Introduzione

- Il VHDL e' costituito da vari formati (types)ed operatori (operators) per consentire simulazione e sintesi a vari livelli
- Nel package STANDARD si trovano descritti quegli oggetti destinati alla descrizione COMPORTAMENTALE (non sempre sintetizzabile)
- Nel package IEEE1164 vi si trovano gli oggetti destinati alla sintesi ed alla simulazione logica
- Il VHDL e' un linguaggio fortemente basato sulla sintassi

Sintassi

- Le varie espressioni sintattiche scritte in VHDL si possono ricondurre ai seguenti oggetti:
  - Scalari e Vettori
  - Nomi
  - Oggetti:
    - Costanti
    - Segnali
    - Variabili
  - Espressioni

Sintassi - Scalari e Vettori

Scalari	Vettori
character	string
bit	bit_vector
std_logic	std_logic_vector
boolean	
real	
integer	
time	

Sintassi - Scalari e Vettori

Characters

- Un “character” va dichiarato racchiuso tra virgolette singole
  - Es: ‘a’ ‘A’ ’#’ ‘@’ ‘ ‘ ‘ ’ ’
- a meno di caratteri ASCII particolari
  - Es: CR DEL NUL ACK BEL LF

Sintassi - Scalari e Vettori

Strings

- Una “string” e’ un array di caratteri e va dichiarata racchiusa tra virgolette doppie
  - Es: “ciao a tutti”“x”
- In caso di equivoco si usi la dichiarazione esplicita
  - Es: string’(“100100”)

Sintassi - Scalari e Vettori

Bit

- Il BIT assume solo valori ‘0’ o ‘1’ e va dichiarato tra virgolette singole
  - Es: ‘0’ ‘1’
- In caso di equivoco si usi la dichiarazione esplicita
  - Es: bit’(‘0’) bit’(‘1’)

Sintassi - Scalari e Vettori

Bit_vector

- Il Bit_vector e' un array di Bit che assumono solo valori '0' o '1' e va dichiarato tra virgolette doppie, e' comunque consentito adottare una notaziono ottale o esagesimale. IL carattere '_' puo' essere adottato per comodita', ma non viene interpretato
  - Es: "0001_1001" x"00FF"
- In caso di equivoco si usi la dichiarazione esplicita
  - Es: bit_vector'("0110_0101_0011")

Sintassi - Scalari e Vettori

STD_logic

- E’ il “type” piu’ usato per la sintesi logica
- Assume i valori:

'U'	uninitialized		
'X'	unknown	'W'	weak unknown
'0'	0 logic	'L'	weak 0
'1'	1 logic	'H'	weak 1
'Z'	high impedence	'-'	don't care

Sintassi - Scalari e Vettori

STD_logic

- Viene dichiarato racchiuso tra virgolette singole
  - Es: ‘U’ ‘X’ ‘1’ ‘0’
- In caso di equivoco si usi la dichiarazione esplicita
  - Es: std_logic’(‘1’)

Sintassi - Scalari e Vettori

STD_logic_vector

- Viene dichiarato racchiuso tra virgolette doppie
  - Es: “001XX” “UUUU”
- In caso si voglia esprimere un particolare valore espresso secondo una notazione di tipo “unsigned” o “signed” (complemento a 2) si deve impiegare il pakage STD_LOGIC_ARITH
  - Es: signed’(“111001”) (ossia -7)
    unsigned(“111001”) (ossia 57)

```
Library IEEE;
Use IEEE.STD_LOGIC_1164.all;
Use IEEE.STD_LOGIC_ARITH.all;
```

Sintassi - Scalari e Vettori

Boolean

- Assume due soli valori in genere deriva da un operatore che esprime una “relazione” (= <= >= /=)
  ed e’ solitamente impiegato in un test.
- Valori consenti: True, False
  - Es: true TRUE True
    false FALSE False

Sintassi - Scalari e Vettori

Real

- Puo' essere utile per simulazioni ad alto livello
- **NON VIENE SINTETIZZATO**
- DEVE contenere il punto decimale ed eventualmente il segno
  - Es: 1.0 +2.23 - 4.56 -1.0E+38
- Per impiegare un array di numeri reali deve essere opportunamente dichiarato

Sintassi - Scalari e Vettori

Integer

- Puo' essere utile per simulazioni ad alto livello
- **NON SEMPRE VIENE SINTETIZZATO**
- NON DEVE contenere il punto decimale ma puo' eventualmente contenere il segno
  - Es: 10 +223 - 456
- Un intero puo' eventualmente essere espresso in una latra base
  - Es: 16#00F0F#
- Nel pakage STANDARD sono descritti due subset degli Integer: positive e natural

Sintassi - Scalari e Vettori

Time

- E’ la sola grandezza fisica predefinita in VHDL.
- E’ definita nel Pakage STANDARD
- E’ importante separare il valore dall’unita’ di grandezza
  - Es: 10 ns 123 us 6.3 sec
- Unita’ di grandezza consentite:
  fs ps ns us ms sec min hr

Sintassi - Ulteriori tipi

type, subtype

- In VHDL si possono “inventare” delle variabili “su misura

```
TYPE mese IS (gennaio, febbraio, giugno);

TYPE bit IS (‘0’, ‘1’);
```

- E dei sottoinsiemi di queste”

```
SUBTYPE mesefreddo IS

mese range gennaio to febbraio;
```

Sintassi - Nomi

- Ogni oggetto (entity, architerctures, segnali, …) ha un nome simbolico
- Il VHDL e' un linguaggio "Case insentistive" (ossia abcd e' analogo a AbCd)
- Vi sono "nomi riservati" quali:
  in, out, signal, port, library, map, entity, …
  .
- I nomi "relativi" vengono indicati con un "." nella sintassi
  - Es: libray_name.pakage_name.item.neme
    WORK.my_defs.unit_delay

Sintassi - Dichiarazioni di oggetti

- In VHDL vi sono grandezze che mantengono il loro valore immutabile ed altre che possono cambiare valore
  - constants: gandezze fisse
  - signals: rappresentano collegamenti fisici (sono grandezze concorrenti)
  - variables: rappresentano variabili all'interno di un processo (sono grandezze sequenziali)
- Inoltre si possono definire dei puntatori a files (ovviamente per blocchi puramente comportamentali)
- Ogni grandezza impiegata deve essere definita a priori

```
variable x: integer;
signal aBc: bit;
constant Vdd: real := 12.3;
```

Sintassi - Range

- Si puo' vincolare una grandezza a rimanere all'interno di un certo campo di variablita' ("range")
- Il range va specificato in fase di dichiarazione

```
entity COMPARE_digit is
      port (a, b : in integer range 1 to 10;
            c : out boolean);
end COMPARE_digit;

variable ABC: real range 1.0 to 10.0;
```

Sintassi - Costants

- Risultano comode quando in piu' parti del listato si fa riferimento alla stessa grandezza costante
- Le costanti possono essere dichiarate all'interno di un pakage, entity o architecture

```
constant Vdd: Real := 4.5;
constant CYCLE : Time := 100 ns;
constant PI : Real := 3.14;
constant FIVE : std_logic_vector (0 to 3) := "0101";
```

Sintassi - Signals

- Sono l'astrazione dei "collegamenti fisici"
- Fanno comunicare tra loro varie entity
- Un segnale puo' essere inizializzato (ATTENZIONE in fase di SINTESI l'inizializzazione potrebbe essere disattesa!)
- In un' entity un segnale viene dichiarato tramite la `port`

```
signal count: integer range 1 to 10;
signal GROUND: bit :=0 ;
signal SYS_BUS : std_logic_vector (7 downto 0);
port (A, B : in std_logic);
```

Sintassi - Variables

- Una variabile viene impiegata nei `process`
- l'assegnamento del valore ad una variabile avviene istantaneamente in simulazione (all'opposto di un segnale per cui lassegnamento avviene in base al "tempo di simulazione")
- deve essere dichiarata prima di essere usata

```
variable INDEX : integer range 1 to 50;
variable CYCLE : time range 10 ns to 50 ns := 10ns;
varaible MEMORY : bit_vector (0 to 7);
variable x,y,z : integer;
```

Sintassi - expressions

- Sono formule impiegate per calcolare un risultato
- L’operando dipende dalle grandezze usate come operatori
- Alcuni operandi risiedono in appositi pakages
- Generalmente gli operandi devono essere dello stesso tipo
- Altrimenti si deve esplicitare la conversione
  - ERRORE 1 + 1.0
  - CORRETTO1 + INTEGER (1.0)

Sintassi - Operandi

- Logici:
  `and, or, nand, nor, xor`
- Relazionali
  `=, /=, <, <=, >, >=`
- Concatenazione e aritnetici
  `&, +, -, *, /, mod, rem, **, abs`
- Logici:
  `not`

- NOTA: il precedente elenco e' ordinato in base alla priotita'

Pakages STD_logic_arith

- Il pakage STANDARD non consente comparazioni o operazioni aritmetiche tra “`bit_vector`”
- Alcuni venditori provvedono un pakage per definire le operazioni tra std_logic_vector
- Servono per definire se le grandezze impiegate sono di tipo “signed” o “unsigned”
  - ovvero ad esempio come interpretare la grandezza
    “1011” ossia 11 oppure -5 ?)

```
use IEEE.std_logic_signed.all;
use IEEE.std_logic_unsigned.all;
```