LUCRAREA NR. 5
ATRIBUTE

1. Scopul lucrării

Lucrarea prezintă pe larg toate atributele existente în VHDL, atât cele predefinite cât şi cele ce pot fi definite de către proiectant. Se oferă exemple în cazul fiecărui atribut şi se explică pe larg modul lor de utilizare.

2. Consideraţii teoretice

2.1 Definiţie

Un *atribut* este o caracteristică asociată unui tip sau unui obiect (de exemplu, numărul de elemente ale unui tablou) care va putea fi cunoscută în mod dinamic în cursul rulării programului.

Atributele se notează prin adăugarea unui *apostrof* după numele tipului sau al obiectului. Astfel, de exemplu, cel mai mare număr întreg care poate fi reprezentat se scrie INTEGER'HIGH.

Există două mari clase de atribute:

- atribute predefinite;
- atribute care urmează să fie definite de către proiectant.

2.2 Atribute predefinite

Atributele predefinite sunt foarte utilizate în VHDL, ele intrând adeseori în componenţa anumitor funcţii utilitare şi contribuind astfel la simplificarea scrierii anumitor teste. De exemplu, detectarea frontului ascendent (cel mai adesea, al semnalului de tact) sau verificarea stabilităţii nivelului HIGH - în ambele cazuri, pentru un semnal de tip BIT.

Atributele predefinite pot fi:

- valori (tipizate);
- funcţii;
- tipuri;
- intervale de variaţie.

Ele se aplică unor prefixe care pot fi, la rândul lor:
- valori;
- tipuri;
- etichete de blocuri.

În continuare se prezintă lista atributelor predefinite pentru fiecare gen de prefix.

2.2.1 Atribute definite pe tipuri

Vom utiliza în continuare, ca exemplu de lucru, următoarele definiţii de tipuri:

```
-- Tipul NIVEL este definit ca un şir de valori enumerate
type NIVEL is ('U', '0', '1', 'Z');

-- Tipul ADRESĂ e definit ca un întreg cuprins între 7 şi 0
type ADRESĂ is range 7 downto 0;

-- Tipul COD este definit prin următorul tip enumerat
type COD is (NOP, ADD, PLUS1);
```

În continuarea acestei secţiuni, notaţia T desemnează un tip, prefix al atributului, adică un obiect asupra căruia operează atributul. Trebuie făcută distincţia dintre acest obiect şi eventualii parametri ai atributului, care apar între paranteze după numele atributului.

```
obiect_pe_care_operează_atributul'nume_atribut(parametri)
```

T'BASE

Rezultatul său este tipul de bază al lui T, unde T poate fi orice tip sau sub-tip. Acest atribut nu este permis decât în expresia prefixului unui alt atribut (de exemplu T'BASE'LEFT).

T'HIGH

Rezultatul său este o valoare de tipul T. Prefixul T trebuie să fie de tip scalar sau să fie un sub-tip scalar. Valoarea returnată este cea mai mare valoare implementată a tipului T.

```
-- Vom utiliza tipul INTEGER; variabila INT este iniţializată
-- la cea mai mare valoare întreagă, spre deosebire de
-- iniţializarea implicită, care ar fi egală cu INTEGER'LOW.
variable INT: INTEGER := INTEGER'HIGH;
-- Definirea unui sub-tip MMARE10 cu valori cuprinse între 11
-- şi cel mai mare număr întreg reprezentabil
subtype MMARE10 is INTEGER range 11 to INTEGER'HIGH;
```

T'LOW

Rezultatul său este o valoare de tipul T. Prefixul T trebuie să fie de tip scalar sau să fie un sub-tip scalar. Valoarea returnată este cea mai mică valoare implementată a tipului T.

În cazul tipului ADRESĂ definit la începutul acestei secţiuni, ADRESĂ'LOW este egal cu 0.

T'LEFT

Rezultatul său este o valoare de tipul T. Prefixul T trebuie să fie de tip scalar sau să fie un sub-tip scalar. Valoarea returnată corespunde limitei din stânga a tipului T; aceasta nu trebuie confundată cu cea mai mică valoare a lui T. T'LEFT constituie valoarea de iniţializare implicită a tuturor tipurilor scalare.

Pentru tipul INTEGER, INTEGER'LEFT = INTEGER'LOW.

Pentru tipul NIVEL, NIVEL'LEFT = 'U'.

Pentru tipul ADRESĂ, ADRESĂ'LEFT = 7.

T'RIGHT

Rezultatul său este o valoare de tipul T. Prefixul T trebuie să fie de tip scalar sau să fie un sub-tip scalar. Valoarea returnată corespunde limitei din dreapta a tipului T; aceasta nu trebuie confundată cu cea mai mare valoare a lui T.

Pentru tipul INTEGER, INTEGER'RIGHT = INTEGER'HIGH.

Pentru tipul NIVEL, NIVEL'RIGHT = 'Z'.

Pentru tipul ADRESĂ, ADRESĂ'RIGHT = 0.

T'POS (X)

Acest atribut este o funcţie de parametru X; X este de tipul de bază al lui T. Prefixul T este un tip sau un sub-tip enumerat sau fizic. Valoarea returnată de către această funcţie este un întreg universal

(UNIVERSAL_INTEGER) care dă poziţia valorii parametrului X în cadrul enumerării. Poziţia primului element este zero.

De exemplu, TIME'POS (1 ns) este de tipul UNIVERSAL_INTEGER.

T'VAL (X)

Acest atribut este o funcţie de parametru X; X este o expresie de tip întreg. Prefixul T este un tip sau un sub-tip enumerat sau fizic. Valoarea returnată de către această funcţie este cea a cărei poziţie este egală cu X. Poziţia primului element al unei enumerări este zero.

De exemplu, NIVEL'VAL(NIVEL'POS('Z')) are valoarea 'Z'.

T'SUCC (X)

Acest atribut este o funcţie de parametru X; X este de tipul de bază al lui T. Prefixul T este un tip sau un sub-tip enumerat sau fizic. Valoarea returnată de către această funcţie este cea a cărei poziţie este egală cu X + 1.

Observaţie

Se va genera o eroare dacă X este egal cu T'BASE'HIGH.

De exemplu, NIVEL'SUCC(NIVEL'LEFT) are valoarea '0'.
COD'SUCC(NOP) are valoarea ADD.
ADRESĂ'SUCC(6) are valoarea 7.

T'PRED (X)

Acest atribut este o funcţie de parametru X; X este de tipul de bază al lui T. Prefixul T este un tip sau un sub-tip enumerat sau fizic. Valoarea returnată de către această funcţie este cea a cărei poziţie este egală cu X - 1.

Observaţie

Se va genera o eroare dacă X este egal cu T'BASE'LOW.

De exemplu, NIVEL'PRED('Z') are valoarea '1'.
COD'PRED(PLUS1) are valoarea ADD.
ADRESĂ'PRED(6) are valoarea 5.

T'LEFTOF (X)

Acest atribut este o funcţie de parametru X; X este o expresie al cărei tip este tipul de bază al lui T. Prefixul T este un tip sau un sub-tip enumerat sau fizic. Valoarea returnată de către această funcţie este cea a cărei poziţie se află la stânga celei a lui X.

Observaţie

Se va genera o eroare dacă X este egal cu T'BASE'LEFT.

De exemplu, NIVEL'LEFTOF('Z') are valoarea '1'.
COD'LEFTOF(ADD) are valoarea NOP.
ADRESĂ'LEFTOF(6) are valoarea 7.

T'RIGHTOF (X)

Acest atribut este o funcţie de parametru X; X este o expresie al cărei tip este tipul de bază al lui T. Prefixul T este un tip sau un sub-tip enumerat sau fizic. Valoarea returnată de către această funcţie este cea a cărei poziţie se află la dreapta celei a lui X.

Observaţie

Se va genera o eroare dacă X este egal cu T'BASE'RIGHT.

De exemplu, NIVEL'RIGHTOF('0') are valoarea '1'.
COD'RIGHTOF(ADD) are valoarea PLUS1.
ADRESĂ'RIGHTOF(6) are valoarea 5.

T'ASCENDING

Acest atribut este o funcţie care returnează o valoare booleană TRUE dacă intervalul tipului este crescător (**to**) şi FALSE în caz contrar (**downto**). Prefixul T este un tip sau un sub-tip enumerat sau fizic.

NIVEL'ASCENDING este TRUE.
COD'ASCENDING este TRUE.
ADRESĂ'ASCENDING este FALSE.

T'IMAGE (X)

Acest atribut este o funcţie care are drept parametru o valoare de tip scalar X şi care returnează un şir de caractere (de tip STRING) reprezentând

valoarea respectivă. Prefixul T este deci un tip sau un sub-tip scalar. Acest tip este acelaşi cu tipul lui X.

De exemplu, COD'IMAGE(ADD) returnează şirul de caractere ”ADD”. Acest atribut ar putea fi utilizat, de exemplu, în mesajul unei instrucţiuni **assert**:

```
assert CONDIŢIE report "Valoarea este:"& COD'IMAGE(ADD);
```

T'VALUE (X)

Acest atribut este o funcţie care are drept parametru un şir de caractere (de tip STRING) şi returnează valoarea de tipul T care îi corespunde. Dacă nu se recunoaşte nici o valoare a tipului, se generează o eroare. Prefixul T este un tip sau un sub-tip scalar. X este deci de tipul STRING. Acest atribut nu este sensibil la majuscule; este complementarul atributului precedent.

COD'VALUE("ADD") returnează valoarea ADD de tipul COD.
COD'VALUE ("adD") returnează valoarea ADD de tipul COD.
COD'VALUE ("ALPHA") returnează o eroare.

2.2.2 Atribute definite pe tipuri sau pe sub-tipuri tablou

Vom utiliza în continuare următorul cadru de exemple:

```
constant NUME: STRING := "MIHĂILESCU";
variable PRENUME: STRING (8 downto 1) := "BOGDAN";
subtype CUVÂNT is BIT_VECTOR (0 to 7);
subtype BIT_COD is BIT_VECTOR (3 downto 1);
type MEMORIE is array (INTEGER range <>) of CUVÂNT;
subtype BLOC is MEMORIE (16 downto 1);
type ECRAN is array(INTEGER range<>, INTEGER range<>) of BIT;
variable MEM: MEMORIE (1 to 256);
variable BL: BLOC;
variable CV: CUVÂNT;
alias COD: BIT_COD is CV (0 to 2);
variable VIZ: ECRAN (255 downto 1, 1 to 1024);
```

În continuarea acestei secţiuni, notaţia A desemnează un tablou (sau un sub-tip de tablou), prefix al atributului, adică un obiect asupra căruia operează atributul. Acest obiect trebuie distins de eventualii parametri ai atributului, aceştia din urmă apărând între paranteze după numele atributului.

```
obiect_pe_care_operează_atributul'nume_atribut(parametri)
```

A'LEFT[(N)]

Acest atribut este o funcţie care are drept parametru o expresie statică de tip întreg universal N a cărui valoare nu trebuie să depăşească numărul de dimensiuni ale tabloului A. N este opţional, având implicit valoarea 1. Prefixul A poate fi un obiect de tip tablou, sau un obiect de sub-tip constrâns de tablou, sau un alias.

Valoarea returnată de această funcţie este valoarea capătului din stânga al celui de-al N-lea index în intervalul de variaţie al declaraţiei lui A. Dacă A este un alias al unui obiect de tip tablou, valoarea returnată va fi extrasă din declaraţia lui A şi nu din cea a obiectului.

Aşadar, NUME'LEFT este egal cu un întreg universal de valoare necunoscută, în schimb NUME (NUME'LEFT) este egal cu 'M'.

PRENUME'LEFT este 8.
MEM'LEFT este 1.
VIZ'LEFT (2) este 1.
BL'LEFT este 16.
CV'LEFT este 0.
COD'LEFT este 3.

A'RIGHT[(N)]

Acest atribut este o funcţie care are drept parametru o expresie statică de tip întreg universal N a cărui valoare nu trebuie să depăşească numărul de dimensiuni ale tabloului A. N este opţional, având implicit valoarea 1. Prefixul A poate fi un obiect de tip tablou, sau un obiect de sub-tip constrâns de tablou, sau un alias.

Valoarea returnată de această funcţie este valoarea capătului din dreapta al celui de-al N-lea index în intervalul de variaţie al declaraţiei lui A. Dacă A este un alias al unui obiect de tip tablou, valoarea returnată va fi extrasă din declaraţia lui A şi nu din cea a obiectului.

Aşadar, NUME'RIGHT este egal cu un întreg universal de valoare necunoscută, în schimb NUME (NUME'RIGHT) este egal cu 'U'.

PRENUME'RIGHT este 1.
MEM'RIGHT este 256.
VIZ'RIGHT (2) este 1024.
BL'RIGHT este 1.

CV'RIGHT este 7.
COD'RIGHT este 1.

A'HIGH[(N)]

Acest atribut este o funcţie care are drept parametru o expresie statică de tip întreg universal N a cărui valoare nu trebuie să depăşească numărul de dimensiuni ale tabloului A. N este opţional, având implicit valoarea 1. Prefixul A poate fi un obiect de tip tablou, sau un obiect de sub-tip constrâns de tablou, sau un alias de tablou.

Valoarea returnată de această funcţie este valoarea capătului superior al celui de-al N-lea index în intervalul de variaţie al declaraţiei lui A. Dacă A este un alias al unui obiect de tip tablou, valoarea returnată va fi extrasă din declaraţia lui A şi nu din cea a obiectului.

Aşadar, NUME'HIGH este egal cu un întreg universal de valoare necunoscută, în schimb NUME (NUME'HIGH) este egal cu 'U', deoarece tipul STRING este definit cu un index de tip POSITIVE, care este crescător de la 1 la INTEGER'HIGH.

PRENUME'HIGH este 8.
MEM'HIGH este 256.
VIZ'HIGH (2) este 1024.
BL'HIGH este 16.
CV'HIGH este 7.
COD'HIGH este 3.

A'LOW [(N)]

Acest atribut este o funcţie care are drept parametru o expresie statică de tip întreg universal N a cărui valoare nu trebuie să depăşească numărul de dimensiuni ale tabloului A. N este opţional, având implicit valoarea 1. Prefixul A poate fi un obiect de tip tablou, sau un obiect de sub-tip constrâns de tablou, sau un alias de tablou.

Valoarea returnată de această funcţie este valoarea capătului inferior al celui de-al N-lea index în intervalul de variaţie al declaraţiei lui A. Dacă A este un alias al unui obiect de tip tablou, valoarea returnată va fi extrasă din declaraţia lui A şi nu din cea a obiectului.

Aşadar, NUME'LOW este egal cu un întreg universal de valoare necunoscută, în schimb NUME (NUME'LOW) este egal cu 'M', deoarece tipul STRING este definit cu un index de tip POSITIVE, care este crescător de la 1 la INTEGER'HIGH.

PRENUME'LOW este 1.
MEM'LOW este 1.
VIZ'LOW (2) este 1.
BL'LOW este 1.
CV'LOW este 0.
COD'LOW este 1.

A'RANGE [(N)]

Acest atribut este un interval de variaţie care are drept parametru o expresie statică de tip întreg universal N a cărui valoare nu trebuie să depăşească numărul de dimensiuni ale tabloului A. N este opţional, având implicit valoarea 1. Prefixul A poate fi un obiect de tip tablou, sau un alias de tablou, sau un obiect de sub-tip constrâns de tip tablou.

Valoarea returnată de această funcţie este intervalul de variaţie al celui de-al N-lea index din declaraţia lui A. Dacă A este un alias al unui obiect de tip tablou, valoarea returnată va fi extrasă din declaraţia lui A şi nu din cea a obiectului.

Dacă cel de-al N-lea index este crescător, atunci A'RANGE(N) va avea ca valoare intervalul de variaţie A'LEFT(N) **to** A'RIGHT(N). Dacă cel de-al N-lea index este descrescător, atunci A'RANGE(N) va avea ca valoare intervalul de variaţie A'RIGHT(N) **downto** A'LEFT(N).

Aşadar, NUME'RANGE este un interval de variaţie crescător căruia nu i se cunosc decât valorile inferioară şi superioară. Totuşi, se poate scrie:

```
for i in NUME'RANGE loop
-- Se vor baleia indicii corespunzători lui 'M', apoi lui
-- 'I',... până la 'U'
```

PRENUME'RANGE este 8 **downto** 1.
MEM'RANGE este 1 **to** 256.
VIZ'RANGE (2) este 1 **to** 1024.
BL'RANGE este 16 **downto** 1.
CV'RANGE este 0 **to** 7.
COD'RANGE este 3 **downto** 1.

A'REVERSE_RANGE [(N)]

Acest atribut este un interval de variaţie care are drept parametru o expresie statică de tip întreg universal N a cărui valoare nu trebuie să

depăşească numărul de dimensiuni ale tabloului A. N este opţional, având implicit valoarea 1. Prefixul A poate fi un obiect de tip tablou, sau un alias de tablou, sau un obiect de sub-tip constrâns de tip tablou.

Valoarea returnată de această funcţie este intervalul de variaţie al celui de-al N-lea index din declaraţia lui A, luat însă în ordine inversă. Dacă A este un alias al unui obiect de tip tablou, valoarea returnată va fi extrasă din declaraţia lui A şi nu din cea a obiectului.

Dacă cel de-al N-lea index este crescător, atunci A'REVERSE_RANGE(N) va avea ca valoare intervalul de variaţie A'RIGHT(N) **downto** A'LEFT(N). Dacă cel de-al N-lea index este descrescător, atunci A'REVERSE_RANGE(N) va avea ca valoare intervalul de variaţie A'LEFT(N) **to** A'RIGHT(N).

Aşadar, NUME'REVERSE_RANGE este un interval de variaţie descrescător căruia nu i se cunosc decât valorile superioară şi inferioară. Totuşi, se poate scrie:

```
for i in NUME'REVERSE_RANGE loop
-- Se vor baleia indicii corespunzători lui 'U', apoi lui
-- 'C',... până la 'M'
```

PRENUME'REVERSE_RANGE este 1 **to** 8.
MEM'REVERSE_RANGE este 256 **downto** 1.
VIZ'REVERSE_RANGE (2) este 1024 **downto** 1.
BL'REVERSE_RANGE este 1 **to** 16.
CV'REVERSE_RANGE este 7 **downto** 0.
COD'REVERSE_RANGE este 1 **to** 3.

A'LENGTH[(N)]

Acest atribut returnează o valoare de tip întreg universal care corespunde numărului de valori din cel de-al N-lea index al declaraţiei lui A. N este o expresie statică de tip întreg universal a cărei valoare nu trebuie să depăşească numărul de dimensiuni ale tabloului A. N este opţional, având implicit valoarea 1. Prefixul A poate fi un obiect de tip tablou, sau un alias de tablou sau un obiect de sub-tip constrâns de tip tablou. Dacă A este un alias al unui obiect de tip tablou, valoarea returnată va fi extrasă din declaraţia lui A şi nu din cea a obiectului.

De fapt, A'LENGTH(N) este egal cu A'HIGH(N) - A'LOW(N) + 1.
NUME'LENGTH este 10.

PRENUME'LENGTH este 8.
MEM'LENGTH este 256.
VIZ'LENGTH (2) este 1024.
BL'LENGTH este 16.
CV'LENGTH este 8.
COD'LENGTH este 3.

A'ASCENDING[(N)]

Acest atribut returnează o valoare booleană, care este TRUE dacă cel de-al N-lea index al tabloului A este crescător (**to**) şi FALSE în caz contrar (**downto**). N este o expresie statică de tip întreg universal a cărei valoare nu trebuie să depăşească numărul de dimensiuni ale tabloului A. N este opţional, având implicit valoarea 1. Prefixul A poate fi un obiect de tip tablou, sau un alias de tablou, sau un obiect de sub-tip constrâns de tip tablou. Dacă A este un alias al unui obiect de tip tablou, valoarea returnată va fi extrasă din declaraţia lui A şi nu din cea a obiectului.

NUME'ASCENDING este TRUE.
PRENUME'ASCENDING este FALSE.
MEM'ASCENDING este TRUE.
VIZ'ASCENDING (2) este TRUE.
BL'ASCENDING este FALSE.
CV'ASCENDING este TRUE.
COD'ASCENDING este TRUE.

2.2.3 Atribute definite pe semnale

Atributele definite pe semnale pot fi grupate în două categorii care diferă prin tipul rezultatului: semnal sau funcţie.

În continuarea acestei secţiuni, notaţia S desemnează un semnal, prefix al atributului, adică un obiect asupra căruia operează atributul. Acest obiect trebuie distins de eventualii parametri ai atributului, aceştia din urmă apărând între paranteze după numele atributului.

a) Atribute semnal

S'DELAYED [(T)]

Acest atribut este un semnal identic cu S, dar întârziat cu T unităţi de timp. T este deci o expresie statică pozitivă sau nulă de tipul fizic predefinit TIME. Ea poate fi omisă, caz în care va lua valoarea 0 ns. Atenţie,

S'DELAYED (0 ns) nu este identic cu S atunci când S tocmai şi-a schimbat valoarea în acelaşi pas al simulării.

Fiind dat T un timp pozitiv sau nul, iar R – un semnal de acelaşi subtip ca şi cel al semnalului S, dacă R şi S au aceleaşi valori iniţiale, atunci R este identic cu S'DELAYED(T) oricare ar fi T.

Procesul echivalent ar putea fi următorul:

```
process(S)
begin
      R <= transport S after T;
end process;
```

S'STABLE [(T)]

Acest atribut este un semnal de tip boolean. S este un semnal static. T este o expresie statică pozitivă sau nulă de tipul fizic predefinit TIME. Ea poate fi omisă, caz în care va lua valoarea 0 ns. Rezultatul este adevărat dacă în decursul intervalului de timp T nu a apărut nici un eveniment pe semnalul S şi fals în caz contrar.

Observaţii

S'STABLE(0 ns) este fals dacă S tocmai şi-a schimbat valoarea. Aceasta se reduce la a afirma că dacă semnalul S întârziat cu 0 ns este egal cu semnalul S, atunci S este stabil de 0 ns, ceea ce se scrie:

S'STABLE(0 ns) = (S'DELAYED(0 ns) = S)

Dacă S'STABLE(T) are valoarea FALSE, atunci înseamnă că, prin definiţie, există un timp t, cu 0 ns ≤ t ≤ T, astfel încât S'DELAYED(t) să fie diferit de S.

Dacă T este cea mai mică valoare de timp astfel încât S'STABLE(T) are valoarea FALSE, atunci pentru toţi t (0 ns ≤ t ≤ T), S'DELAYED(t) = S.

S'QUIET [(T)]

Acest atribut este un semnal de tip boolean. S este un semnal static. T este o expresie statică pozitivă sau nulă de tipul fizic predefinit TIME. Ea poate fi omisă, caz în care va lua valoarea 0 ns. Rezultatul este adevărat dacă în decursul intervalului de timp T semnalul S „a rămas liniştit" (pe el nu a apărut nici un eveniment şi nici o tranzacţie) şi este fals în caz contrar.

Termenul „liniştit” a fost introdus aici pentru a traduce cuvântul QUIET; de fapt, în contextul limbajului VHDL, el are o semnificaţie opusă termenului ACTIV.

Pentru un ciclu de simulare dat, S’QUIET(0 ns) are valoarea TRUE dacă şi numai dacă semnalul S este „liniştit” în decursul acestui ciclu de simulare, adică dacă semnalul S nu se află pe lista semnalelor ai căror piloţi (*driver*-e) sunt luate în considerare pentru ciclul de simulare curent.

S’TRANSACTION

Acest atribut este un semnal de tip predefinit BIT. Dacă S este numele unui semnal static, acest atribut îşi schimbă valoarea (din ‘0’ devine ‘1’ şi invers) de fiecare dată când semnalul S devine activ în decursul unui ciclu de simulare.

b) Atribute funcţie

S’EVENT

Acest atribut este o funcţie de tip boolean. Prefixul S identifică un semnal static. Valoarea returnată de această funcţie este TRUE atunci când tocmai a avut loc un eveniment pe semnalul S în decursul ciclului de simulare.

Un eveniment este caracterizat de o schimbare a valorii unui semnal activ. Pentru un semnal de tip compus, este suficient ca un eveniment să aibă loc pe unul dintre sub-elementele sale pentru ca această funcţie atribut să returneze valoarea TRUE.

S’ACTIVE

Acest atribut este o funcţie de tip boolean. S este numele unui semnal static. Valoarea returnată de această funcţie este TRUE atunci când semnalul S este activ în decursul ciclului de simulare şi FALSE în caz contrar. Dacă semnalul este de tip compus, este suficient ca unul dintre sub-elementele sale scalare să fie activ.

S’LAST_EVENT

Acest atribut este o funcţie care returnează o valoare de tipul predefinit TIME. Prefixul S este numele unui semnal static. Valoarea returnată de această funcţie reprezintă timpul scurs de la ultimul eveniment survenit pe semnalul S.

Pentru un semnal S compus, S'LAST_EVENT returnează valoarea minimă a lui R'LAST_EVENT a fiecărui sub-element R al lui S.

S'LAST_ACTIVE

Acest atribut este o funcţie care returnează o valoare de tipul predefinit TIME. Prefixul S este numele unui semnal static. Valoarea returnată de această funcţie reprezintă timpul scurs din momentul în care semnalul S a fost activ pentru ultima dată.

Pentru un semnal S compus, S'LAST_ACTIVE returnează valoarea minimă a lui R'LAST_ ACTIVE a fiecărui sub-element R al lui S.

S'LAST_VALUE

Acest atribut este o funcţie care returnează o valoare de tipul semnalului S. Prefixul S este numele unui semnal static. Valoarea returnată de această funcţie este cea a semnalului S imediat înaintea ultimei modificări a lui S.

Pentru un semnal S de tip scalar, S'LAST_VALUE este egal cu S'DELAYED(T), unde T este de tip TIME, este ≥ 0 ns şi ia valoarea cea mai mică astfel încât S'STABLE(T) să fie FALSE. Dacă o asemenea valoare de timp nu există, S'LAST_VALUE returnează valoarea semnalului S.

S'DRIVING_VALUE şi **S'DRIVING**

Aceste două atribute sunt funcţii şi sunt oarecum diferite de cele precedente datorită modului lor de utilizare. Ele au fost introduse pentru a se evita necesitatea de a citi valoarea unui port de ieşire (a unui port de mod **out**).

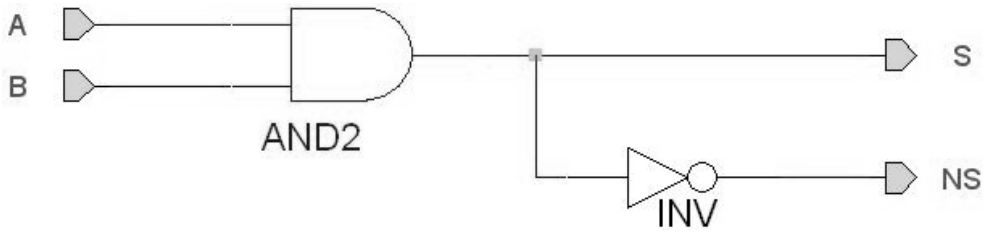

Figura 5.1 *Exemplu ilustrativ pentru atributele S'DRIVING_VALUE şi S'DRIVING*

Examinând schema precedentă, am înclina să o descriem astfel:

```
S <= A and B;
NS<= not S;
```

Din păcate, această descriere este ilegală căci portul de ieşire S nu poate fi citit şi deci nu poate apărea în membrul drept al unei atribuiri.

Această restricţie asupra utilizării în limbajul VHDL a unui port de ieşire poate părea incomodă. De fapt, demersul proiectanţilor VHDL-ului se explică foarte bine. A avea acces la valoarea S semnifică, dacă S este un semnal rezolvat, a avea acces la valoarea rezolvată, adică la valoarea obţinută după apelarea funcţiei de rezoluţie. Or, această valoare rezolvată poate să depindă la rândul ei de alte semnale sursă, exterioare blocului respectiv. A utiliza această valoare în interiorul descrierii noastre ar fi constituit un mijloc (e drept, indirect) de a face ca informaţia să intre pe un port de ieşire (în celelalte blocuri care se execută concurent cu blocul nostru).

Confruntat cu această problemă, proiectantul nu ar fi avut altă soluţie decât:

a) Să schimbe natura portului de ieşire în port bidirecţional (ceea ce ar fi avut urmări profund nocive asupra logicii descrierii: un port de ieşire nu este în nici un caz acelaşi lucru cu un port bidirecţional!)

b) Să declare o variabilă locală (într-un context secvenţial) sau un semnal local (într-un context concurent) care să joace rolul de tampon. Şi în acest caz, această declaraţie inutilă nu este acceptabilă din punct de vedere conceptual.

Atributul DRIVING_VALUE permite rezolvarea acestei probleme. Într-adevăr, spre deosebire de S, care desemnează în cazul precedent semnalul, eventual rezolvat, prezent la ieşire, S'DRIVING_VALUE desemnează exclusiv valoarea cu care procesul contribuie la eventuala funcţie de rezoluţie. S'DRIVING_VALUE poate deci să fie subiectul unei instrucţiuni de atribuire. Astfel, exemplul anterior se va scrie de data aceasta:

```
S <= A and B;
NS<= not S'DRIVING_VALUE;
```

Atributul S'DRIVING este mai puţin folosit. Într-adevăr, în cazul în care, în mod dinamic, nu se ştie dacă semnalul S este deconectat sau nu, următoarea porţiune de cod permite evitarea unei erori:

```
if S'DRIVING then
      V := S'DRIVING_VALUE;
end if;
```

S'DRIVING este un atribut care returnează o valoare booleană cu valoarea TRUE dacă semnalul S nu este deconectat.

5.2.2.4 Atribute definite pe obiecte în sens larg

Următoarele trei atribute au particularitatea de a se putea aplica pe majoritatea obiectelor, în sens larg, din VHDL. Astfel, prefixul X poate desemna o entitate, o variabilă, o etichetă etc.

Nici unul dintre aceste atribute nu are parametri şi toate returnează o valoare de tip STRING (şir de caractere) scris cu litere mici. Principala întrebuinţare a acestor atribute constă în elaborarea de mesaje (sau **report**) care permit trasarea unui cod sursă fără utilizarea instrumentelor software de depanare (*debugging*).

X'SIMPLE_NAME

Acest atribut returnează, sub forma unui şir de caractere, numele X. Astfel, dacă SIG este un semnal, SIG'SIMPLE_NAME va returna şirul de caractere "sig".

X'PATH_NAME

Acest atribut returnează un şir de caractere care, pe lângă numele X, conţine şi suita de etichete care permite revenirea, de-a lungul structurii descrierii date de către proiectant, până la X: etichete de blocuri, de instanţe de componente etc.

X'INSTANCE_NAME

Acest atribut amestecă informaţiile structurale oferite de PATH_NAME cu alte informaţi relative la configuraţie (numele entităţii, numele arhitecturii, numele configuraţiei etc.)

2.3 Atribute definite de către proiectant

Modul de utilizare a acestor atribute poate fi rezumat astfel:

1. Un atribut care nu este predefinit trebuie în primul rând să fie declarat, fapt care permite stabilirea tipului atributului. Sintaxa este următoarea:

```
attribute nume_atribut: tip_atribut;
```

Iată câteva exemple:

```
type IMPLEMENTARE is
      record
            NUMĂR_FIR: INTEGER;
            NUMĂR_PLACĂ: INTEGER;
      end record;
attribute LOCALIZARE: IMPLEMENTARE;
attribute NUMĂR_MULȚIME: INTEGER;
type CAPACITY is range 0 to 1E15
      units
            ff;               -- femtofarad
            pf = 1000 ff;     -- picofarad
            nf = 1000 pf;     -- nanofarad
            uf = 1000 nf;     -- microfarad
            mf = 1000 uf;     -- milifarad
      end units;
attribute CAPACITATE: CAPACITY;
```

2. Atributul trebuie să fie specificat, ceea ce-i conferă o valoare. Aceasta va permite specificarea elementelor la care se referă atributul şi, pentru fiecare dintre aceste elemente, valoarea atributului. O specificaţie de atribut nu se referă decât la elementele declarate în aceeaşi parte declarativă ca şi specificaţia în sine.

Un atribut care nu este predefinit nu poate fi decât o constantă, cu alte cuvinte, el este static. Sintaxa este următoarea:

```
attribute nume_atribut of obiect_la_care_se_referă is
      expresie;
```

Un atribut definit de către proiectant se poate raporta la următoarele elemente VHDL: entităţi, arhitecturi, configuraţii, proceduri, funcţii, pachete, tipuri, sub-tipuri, constante, semnale, variabile, componente şi etichete. Pentru a le desemna, se vor folosi, respectiv, următoarele cuvinte cheie: **entity**, **architecture**, **configuration**, **procedure**, **function**, **package**, **type**, **subtype**, **constant**, **signal**, **variable**, **component** şi **label**.

Un atribut se raportează la anumite elemente, dar el se mai poate raporta şi la toată clasa de elemente (toate entităţile) sau numai la unele dintre ele, în mod obligatoriu numite. Această informaţie va fi dată în cadrul câmpului desemnat mai sus drept „obiect_la_care_se_referă”. Acest câmp are următoarea sintaxă:

```
nume_element {, nume_element}: clasă_element
```

sau, dacă dorim să precizăm valoarea atributului pentru toate celelalte elemente ale unei clase:

```
others: clasă_element
```

sau, dacă dorim, la modul global, să asociem o valoare a atributului pentru o întreagă clasă de elemente:

```
all: clasă_element
```

Expresia dată în sintaxa atributului trebuie, în mod evident, să returneze o valoare compatibilă cu tipul atributului definit în declaraţia atributului. Reluând ultimele exemple, iată câteva specificaţii posibile:

```
attribute NUMĂR_MULŢIME of DATA1: signal is 10;
attribute NUMĂR_MULŢIME of DATA2: signal is 11;
attribute NUMĂR_MULŢIME of others: signal is 0;
attribute CAPACITATE of all: signal is 10 pf;
```

Semnalele DATA1 şi DATA2, în mod obligatoriu declarate în aceeaşi zonă declarativă ca şi această specificaţie, vor avea asignate pentru atributele lor NUMĂR_MULŢIME valorile întregi 10 şi respectiv 11. Orice alt semnal declarat în aceeaşi zonă declarativă ca şi această specificaţie va avea valoarea întreagă 0 ca atribut NUMĂR_MULŢIME.

Orice semnal declarat în aceeaşi zonă declarativă ca şi această specificaţie va avea un atribut CAPACITATE de valoare 10 pf .

3. Atributul este folosit cu ajutorul notaţiei introduse de către apostrof după obiectul (în sensul larg) la care se referă atributul. Sintaxa este următoarea:

```
obiect_la_care_se_referă'nume_atribut
```

Expresia DATA1'NUMĂR_MULŢIME va avea deci valoarea întreagă 10. Expresia DATA2'CAPACITATE va avea deci valoarea 10 pf.

3. Desfăşurarea lucrării

3.1 Se vor testa toate atributele prezentate pe cadrul de exemple din lucrare. Se vor utiliza atributele predefinite pentru descrierea comportamentului unui bistabil JK Master-Slave.

3.2 Se va defini atributul NUMĂR_APARIŢII care să acţioneze asupra unui obiect de tip TIME.

3.3 Se va testa următorul exemplu de utilizare a atributului LAST_EVENT:

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity BISTABIL_D is
        generic(SETUP_TIME: TIME:= 1 ns; HOLD_TIME: TIME:= 2 ns);
        port(D, CLK: in STD_LOGIC;
                Q: out STD_LOGIC);
begin -- Verificarea respectării timpului de prepoziţionare (set-up)
VERIF_SETUP: process(CLK)
        begin
         if (CLK = '1') and (CLK'EVENT) then
          assert (D'LAST_EVENT >= SETUP_TIME)
          report "Încălcarea condiţiei de timp de set-up" severity ERROR;
        end if;
end process VERIF_SETUP;
end BISTABIL_D;

architecture COMPORTAMENTALĂ of BISTABIL_D is
begin
process(CLK)
        begin
         if (CLK = '1') and (CLK'EVENT) then Q <= D after 10 ns;
         end if;
end process;
end COMPORTAMENTALĂ;
```