

# **LIMBAJUL VHDL - 2**



# UNITĂȚI DE PROIECTARE

---

## Unități de proiectare primare

- entitate (interfața sistemului)
  - specificație de pachet (vedere externă a posibilităților puse la dispoziție)
  - configurație (asociere componentă - model)
-

# UNITĂȚI DE PROIECTARE

---

## Unități de proiectare secundare

- arhitectură (descrierea sistemului)
- corp de pachet (descrierea internă a funcționalităților)



# UNITĂȚI DE PROIECTARE

---

## Entitate

**entity** nume\_entitate **is**

{**generic** (listă de parametri generici)}

{**port** (listă de porturi)}

{**begin**

listă de instrucțiuni concurente}

**end** {nume\_entitate}

---

# UNITĂȚI DE PROIECTARE

---

## Entitate

- **numele entității** - **unic** în biblioteca respectivă
- **parametri generici** - pentru a **reutiliza** entitățile
- **port** - informații pentru semnale de interfață (nume, mod, tip, valori inițiale)
- **mod** - cuvinte rezervate - specifică **direcția** semnalelor
- mod: **in, out, inout, buffer, linkage**

# UNITĂȚI DE PROIECTARE

---

## Arhitectură

```
architecture nume_arhitectură of nume_entitate is  
    ... Zona de declarații (tipuri, semnale, constante,  
    funcții, proceduri, componente)  
begin  
    ... Instrucțiuni concurente  
end {nume_arhitectură}
```

---



# UNITĂȚI DE PROIECTARE

---

## Arhitectură

- tipuri de descriere:
  - structurală - interconectare de alte cutii negre
  - comportamentală - funcțională
  - flux de date - descriere algoritmică
  - hibridă - combinații între primele 3
- la o entitate - mai multe arhitecturi posibile
- **Observație** - entitatea și arhitectura trebuie să se găsească în aceeași bibliotecă

# UNITĂȚI DE PROIECTARE

---

## Arhitectură

- **nume\_entitate** trebuie să corespundă cu numele dat entității
- arhitectura face parte din **domeniul concurent**  $\Rightarrow$  nu se admit declarații de variabile
- funcționalitatea descrisă de instrucțiuni concurente care se **execută asincron**



# UNITĂȚI DE PROIECTARE

---

## Specificație de pachet

**package** nume\_pachet **is**

definiții;

. . .       -- conținutul pachetului;

declarații;

**end** nume\_pachet;

# UNITĂȚI DE PROIECTARE

---

## Corp de pachet

```
package body nume_pachet is  
{declarații interne}  
...      -- subprograme;  
end nume_pachet;
```

# UNITĂȚI DE PROIECTARE

---

## Corp de pachet

- conține **algoritmii** (strict secvențiali) pentru subprograme
- face parte din **domeniul secvențial** - nu se pot declara semnale



# UNITĂȚI DE PROIECTARE

## Configurație

```
configuration nume_configurație of nume_entitate is  
    {Zona declarativă (numai clauza use și specificarea  
    atributelor)}  
    {Zona rezervată configurației}  
end {nume_configurație}  
for eticheta_instanței_componentei :  
    nume_componentă use entity nume_entitate  
    (numele_arhitecturii){parametrii generici}  
    {corespondența porturi formale / porturi actuale};  
end for;
```

# UNITĂȚI DE PROIECTARE

---

## Configurație

- descrie **corespondența** dintre componente (declarată în arhitecturi structurale sau hibride) și arhitecturi precizate pentru entitate

## Comunicație

- procesul comunicării - implică **transmiterea informației**: sursă - destinație
  - **semnal** - purtător de informație
  - în general - semnal = fenomen fizic care se poate modifica în timp și/sau în spațiu, iar modificările se pot specifica prin instrucțiuni formale
  - semnale: electrice, mecanice, acustice, optice ...
-



## Clasificarea semnalelor (și în VHDL)

- **externe** - purtătoare de informație între dispozitive
  - reprezintă interfața
  - în VHDL se declară **numai în entitate**
- **interne** - purtătoare de informație în interiorul dispozitivelor
  - nu sunt vizibile
  - în VHDL se declară **numai în arhitecturi**

## Semnale electrice

- rol esențial în orice dispozitiv electronic
- permit **analizarea relațiilor temporale**
- în VHDL semnalele conțin și informații prezente și viitoare (istoria - history)
- linii de semnal:
  - singulare (de ex.: Clock) - o sigură valoare binară
  - multiple (magistrale) - combinație de valori binare (vectori de biți)

## Semnale în VHDL

- în VHDL semnalul = corespunde reprezentării hardware a conceptului de purtător de informație
  - **reprezentarea** = structură de date simplă sau complexă, funcție de tipul datelor purtate de semnal
  - **declarațiile** de semnal - în domeniul concurent (entitate și arhitectură)
-



## Semnale în VHDL

- acces la valori trecute, prezente și viitoare - prin **pilot (driver)** de semnal
- se memorează evenimentele care indică o **schimbare de valoare** la un moment de timp bine definit
- pot fi **modificate numai valorile** (evenimentele) **viitoare**

## Semnale în VHDL

- operația de **atribuire a unei valori**:
  - prin conectare la un port de ieșire a unei componente
  - în domeniul concurent (corespunde descrierii flux de date)
  - în domeniul secvențial

# SEMNALE

---

## Declararea semnalelor

### ■ semnale externe

- port - canal de comunicare dinamică între o entitate (sau un bloc) și mediul înconjurător

#### ■ caracteristici:

- nume
- mod - sensul fluxului de informație
- tip
- eventual valoare inițială

### ■ semnale interne

- cuvânt cheie **signal**
  - fără declarație de mod
-



# SEMNALE

---

## Vizibilitatea semnalelor

- determinată de **locul declarației**
  - **reguli:**
    - semnal declarat în pachet - văzut de unitățile de proiectare care utilizează pachetul
    - orice port - văzut în toate arhitecturile entității
    - semnal declarat în zona de declarații a arhitecturii - văzut numai în arhitectura respectivă
    - semnal declarat într-un bloc din arhitectură - văzut doar în acel bloc
-

## Asignarea semnalelor

- instrucțiunile de asignare de valori modifică valoarea viitoare (modifică piloții)
- **element de formă de undă** - o pereche valoare + **after** + întârziere
- valoare
  - tip compatibil cu semnalul
  - poate fi:
    - constantă
    - rezultatul unei expresii

## Asignarea semnalelor

### ■ întârzieri

- obligatoriu de tip Time
- apar obligatoriu în ordine crescătoare

### ■ întârziere nulă

- nu există dispozitive fizice care nu au timpi de propagare a semnalelor electrice (întârzieri)
  - întârziere delta - reprezintă o cauzalitate - este o întârziere nulă pentru simulare
-



## Asignarea semnalelor

- 2 dimensiuni ale timpului

- timp real

- văzut de proiectant
- se măsoară în pași de simulare
- folosește unități de timp Time

- timp delta

- gestionat de simulator
  - în fiecare pas de simulare se alocă felii de timp infinitezimal pt. a gestiona succesiunea asignărilor
  - exprimarea cauzalității generate de succesiunea asignărilor
-

## Asignarea semnalelor

- modele de transmisie
  - inerțial
    - filtrare impulsuri mai mici decât timpul de transmisie
    - modul implicit
  - transport
    - cuvânt cheie: **transport**
    - transmiterea oricărui impuls, indiferent de durata sa

# PARAMETRI GENERICI

---

## Scop

- transmiterea unei **informații statice** unui bloc
- blocuri generice = blocuri parametrizate
- în interiorul blocurilor
  - văzuți ca și constante
  - manipulați ca și constante



# PARAMETRI GENERICI

---

## Blocuri generice

- entitate
    - se poate compila
    - se găsește în bibliotecă
    - perechea entitate/arhitectură nu se poate simula
  - bloc intern declarat cu **block**
    - nu poate fi instanțiat din exteriorul arhitecturii în care se găsește și nici din interior
    - se poate instanția doar în momentul declarării
-

# PARAMETRI GENERICI

---

## Parametri declarați generici

- dimensiuni de obiecte complexe (vectori, magistrale ...)
  - iteratori pt. bucle **for**
  - parametri de temporizare:
    - întârzieri
    - timp de setup
    - timp de hold
    - timpi de comutare
-

# PARAMETRI GENERICI

## Sintaxa

```
generic (parametru1 {, alt_parametru} : tip_parametru  
{:= valoare_implicită});  
  
parametru2 {, alt_parametru} : tip_parametru  
{:= valoare_implicită};  
  
.....  
  
parametruN {, alt_parametru} : tip_parametru  
{:= valoare_implicită});
```



# PARAMETRI GENERICI

## Exemplu

```
POARTA_ȘI_NU: block
generic (nr_intrări: Natural := 3);
port (intrări: in Bit_Vector (1 to nr_intrări);
      ieșire: out Bit);
generic map (nr_intrări => 4); -- Instanțierea unei porți ȘI-NU cu 4 intrări
begin
--Zona de instrucțiuni din cadrul blocului
process (intrări)
variable V: Bit := '1';
begin
for I in 1 to nr_intrări loop
    V := V nand intrări(I);
end loop;
ieșire <= V;
end process;
end POARTA_ȘI_NU;
```

# CONSTANTE

---

## Scop

- informație statică declarată în interiorul modelului → arhitectură
  - valoare de inițializare care nu mai poate fi modificată
  - tipul valorii de inițializare
    - identic cu cel din declarație
    - nu poate fi acces sau fișier
  - declarare de constantă cu valoare amânată - în specificație de pachet
-

# OPERATORI

---

## Clase și priorități

- 7 clase, cu prioritate crescătoare de la 1 la 7:
  - 1. operatori logici: and, or, nand, nor, xor, xnor
  - 2. operatori relaționali: =, /=, <, <=, >, >=
  - 3. operatori de deplasare: sll, slr, sla, sra, rol, ror
  - 4. operatori de adunare: +, -, &
  - 5. operatori de semn: +, -
  - 6. operatori de înmulțire: \*, /, mod, rem
  - 7. operatori diverși: \*\*, abs, not



# OPERATORI

---

## Caracteristici

- operatorii logici
    - **predefiniți** pentru realizarea operațiilor logice: ȘI, SAU, ȘI-NU, SAU-NU, SAU-EXCLUSIV, COINCIDENȚĂ
    - operanzi de tip Boolean (False, True) și Bit ('0', '1')
    - funcționali pe vectori de elemente de tip Boolean sau Bit, dacă au aceeași lungime
-

# OPERATORI

---

## Caracteristici

- operatorii relaționali
  - rezultat de tip Boolean (False, True)
  - = și /= nu sunt definiți pt. tip fișier
  - la tipurile enumerate, primul element este considerat cel mai mic
    - exemplu: la tipul Boolean, False e mai mic decât True

# OPERATORI

---

## Caracteristici

- operatorii de deplasare - binari
  - operează pe vectori cu elemente de tip Bit sau Boolean
- operatorii de adunare - binari
  - & - definit pe vectori (tipul String = vector de caractere)
- operatorii de semn - unari
  - au prioritate mai mică decât înmulțirea → utilizarea parantezelor pt. evitarea erorilor



# OPERATORI

---

## Caracteristici

- operatori de înmulțire - binari
- operatori diverși
  - **\*\*** - ridicare la putere
    - operandul din stânga de tip întreg sau flotant
    - puterea - obligatoriu tip întreg
  - **abs** - pe orice tip numeric
  - **not**
    - operator logic, unar
    - operează pe obiecte de tip Boolean și Bit și pe vectori de astfel de elemente

# TIPURI DE DATE

## Generalități

- VHDL puternic **tipizat**
  - fiecare semnal, variabilă, constantă are un tip (definit înainte de utilizare)
  - parametrii procedurilor și funcțiilor și rezultatul returnat de funcții - au obligatoriu un tip
- tipizarea obiectelor - protejează instrucțiunile de atribuire
- nivel de abstractizare relativ la implementarea structurilor de date → prin asociere de reprezentare simbolică independentă de partea hardware

# TIPURI DE DATE

---

## Tipuri de date

- 4 tipuri de date:
  - scalare - valoarea constituită dintr-un element
  - compuse - valoarea constituită din mai multe elemente
  - acces (pointeri)
  - fișier



# TIPURI DE DATE

## Clase de obiecte - familii de tipuri

	Constante	Variabile	Semnale	Fișiere
Scalare	DA	DA	DA	NU
Compuse	DA	DA	DA	NU
Acces	NU	DA	NU	NU
Fișier	NU	NU	NU	DA

# TIPURI DE DATE

---

## Tipuri scalare

- 4 tipuri: enumerate, întregi, flotante, fizice
- ordonate (pot fi comparate)
- interval de validitate - restrânge valorile posibile
  - **range** expresie 1 **to** expresie2;
  - **range** expresie 3 **downto** expresie4;

# TIPURI DE DATE

## Tipuri scalare enumerate

- **type** Nume **is** (valoare\_simbolică1, valoare simbolică2, ...);
- simbolurile: identificatori sau caractere
- predefinite în pachetul Standard:
  - **type** Boolean **is** (False, True);
  - **type** Bit **is** ('0', '1');
  - **type** Severity\_Level **is** (Note, Warning, Error, Failure);
  - Character - toate caracterele admise în VHDL
- poziția - induce relația de ordine între elemente



# TIPURI DE DATE

---

## Tipuri scalare întregi

- sunt definiți operatorii aritmetici
  - în declarație se indică domeniul (**range**)
    - exemplu: **type** Nume **is range** 1 to 15;
  - tipul Integer predefinit în pachetul Standard
  - tipurile Positive și Natural - subtipuri ale Integer
  - conversie implicită în Universal\_Integer - tip virtual
-

# TIPURI DE DATE

---

## Tipuri scalare flotante

- sunt definiți operatorii aritmetici
  - în declarație se indică domeniul (**range**)
  - tipul Real predefinit în pachetul Standard
  - intervalul domeniului - cu precizie de minimum 6 cifre după virgulă
  - conversie implicită în Universal\_Real - tip virtual
-

# TIPURI DE DATE

## Tipuri scalare fizice

- în VHDL - noțiune de unitate de cantitate
- caracteristici:
  - unitatea de bază
  - intervalul valorilor autorizate
  - eventual colecție de subunități cu corespondențele lor
- sunt definiți operatorii aritmetici
- între număr și unitatea de măsură - spațiu!!!
- toate valorile unui tip fizic și toate valorile de conversie trebuie să fie numere întregi



# TIPURI DE DATE

---

## Tipuri scalare fizice

- **Time** - singurul tip fizic predefinit în pachetul Standard → tip utilizat de simulator
  - unitatea de bază - femptosecunda
  - definit de un interval cu capetele exprimate pe 64 biți
  - unități: fs, ps, ns, ms, sec, min, hr

# TIPURI DE DATE

---

## Tipuri compuse

- 2 tipuri compuse:
  - tablouri - colecție de obiecte de același tip
  - articole - colecție de obiecte de tipuri diferite

# TIPURI DE DATE

---

## Tipuri compuse

- tablouri
    - structuri omogene
    - elementele accesibile pe baza unor indecși
    - definirea:
      - specificarea tipului
      - indicarea numărului indecșilor (tip discret)
      - specificarea tipului elementelor (fără tipul fișier)
    - vector - tablou cu un singur index
-



# TIPURI DE DATE

## Tipuri compuse

### ■ tablouri

#### ■ constrânse




- intervalul de variație al indecșilor cunoscut cu anticipație
- sensul de variație a indecșilor - specificat cu **to** și **downto**
- exemple:
  - **type** Adresă **is array** (0 **to** 15) **of** Bit;
  - **type** Word **is array** (31 **downto** 0) **of** Bit;
  - **type** Memory **is array** (Adresă) **of** Word;

# TIPURI DE DATE

## Tipuri compuse

### ■ tablouri

#### ■ neconstrânse

- intervalul de variație al indecșilor cunoscut numai în timpul simulării
-  (box) amână definirea intervalului de indexare și a direcției de variație
- 2 tablouri neconstrânse în pachetul Standard:
  - **type** Bit\_vector **is array** (Natural range ) **of** Bit;
  - **type** String **is array** (Positive range ) **of** Character;

# TIPURI DE DATE

---

## Tipuri compuse

- articole
  - elemente (câmpuri) diferite
  - enumerare câmpuri între **record ... end record**
  - selectarea prin notația cu punct



# TIPURI DE DATE

---

## Tipuri compuse

- indicarea valorilor - notația prin agregare
  - asociere pozițională
    - contează ordinea în lista elementelor
  - asociere prin denumire
    - nume => valoare
    - ordinea câmpurilor nu contează
  - asociere mixtă
    - începe cu partea pozițională
  - pentru inițializare se folosește **“others”**

# TIPURI DE DATE

---

## Tipuri acces (pointeri)

- indică spre obiecte definite anterior
- pt. crearea dinamică de obiecte → alocarea dinamică a memoriei
- alocarea - **new**; dezalocarea - **delete**
- la declarare - inițializare cu valoarea implicită **null**

# TIPURI DE DATE

---

## Tipuri fișier

- fișiere cu acces secvențial
- la declarare trebuie precizat tipul de date din fișier
- 3 subprograme create automat la declararea de tip fișier: Read, Write, Endfile



# TIPURI DE DATE

---

## Conversii de tip

- conversia - **foarte restrictivă**
- posibilă în 3 cazuri:
  - tipuri cu reprezentare întreagă sau flotantă
  - pentru tablouri
    - aceleași dimensiuni
    - aceleași tipuri de elemente
    - indecșii trebuie să fie convertibili
  - conversia unui tip în el însuși

## Generalități

- caracteristică asociată unui tip sau unui obiect, care poate fi cunoscută în mod dinamic, în timpul rulării
- notație - adăugarea unui **apostrof** după numele tipului sau obiectului
- attribute:
  - predefinite
  - definite de proiectant

# ATTRIBUTE

---

## Attribute predefinite

- pot fi: valori (tipizate), funcții, tipuri, intervale de variație
  - se aplică unor prefixuri care pot fi: valori, tipuri, etichete
  - simplifică scrierea
  - apar în funcții utilitare
-



## Attribute predefinite

- attribute definite pe tipuri
  - T desemnează un tip (obiectul asupra căruia acționează atributul), prefix al atributului
  - `obiect'nume_atribut(parametri)`

# ATTRIBUTE

## Attribute predefined

- attribute definite pe tipuri
  - tip sau subtip **scalar** T; X = tip scalar

■ Atribut	Rezultat
T'left	Limita la stânga a lui T
T'right	Limita la dreapta a lui T
T'low	Limita inferioară a lui T
T'high	Limita superioară a lui T
T'base	Tipul de bază a lui T
T'image(X)	Șirul X
T'value(X)	Valoare de tip T

# ATTRIBUTE

## Attribute predefined

- attribute definite pe tipuri
  - tip sau subtip **discret** sau **fizic** T; X membru al lui T; N număr întreg

■ Atribut	Rezultat
T'pos(X)	Numărul poziției lui X în T
T'val(N)	Valoarea la poziția N în T
T'leftof(X)	Valoarea în T, cu o poziție în stânga lui X
T'rightof(X)	Valoarea în T, cu o poziție în dreapta lui X
T'pred(X)	Valoarea în T, cu o poziție mai mică decât X
T'succ(X)	Valoarea în T, cu o poziție mai mare decât X
T'ascending	Valoare booleană pt. interval crescător sau descrescător



# ATTRIBUTE

## Attribute predefined

- attribute definite pe tipuri sau subtipuri **tablou**
  - $A = \text{tablou}$ ;  $N = \text{număr întreg între } 1 \text{ și numărul dimensiunilor lui } A$

■ Atribut	Rezultat
-----------	----------

$A'_{\text{left}}(N)$	Limita stânga a domeniului indicelui dimensiunii $N$ a lui $A$
$A'_{\text{right}}(N)$	Limita dreapta a domeniului indicelui dimensiunii $N$ a lui $A$
$A'_{\text{low}}(N)$	Limita inferioară a domeniului indicelui dimensiunii $N$ a lui $A$
$A'_{\text{high}}(N)$	Limita superioară a domeniului indicelui dimensiunii $N$ a lui $A$
$A'_{\text{range}}(N)$	Domeniul indicelui dimensiunii $N$ a lui $A$
$A'_{\text{reverse\_range}}(N)$	Inversul domeniului indicelui dimensiunii $N$ a lui $A$
$A'_{\text{length}}(N)$	Lungimea domeniului indicelui dimensiunii $N$ a lui $A$
$A'_{\text{ascending}}(N)$	Valoare booleană pentru direcția indicelui dimensiunii $N$ a lui $A$

# ATTRIBUTE

## Attribute predefined

- attribute definite pe semnale S
  - attribute semnal

■ Atribut	Rezultat
-----------	----------

S'delayed(T)	Semnal S întârziat cu T unități de timp
S'stable(T)	Valoare booleană True dacă S e fără evenimente în T
S'quiet(T)	Valoare booleană True dacă S e inactiv în T
S'transaction	Modificare valoare Bit de câte ori S este activ

# ATTRIBUTE

## Attribute predefinite

- attribute definite pe semnale S
  - attribute funcție

■ Atribut	Rezultat
-----------	----------

S'event	Valoare booleană True pt. eveniment pe S
S'active	Valoare booleană True dacă S e activ
S'last_event	Timpul trecut de la ultimul eveniment pe S (valoare Time)
S'last_active	Timpul trecut de la ultima activare a lui S (valoare Time)
S'last_value	S imediat înainte de ultima modificare
S'driving_value	Permite o operație de atribuire
S'driving	Valoare booleană dacă S nu este deconectat



# ATTRIBUTE

---

## Atribute predefinite

- atribute definite pe obiecte în sens larg X
  - utilizate pentru elaborare de mesaje

### ■ Atribut

### Rezultat

X'simple\_name

Numele X

X'path\_name

Numele X și etichetele de revenire la X

X'instance\_name

Numele X, etichetele de revenire la X,  
informații de configurare

# ATTRIBUTE

---

## Attribute definite de utilizator

- declarare atribut
  - **attribute** nume\_atribut: tip\_atribut;
- specificare atribut (primește valoare)
  - **attribute** nume\_atribut **of** obiect **is** expresie;
  - nu pot fi decât constante, deci sunt statice
- utilizare atribut - cu notația cu apostrof
  - obiect'nume\_atribut

# ATTRIBUTE

## Atribute definite de utilizator

- se pot raporta la:
  - entități, arhitecturi, configurații, pachete, proceduri, funcții, tipuri, subtipuri, constante, semnale, variabile, componente, etichete
- raportarea se poate face:
  - la anumite elemente, care trebuie numite în câmpul “obiect”
    - `nume_element`, `{nume_element}`: clasă\_element
    - pentru toate celelalte elemente ale unei clase:
      - **others**: clasă\_element
  - la toată clasa de elemente
    - **all**.clasă\_elemente