test adecvate pentru simularea şi verificarea modelelor respective.

## 2. Modelarea data-flow. Modelul data-flow al unui bistabil tip D

Modelarea data-flow reprezintă stilul de modelare în care transferul (propagarea) datelor în cadrul unei entităţi este exprimată cu ajutorul instrucţiunilor concurente de atribuire pentru semnale. În cadrul acestui stil de modelare structura entităţii nu este specificată în mod explicit dar poate fi dedusă, implicit, pe baza operatorilor logici din expresiile care se atribuie semnalelor.

Sintaxa instrucţiunii de atribuire pentru semnale care defineşte stilul data-flow este:

```
nume_semnal <= expresie [after val_timp];
```

Exemplu:

```
w1 <= s1 nor s2 after 2 ns;
```

În cadrul sintaxei, `expresie` reprezintă o expresie logică care conţine nume de semnale şi operatori logici (not, and, or, nand, nor, xor, xnor). Opţional, instrucţiunea poate conţine clauza **`after`** pentru a specifica o valoare a timpului de întârziere.

Instrucţiunea concurentă de atribuire pentru semnale se execută ori de câte ori apare un eveniment al semnalelor din expresie.

În lucrările de laborator anterioare a fost prezentată modelarea cu ecuaţii a sumatorului pe un bit şi diverse posibilităţi de testare a modelului respectiv. În prezenta lucrare, în prima parte, este prezentată modelarea data-flow în VHDL a unui bistabil de tip D, precum şi un circuit de test pentru simularea şi verificarea modelului.

### 2.1 Modelarea data-flow a bistabilului tip D

Aşa cum s-a precizat anterior, în continuare se prezintă modelarea data-flow a bistabilului tip D. Se consideră cazul bistabilului tip D având intrări Clear şi Preset asincrone, active pe 0 logic.

Structura bistabilului tip D ce urmează a fi modelat este prezentată in figura 1 şi conţine numai porţi logice de tip NAND cu 3 intrări.

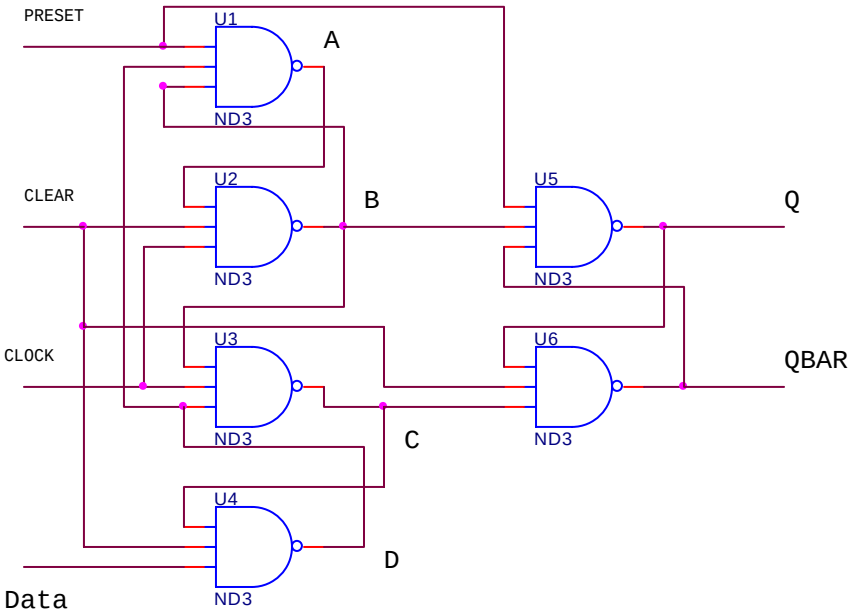


**Figura 1 -** *Bistabil tip D implementat cu porți logice NAND3*

Declarația de entitate și arhitectura în care este utilizat stilul data-flow pentru modelarea bistabilului tip D din figura 1 sunt prezentate mai jos:

**(fişier D_FF.vhd)**

```
1   library IEEE;
2   use IEEE.std_logic_1164.all;

3   entity DFF is
4     port (Preset: in Bit;
5           Clear: in Bit;
6           Clock: in Bit;
7           Data: in Bit;
8           Q: out Bit;
9           QBar: out Bit);
10  end DFF;

14  architecture Dataflow of DFF is
16    signal  A ,B ,C ,D: Bit;
17    signal QInt, QBarInt: Bit;
19  begin
21    A <= not (Preset and D and B) after 1 ns;
22    B <= not (A and Clear and Clock) after 1 ns;
23    C <= not (B and Clock and D) after 1 ns;
24    D<= not ( C and Clear and Data) after  1 ns;

26    QInt <= not (Preset and B and QBarInt ) after 1 ns;
27    QBarInt <= not (QInt and Clear and C) after 1 ns;

29    Q <= QInt;
30    QBar <= QBarInt;
32  end;
```

Ca şi în reprezentarea cu ecuaţii a sumatorului complet pe un bit, pentru a modela comportamentul bistabilului sunt utilizate instrucţiunile de atribuire pentru semnale (liniile 21-30).

Spre deosebire de modelul sumatorului pe un bit, în modelul bistabilului există o corespondenţă unu la unu între atribuirea pentru semnale şi porţi. Semnalele utilizate în specificaţiile de atribuire reprezintă căile de date ale bistabilului. În consecinţă, această descriere poartă denumirea de descriere *data-flow* (*"dataflow decription"* ).

*Observaţii:* Aşa-numita descriere cu ecuaţii, utilizată în cazul sumatorului pe 1 bit, nu reprezintă un stil de modelare distinct. Din punct de vedere al terminologiei limbajulu VHDL, orice instrucţiune concurentă de atribuire pentru semnale defineşte stilul de modelare data-flow. În descrierea bistabilului D instrucţiunile de atribuire pentru semnale nu diferă ca formă faţă de cele din cazul sumatorului pe un bit. Fiecare instrucţiune de atribuire din arhitectura de mai sus a bistabilului corespunde câte unei porţi NAND3, dar în acelaşi timp poate corespunde şi unui bloc alcătuit din 2 porţi AND şi un inversor.

În partea declarativă a arhitecturii, declaraţia de semnal de la linia 16 defineşte semnalele interne ale bistabilului, utilizate pentru interconectarea porţilor. Declaraţia de semnal de la linia 17 are un scop diferit care va fi discutat mai jos.

Mai întâi se observă faptul că semnalele din interfaţă (Q şi QBar) sunt declarate ca mod *out*, deci pot fi doar modificate (scrise) dar nu pot fi citite. Un port de mod *out* se află întotdeauna în partea stângă a instrucţiunii de atribuire pentru semnale. Examinând schema internă a bistabilului putem observa că semnalele Q şi QBar intră în bucla de reacţie ceea ce implică modificarea (scrierea) dar şi citirea lor. Un semnal intern sau un port de mod *in* sau *inout,* pentru a putea fi citit, trebuie să fie plasat în partea dreapta a instrucţiunii de atribuire pentru semnale.

Soluţia prezentată introduce semnale interne şi atribuirea valorilor lor porturilor de ieşire (liniile 29 şi 30).

Specificaţiile de atribuire (cu excepţia ultimilor două) au asociate clauza "*after 1 ns*". Aceasta reprezintă întârzierea de propagare a porţii (timpul de întârziere dintre sosirea frontului de undă la intrare şi apariţia unei modificări la ieşirea ei), dar şi întârzierea inerţială (lăţimea celui mai îngust puls propagat prin poartă).

Astfel, considerând instrucţiunea de atribuire de la linia 21, dacă semnalul de la intrarea "Preset" are un eveniment la momentul de timp de 42ns, semnalul intern "A" nu se va modifica decât la momentul de timp de 43ns. Lipsa unei clauze *after* semnifică implicit existenţa unei clauze *after 0ns*.

Totuşi, lipsa clauzei *after* nu înseamnă modificarea imediată a ieşirii odată cu semnalul de intrare. Toate atribuirile de semnale au asociate o întârziere implicită infinitezimală **Δ**. O caracteristică a întârzierii **Δ** este aceea că oricâte cicluri *delta* apar succesiv la un moment dat de timp real ele nu avansează în timp real.

Astfel, timpul în VHDL poate fi considerat ca având două dimensiuni, prima reprezentând timpul real, a doua reprezentând intervalele *delta*. Fiecare avans în timp real setează numărul curent al intervalelor *delta* la zero.

### 2.2 Testarea bistabilului de tip D

Circuitul de test pentru bistabil este prezentat mai jos :
**(fişier Test_D_FF.vhd)**

```
35  library IEEE;
36  use IEEE.std_logic_1164.all;
```

```
37  entity Test_DFF is end;

41  architecture Driver of Test_DFF is
43      component DFF
44          port (Preset , Clear, Clock, Data: in Bit;
45                Q, QBar:out Bit);
46      end component;

48      signal Preset, Clear : Bit :='1';
49      signal Clock, Data, Q, QBar :Bit;
51  begin
53      UUT: DFF port map (Preset, Clear, Clock, Data, Q, QBar);
55      Stimulus:
56      process
57      begin
61          Preset <= '0' ; wait for 5 ns;
62          Preset <= '1' ; wait for 5 ns;
63          Clear <= '0' ; wait for 5 ns;
64          Clear <= '1' ; wait for 5 ns;

68          Preset <='0'; Clear  <='0' ; wait for 5 ns;
69          Preset <='1' ;Clear <='1' ; wait for 5 ns;

73          Clear <= '0' , '1'  after 5 ns; wait for 10 ns;

77          Data <= '1'; Clock <= '0' after 1 ns, '1' after 5 ns ;
78          wait for 10 ns;
79          Data <= '0' ;  Clock <= '0'  after  1  ns, '1' after  5  ns;
80          wait for 10 ns;
            -- clear
84          Clear <='0', '1' after 5 ns; wait for 10 ns;

88          Data <= '0';
89          Preset <= '0', '1'   after 10 ns;
90          Clock <= '0', '1' after 5 ns; wait for 10 ns;
            -- stop simulation
94          wait;

96      end process;
98   end;
```

În cazul circutului de test de mai sus nu s-a mai utilizat metoda de codare a stimulilor şi răspunsurilor într-un tabel, ei fiind descrişi liniar. Aceasta deoarece testele pentru un circuit secvențial (aşa cum este bistabilul D), în special cerințele de timp, sunt mai complexe decât pentru un circuit combinațional.

În cadrul entității de test semnalele interne declarate la liniile 48-49 au fost denumite identic cu porturile componentei DFF. Ca urmare, în instrucțiunea de instanțiere a componentei DFF (linia 53) lista de după specificația **port map** sunt semnalele din entitatea de test la care se conectează porturile componentei DFF în ordinea în care ele au fost definite la declararea componentei (liniile 43-46). Ca urmare, portul Preset se conectează la semnalul Preset, ş.a.m.d. Pentru a exista corespondența conexiunii port-semnal, **este obligatoriu** ca în specificația **port map** ordinea semnalelor să coincidă cu ordinea porturilor componentei.

Semnalele de pe linia 48 (Preset şi Clear) sunt inițializate explicit cu valoarea ‘1’. În partea dreaptă a instrucțiunii de atribuire de pe linia 77 sunt prezentate elemente de formă de undă multiple, o formă convenabilă de a atribui forme de undă complexe unui semnal.

Construcția din partea dreaptă a instrucțiunii de atribuire este denumită formă de undă, iar fiecare element al formei "*expresie* **after** *timp*" este denumit element al formei de undă. Elementele sunt delimitate prin virgulă.

Putem rescrie atribuirea de semnal lui "*Clock*" din linia 77 fără a utiliza forme de undă multiple, dar codul rezultat este mai complex.

```
Clock <= '0' after 1 ns;
 wait for 1 ns;
Clock <= '1' after 4 ns;
```

Nu este corectă următoarea versiune:

```
Clock <= '0' after 1 ns;
Clock <= '1' after 5 ns;
```

Cele două instrucțiuni, fiind secvențiale, se execută una după cealaltă, dar la acelaşi moment de timp.

*Observaţie:* momentul executării unei instrucțiuni de atribuire dintr-un proces este determinat de momentul activării procesului şi de eventualele instrucțiuni *wait*. Ca urmare, nu trebuie confundat momentul executării unei instrucțiuni de atribuire cu momentele de timp la care sunt prevazute a avea loc atribuirile.

Ca urmare, ținând cont de observația anterioară, cea de-a doua instrucțiune se execută înainte de a fi avut loc atribuirea prevăzută de prima instrucțiune. Când a doua instrucțiune este executată, se ține cont de faptul că întârzierea de 5 ns reprezintă nu numai întârzierea de propagare dar şi întârzierea inerțială.

La momentul 1 ns , valoarea '0' nu va apare la ieşire, deoarece VHDL îl va interpreta ca un puls cu durată mai mică de 5 ns.

Dacă nu este specificată nici o limită de rejecție într-o instrucțiune de atribuire inerțială, atunci întârzierea primului element de formă de undă devine implicit limita de rejecție. De exemplu, în instrucțiunea următoare limita de rejecție este 1 ns.

```
Clock <= '0'  after  1  ns, '1' after  5  ns;
```

În instrucțiunea de atribuire, valorile de timp specificate după **after** în cadrul elementelor formei de undă, sunt valori față de momentul executării instrucțiunii şi nu față de momentul evenimentului anterior! De exemplu, dacă instrucțiunea de mai sus se execută la momentul T, atunci Clock va lua valoarea '0' la momentul T + 1 ns, apoi valoarea '1' la momentul T + 5 ns.

În cadrul procesului din arhitectura entității de test de mai sus, formele de undă corespunzătoare celor 4 semnale de la intrarea bistabilului pot fi generate doar cu 4 instrucțiuni de atribuire, câte una pentru fiecare semnal. În acest fel, pentru fiecare semnal se pot stabili cu mai multă uşurință momentele de timp la care să existe tranziții.

De exemplu, forma de undă pentru semnalul "Preset" poate fi generată astfel:

```
Preset <= '0', '1' after 5 ns, '0' after 20 ns,  '1' after 25 ns, '0' after 70 ns, 1 after 80 ns
```

Rezultatele simulării entității Test_DFF cu arhitectura Driver sint prezentate in figura 2.

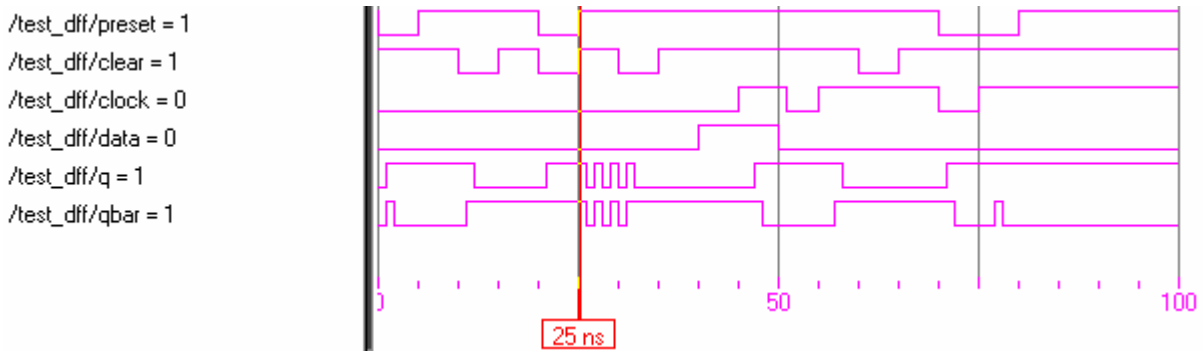


Fig.2 Formele de undă obtinute in urma simulării circuitului de test pentru verificarea modelului data-flow al bistabilului D.

### 2.3 Aplicații 1

1. În modelul data-flow al bistabilului, se poate renunța la semnalele interne Qint şi QbarInt? Ce modificări trebuie efectuate în acest scop?

2. Desenați pe o foaie de hârtie formele de undă ce urmează a fi generate pentru semnalele de la intrarea bistabilului, pe baza instrucțiunilor de atribuire din cadrul procesului din entitatea de test.

3.Copiați în directorul de lucru fişierele sursă D_FF.vhd şi Test_D_FF.vhd

4. Compilați fişierele sursă şi simulați entitatea de test.

5. Vizualizați şi examinați cu atenție formele de undă rezultate în urma simulării.

6. În fişierul Test_D_FF.vhd, creați încă o arhitectură cu numele Driver1, similară cu arhitectura Driver, dar în care formele de undă ale semnalelor de la intrarea bistabilului să fie generate în cadrul procesului de câte o singură instrucțiune pentru fiecare din cele 4 semnale. Recompilați fişierul şi repetați simularea pe baza arhitecturii Driver1.

7. Creați în fişierul Test_D_FF.vhd, încă o arhitectură cu numele Driver2, similară cu Driver1 dar în care cele 4 instrucțiuni de atribuire pentru semnalele de la intrare sunt considerate instrucțiuni concurente. Există vreo diferență față de cazul în care instrucțiunile sunt secvențiale (în cadrul unui proces)? Argumentați răspunsul. Recompilați fişierul şi reluați simularea entității de test pe baza arhitecturii Driver2.

## 3. Modelarea structurală - Latch pe 8 biți

### 3.1 Specificații

Circuitul latch este un automat asincron care modelează funcționarea unui zăvor (sau "latch"). Zăvorul are două poziții: una pentru închis şi alta pentru deschis. Prin împingere zăvorul este închis şi orice efort ulterior de închidere nu mai are nici un efect, dar prin aplicarea unui efort în sens invers (tragere) zăvorul este deschis şi va rămîne deschis indiferent de cîte ori se va mai trage.

Nu poate fi definită poziția zăvorului dacă simultan se aplică şi o tragere şi o împingere. In concluzie, rezultă că dintr-o serie de acțiuni succesive, fie de împingere, fie de tragere, are efect doar prima, iar aplicarea simultană celor două acțiuni opuse provoacă o nedeterminare în fixarea poziției zăvorului.

Pentru a modela un latch pe 8 biți utilizăm modelul bistabilului de tip D descris anterior. Structura latch-ului pe 8 biți este prezentata in figura 3.

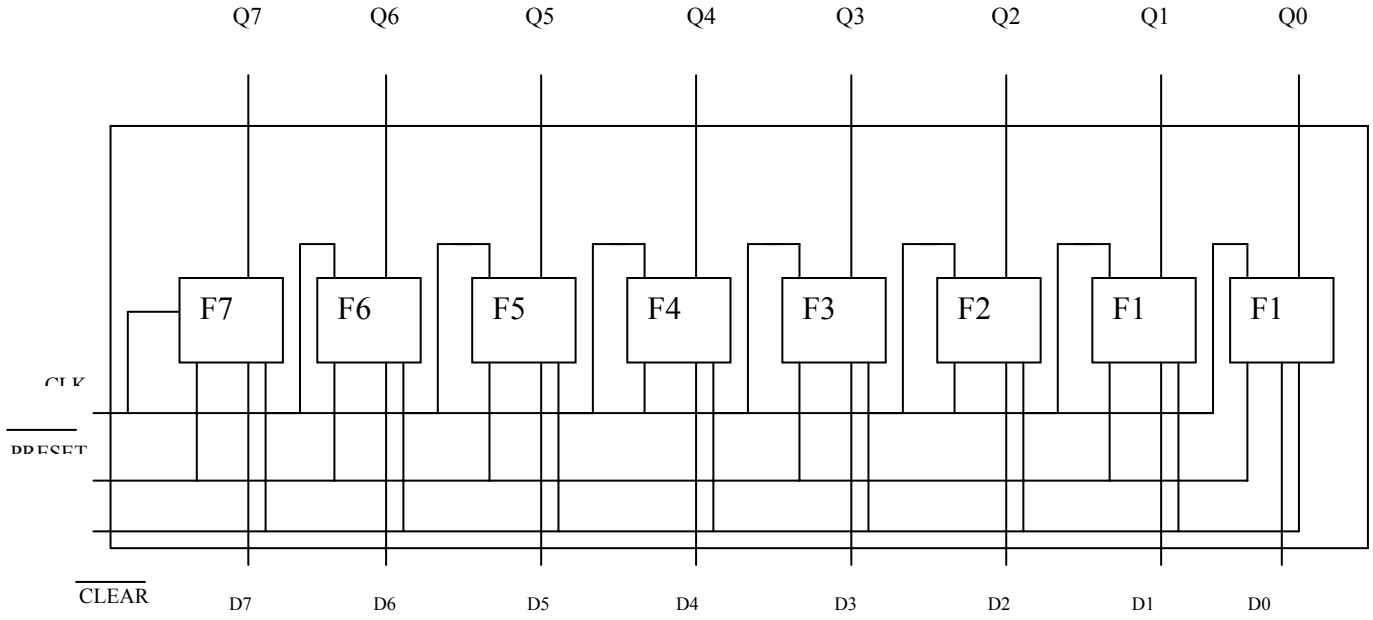


**Figura 3 -** *Structura unui latch-ul pe 8 biți*

### 3.2 Implementarea modelului structural al latch-ului

Declarația de entitate şi arhitectura a latch-ului pe 8 biti sunt prezentate mai jos:
**(fisier Latch8.vhd)**

```
1   library IEEE;
2   use IEEE.std_logic_1164.all;
3   entity  Latch8 is
4     port (D:   in Bit_Vector (7 downto 0);
5           Clk: in Bit;
6           Pre: in Bit;
7           Clr: in Bit;
8           Q:   out Bit_Vector ( 7 downto 0) );
9   end Latch8;

13  architecture Structure of Latch8 is
15    component DFF
16          port (Preset : in Bit;   Clear  : in Bit;
18                Clock  : in Bit;   Data   : in Bit;
20                Q      : out Bit;  Qbar   : out Bit  );
22    end component;
24    signal QBar: Bit_Vector (7 downto 0);
26  begin
28    F7: DFF port map ( Pre, Clr, Clk, D (7), Q (7), QBar (7) );
29    F6: DFF port map ( Pre, Clr, Clk, D (6), Q (6), QBar (6) );
30    F5: DFF port map ( Pre, Clr, Clk, D (5), Q (5), QBar (5) );
31    F4: DFF port map ( Pre, Clr, Clk, D (4), Q (4), QBar (4) );
32    F3: DFF port map ( Pre, Clr, Clk, D (3), Q (3), QBar (3) );
33    F2: DFF port map ( Pre, Clr, Clk, D (2), Q (2), QBar (2) );
34    F1: DFF port map ( Pre, Clr, Clk, D (1), Q (1), QBar (1) );
35    F0: DFF port map ( Pre, Clr, Clk, D (0), Q (0), QBar (0) );
37  end;
```

Arhitectura modelului este descrisă în întregime ca interconexiuni între instanţele componentelor. Acest tip de descriere poartă denumirea de descriere structurală.

Semnalele “D”, “Q”, “QBar” sînt de tip predefinit “Bit_Vector”, care este un tablou unidimensional de biţi. Indexul acestui vector de biţi parcurge elementele în ordine descrescătoare. Ordinea elementelor lui “D” este deci următoarea: D(7), D(6), ...,D(0).

În corpul arhitecturii este declarată componenta DFF a cărei interfaţă corespunde cu cea a modelului modelului bistabilului D prezentat anterior. Acum este posibilă crearea a 8 instanţe bistabilului.

Fiecare instanţă a bistabilului este din punct de vedere logic distinctă de oricare alta. Evenimentele dintr-o instanţă nu afectează evenimentele din altă instanţă.

Specificaţia de instanţiere a unei componente se referă la o componentă declarată într-o declaraţie de componentă si nu la o entitate declarată într-o declaraţie de entitate.

Trebuie precizat in consecinţă, pentru fiecare instanţiere a unei componente, ce declaraţie de entitate, din ce bibliotecă şi ce arhitectură asociată entităţii sa fie selectate.

Procesul prin care în VHDL se construiesc aceste copii multiple, distincte, unei componente, se numeşte detaliere (“*elaboration”*) şi implică determinarea perechii entitate-arhitectură desemnată instanţei respective.

Apoi, o copie a perechii entitate-arhitectură este creată şi atribuită instanţei. In final, sînt aplicate particularizările fiecărei instanţe (cum ar fi conexiunile semnalelor).

Atribuirea perechii entitate-arhitectură instanţei poate fi efectuată în întregime de utilizator, dar poate fi subiectul regulilor implicite ale VHDL. Conform acestor reguli, o entitate cu următoarele caracteristici este selectată pentru a fi atribuită unei instanţe a unei componente date, dacă:

- entitatea are acelaşi nume cu cel al componentei
- pentru fiecare port al componentei există un port al entităţii cu acelaşi nume, acelaşi mod (direcţie) şi tip.

Pentru o entitate dată este utilizată arhitectura cel mai recent compilată. Inexistenţa unei arhitecturi este o eroare.

### 3.3 Testarea modeluli structural al latch-ului pe 8 biţi

Mai jos este prezentat un circuit de test pentru latch-ul pe 8 biti :
**(fisier Test_Latch8.vhd)**

```
40  library IEEE;
41  use IEEE.std_logic_1164.all;

42  entity  Test_Latch8 is end;

46  architecture Driver of Test_Latch8 is

48    component Latch8
49          port ( D: in Bit_Vector (7 downto 0);
50                 Clk : in Bit;
51                 Pre : in Bit;
52                 Clr: in Bit;
53                 Q: out Bit_Vector ( 7 downto 0));
54    end component;

56    signal D, Q: Bit_Vector (7 downto 0);
57    signal Clk, Pre, Clr: Bit := '1' ;

59  begin
61    UUT: Latch8 port map (D, Clk, pre, Clr, Q);
```

```
63    Stimulus:
64    process
65          variable Temp : Bit_Vector ( 7 downto 0);
66       begin
68        --set the latch--
70          Pre <= '0', '1' after 5ns;
70          wait for 10ns;
73        --clear the latch--
75          Clr <= '0', '1' after 5 ns;
76          wait for 10 ns;
78        --load the latch--
80          Temp := "00010011";
81          for i in 1 to 8 loop
82                D <= temp;
83                Clk <= '0' after 0 ns,  '1' after 5 ns;
84                wait for 10 ns;
85                assert Q = Temp report "Load failed";
86                Temp := Temp(0) & Temp ( 7 downto 1);
87          end loop;
89          wait;
91    end process;
93  end;
```

În cadrul testului după setare şi inițializare este încărcat stimulul “00010011” şi se verifică corectitudinea răspunsului. Apoi, se deplasează cu o poziție fiecare bit la dreapta, se reîncarcă stimulul şi se verifică răspunsul.

Această secvență de deplasare, încărcare şi verificare are loc pînă la epuizarea tuturor permutărilor circulare a biților stimulului.

In modelul de mai sus sînt introduse declarațiide variabile. Acestea sînt elemente de memorare locale proceselor şi nu au corespondență în hardware. In VHDL, variabilele prezinta aceleasi proprietati ca in limbajele de programare Pascal, C sau C++.

Spre deosebire de instrucțiunile de atribuire pentru semnale unde acestea aveau loc dupa un interval de timp (masurabil in intervale de timp **Δ**), în cazul variabilelor atribuirea are loc instantaneu.

Astfel dacă instrucțiunile de atribuire în cazul semnalelor:

A <=B ;

B <=A ;

au ca efect interschimbarea valorilor variabilelor A şi B, iar în cazul variabilelor

A :=B;

B :=A;

au ca efect plasarea valorii lui B în A .

In arhitecturile structurale din VHDL semnalele sînt asimilate interconexiunilor. In consecinta, obiectele de tip semnal prezinta semnificatie fizica.

In arhitecturile comportamentale semnalele pot fi interpretate ca variabile de stare.

Variabila este încărcată cu vectorul de test. Instrucțiunea *assert* prezintă clauza *report*. Mesajul “ *Load Failed*” este afişat de fiecare dată cînd condiția testată de *assert* este falsă.

La linia 86 are loc deplasarea circulară a biților. Expresia din partea dreaptă a instrucțiunii de atribuire *“Temp := Temp(0) & Temp ( 7 downto 1);”* utilizează operatorul de concatenare “*&*” care este definit pentru pentru tablouri unidimensionale de orice tip.

Operandul din stînga ,*Temp(0),* furnizează bitul cel mai la dreapta al variabilei *Temp*. Operandul din dreapta*, Temp (7 downto 1),* furnizează primii 7 biți din stînga ai variabilei Temp.

Operatorul de concatenare generează un rezultat a cărui lungime este suma lungimii operanzilor (aici 8), ale cărui elemente sînt formate din cele ale operandului din stînga urmate de elementele operandului din dreapta.

Astfel, această expresie deplasează cu o poziție la dreapta variabila *Temp*. Noua valoare obținută prin deplasare este memorată apoi în *Temp*.

### 3.4 Aplicații 2

**1.** Copiaţi fişierele sursă Latch8.vhd şi Test_Latch8.vhd în directorul de lucru.
**2.** Compilati fişierele sursă şi simulaţi entitatea de test.
**3.** Examinati cu atentie formele de unda din fereastra Wave şi explicati-le.
**4.** În fişierul D_FF.vhd descrieţi o noua arhitectura denumita Comportamental a entitatii DFF utilizând constructia *process* şi instrucțiuni secvențiale în locul instrucțiunilor de atribuire concurente pentru semnale.
**5.** Scrieti codul VHDL pentru modelarea structurala a latch-ul pe 16 biti si circuitul de test asociat. Ce constructie VHDL va poate ajuta la ultima cerinta?