

ELEMENTOS DEL LENGUAJE

- **Identificadores**

- Sirven para dar nombres a los elementos del VHDL. Se deben seguir reglas mnemotécnicas.
- No tienen longitud máxima.
- Pueden contener los caracteres “A” a “Z” y “a” a “z”, los caracteres numéricos de 0 a 9 y el carácter subrayado “_”. VHDL no es case sensitive.
- Deben comenzar con un carácter alfabético, no pueden terminar con “_” ni contener “_”.
- No puede utilizarse como identificador a una palabra reservada.

Lenguajes de Descripción de Hardware

Palabras reservadas

- Es un conjunto de palabras que tienen significado especial en VHDL.
- Sirven para definir sentencias en los modelos.
- Existen aproximadamente 100 palabras reservadas en el lenguaje.

Lenguajes de Descripción de Hardware

Palabras reservadas

- Las más comunes son:

After	all	and	architecture	array	begin	block	body
bus	case	component	else	elsif	end	entity	for
function	generate	generic	if	in	inout	is	library
nand	next	nor	not	null	or	out	Package
port	procedure	process	signal	subtype	then	to	Until
use	variable	wait	when	while	with	configuration	loop
map	constant	xor					

Lenguajes de Descripción de Hardware

Símbolos especiales

+ - / () . , : ;

& ' < > = #

=> ** := /= >= <= <>

Lenguajes de Descripción de Hardware

OBJETOS VHDL

- Hay cuatro clases de objetos: constantes, variables, señales y ficheros.
- Todos los objetos deben declararse para poder ser utilizados.

Lenguajes de Descripción de Hardware

Constantes

- Mantienen siempre su valor inicial. No puede ser modificada una vez que ha sido creada.
- Cualquier constante podría ser substituida en el código por un literal con su valor.
- Se recomienda utilizar constantes para mejorar la legibilidad del código.
- Hacen más fáciles las actualizaciones del código, ya que es más fácil cambiar la definición de la constante que seguir el código y reemplazar valores.
- Algunos ejemplos son:

```
Constant pi:real:= 3.141591 ;  
Constant bitspalabra:integer:= 8 ;  
Constant numeropalabras: integer:=64;  
Constant numerobits: integer:=  
bitspalabra*numeropalabras ;
```

Lenguajes de Descripción de Hardware

Variables

- Las variables pueden cambiar su valor una vez que han sido creadas.
- Para realizar esto se utilizan las sentencias de asignación.
- No tienen ninguna analogía con el hardware real.
- Se utilizan cuando se modela en el estilo algorítmico para almacenar resultados parciales.
- Algunos ejemplos de declaración:

```
Variable indice1, indice2, indice3: integer:= 0;  
Variable comparacion: boolean;
```

- Para modificar una variable se utiliza la sintaxis:
Identificador := expresión ;

Lenguajes de Descripción de Hardware

Variables

Deben tenerse en cuenta los siguientes aspectos:

- La asignación es inmediata. Se realiza después de haberse evaluado la sentencia de asignación.
- Si la sentencia siguiente utiliza el valor de la variable, se utilizará el valor nuevo.
- Normalmente se declaran en la parte declarativa de los procesos.
- Son visible sólo dentro de los procesos.
- Se evita de esta forma que cuando un proceso modifique el valor de una variable se afecte a otro proceso ejecutándose concurrentemente.

Lenguajes de Descripción de Hardware

Señales

- Pueden considerarse como una abstracción de una conexión física o un bus.
- Se utilizan para interconectar componentes de un circuito y para sincronizar la ejecución y suspensión de procesos.
- No están restringidas a un proceso.
- Se declaran en la arquitectura del dispositivo.
- Los puertos son señales especiales que se utilizan para interconectar dispositivos.
- La señal puede modificarse mediante las sentencias de asignación a señales.
- Su valor no cambia de inmediato. Lo hace luego de ser concluidos todos los procesos activos.

Lenguajes de Descripción de Hardware

Señales

- Algunos ejemplos de declaración de señales son:

```
signal reloj:std_logic:= '0';  
signal comparacion: bit;  
port(a,b:in integer range 0 to7;  
      c:out integer range 0 to 7);
```

Lenguajes de Descripción de Hardware

TIPOS DE DATOS

- En VHDL todas las señales deben tener un tipo de dato asociado que limita el número de valores posibles. El tipo puede ser fijado en la declaración de puertos o en la declaración de señales dentro de la arquitectura.

```
entity FULLADDER is
port ( A, B, CARRYIN : in bit ;
      SUM, CARRY      : out bit ) ;
end FULLADDER ;

architecture MIX of FULLADDER is
component HALFADDER
port ( A, B in bit ;
      SUM, CARRY : out bit ) ;
end component;
signal WSUM, WCARRY1, WCARRY2 : bit
....
```

Lenguajes de Descripción de Hardware

TIPOS DE DATOS

- Existe un cierto número de tipos ya definidos en el paquete standard, pudiendo el usuario definir sus propios tipos.

```
boolean ..... verdadero , falso
real ..... -23.45, 10, 56.34x10e-5, ....
bit ..... '0' , '1'
time ..... 40ns, 5ps , .....
string ..... "Error", "10110" , ....
character ..... "A", "H", "8", ...
integer.....2, 4, 379, ....
```

```
package STANDARD is
type BOOLEAN is ( FALSE, TRUE ) ;
type BIT is ( '0' , '1' ) ;
type INTEGER is range ....
type REAL is range ....
...
end STANDARD;
```

Lenguajes de Descripción de Hardware

TIPOS DE DATOS

- El tipo de dato time sirve para modelar retardos en compuertas. Puede ser multiplicado por un entero o un real.

```
..  
...  
...  
CLK <= not CLK after 0.025 ns  
CLK <= not CLK after PERIOD0/2  
...  
...
```

Lenguajes de Descripción de Hardware

DEFINICIÓN DE ARREGLOS (ARRAYS)

- Los arrays son útiles para agrupar señales del mismo tipo. VHDL predefine dos tipos arrays que son bit_vector y string, que son arrays de bits y caracteres respectivamente.

```
...  
...  
signal BUS_A, BUS_Z : bit_vector ( 3 downto 0 )  
;  
...  
...
```

Lenguajes de Descripción de Hardware

DEFINICIÓN DE ARREGLOS (ARRAYS)

- Ejemplos de asignación de señales con arrays

```
...  
BUS_Z <= BUS_A  
...
```

- Esta asignación significa:

```
BUS_Z(3) <<<<<< BUS_A(3)  
BUS_Z(2) <<<<<< BUS_A(2)  
BUS_Z(1) <<<<<< BUS_A(1)  
BUS_Z(0) <<<<<< BUS_A(0)
```

- En cambio en la asignación:

```
...  
BUS_Z(3) <= BUS_A(0)  
...
```

- Se asigna el bit de la posición cero de BUS_A al bit de la posición 3 de BUS_Z sin cambiar los demás.

Lenguajes de Descripción de Hardware

DEFINICIÓN DE ARREGLOS (ARRAYS)

- En el siguiente ejemplo :
architecture EJEMPLO of ARRAYS is
 signal BUS_Z : bit_vector (3 downto 0) ;
 signal BUS_A : bit_vector (0 to 3) ;
begin
 BUS_Z <= BUS_A ;
end EJEMPLO ;

- Quedan definidos los arrays de datos de la siguiente forma:

```
Z = [ z3, z2, z1, z0 ]  
A = [ a0, a1, a2, a3 ]
```

- Con la asignación de señal del ejemplo se efectúa la siguiente operación:

```
BUS_Z(3) <<<<<< BUS_A(0)  
BUS_Z(2) <<<<<< BUS_A(1)  
BUS_Z(1) <<<<<< BUS_A(2)  
BUS_Z(0) <<<<<< BUS_A(3)
```

Lenguajes de Descripción de Hardware

MODELADO CON ENTEROS O VECTORES DE BITS

- Los dos ejemplos que siguen sintetizarán el mismo circuito:

- Ejemplo 1

```
...
Architecture EJEMPL01
of TIPOS is
Signal SEL : bit
Signal A, B, Z :
integer range 0 to 3
;
Begin
A <= 2;
B <= 3;
...
```

- Ejemplo 2

```
...
Architecture EJEMPL02
of TIPOS is
Signal SEL : bit
Signal A, B, Z :
bit_vector ( 1downto
0) ;
Begin
A <= "10";
B <= "11";
...
```

Lenguajes de Descripción de Hardware

LITERALES "BIT STRING"

- Nota: los valores de bits simples son encerrados entre comillas simples. Los vectores son encerrados entre comillas dobles.
- Ejemplo:

```
architecture EJEMPLO of ASIGNACIÓN is
signal BUS_Z : bit_vector ( 3 downto 0) ;
signal BIG_BUS : bit_vector ( 15 downto 0) ;
begin
BUS_Z (3) <= '1' ;
BUS_Z <= "1100" ;
BIG_BUS <= " 0000_1111_1010_1111" ;
end EXAMPLE ;
```

Lenguajes de Descripción de Hardware

CONCATENACIÓN

- Ejemplo 1:

```
architecture EJEMPL01 of CONCATENACION is
signal BYTE: bit_vector (7 downto 0);
signal BUS_A, BUS_B : bit_vector (3 downto 0);
begin
    BYTE <= BUS_A & BUS_B ;
end EXAMPLE1 ;
```

- Ejemplo 2:

```
architecture EJEMPL02 of CONCATENACION is
signal BUS_Z : bit_vector (3 downto 0);
signal BIT_A,BIT_B ,BIT_C , BIT_D : bit ;
begin
    BUS_Z <= BIT_A & BIT_B & BIT_C & BIT_D ;
end EXAMPLE2 ;
```

Lenguajes de Descripción de Hardware

CLASIFICACIÓN DE TIPOS

- **TIPOS ESCALARES** : contienen exactamente un valor. Pueden ser: INTEGER, REAL, BIT, ENUMERATED Y PHYSICAL.
- **TIPOS COMPUESTOS**: Pueden contener mas de un valor. Son ARRAYS cuando contienen datos del mismo tipo o RECORD cuando tienen datos de diferentes tipos.
- **OTROS TIPOS**: FILE, ACCESS
- **TIPOS ENUMERADOS**
 - Los diseñadores pueden crear sus propios tipos de datos para facilitar la comprensión del código.
 - Son frecuentemente usados para la descripción de máquinas de estados, en donde los diseñadores describen cada estado con un nombre y no con un dato binario.
 - Estos nombres deben ser declarados en una lista.
 - Sólo los nombres declarados pueden ser utilizado en las asignaciones de señales.

Lenguajes de Descripción de Hardware

Ejemplo:

```
architecture EJEMPLO of ENUMERACIÓN is
  type ESTADO_T is ( RESET, START, EXECUTE,
    FINISH ) ;
  signal ESTADOACTUAL : ESTADO_T ;
  signal ESTADOPROXIMO : ESTADO_T ;
  signal VEC_DOS_BIT : bit_vector ( 1 downto 0 )
  ;
begin
  --asignacioes válidas:
  ESTADOPROXIMO <= ESTADOACTUAL
  ESTADOACTUAL <= RESET
  --asignaciones no válidas
  ESTADOACTUAL <= "00"
  ESTADOACTUAL <= VEC_DOS_BIT
end EJEMPLO ;
```

Lenguajes de Descripción de Hardware

PROBLEMAS CON EL TIPO BIT

- Puede adoptar solo los valores 0 ó 1.
 - No puede modelar buses.
 - No se pueden modelar los efectos de un cero o un uno débil a la salida de buffers.
- En muchos casos es imposible modelar y verificar el correcto reset del sistema.

Lenguajes de Descripción de Hardware

PROBLEMAS CON EL TIPO BIT

- Se crea un nuevo tipo de datos denominado TIPO LÓGICO STANDARD IEEE, con los siguientes valores:

```
type STD_ULOGIC is (  
  'U', --NO INICIALIZADO  
  'X', --UNO O CERO FUERTE (= DESCONOCIDO)  
  '0', --CERO FUERTE  
  '1', --UNO FUERTE  
  'Z', --ALTA IMPEDANCIA  
  'W', --UNO O CERO DÉBIL (= DESCONOCIDO)  
  'L', --CERO DÉBIL  
  'H', --UNO DÉBIL  
  '--', --SIN CUIDADO) ;
```

- El nuevo tipo de dato se llama `std_ulogic`, definido en el paquete `std_logic_1164` ubicado en la librería IEEE.

Lenguajes de Descripción de Hardware

CONVERSIÓN DE TIPOS

- En las operaciones de asignación de señal se debe operar con datos del mismo tipo.
- Cuando en el modelo coexisten datos de diferentes tipos, es necesario hacer una conversión de tipos previa a la asignación.
- Esto se realiza con las funciones de conversión, las que deben ser definidas por el usuario, o bien estar declaradas en los paquetes (*package*).

Lenguajes de Descripción de Hardware

CONVERSIÓN DE TIPOS

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
ENTITY adder IS
PORT (op1, op2 : IN UNSIGNED(7 DOWNTO 0);
      result    : OUT INTEGER);
END adder;

ARCHITECTURE maxp1d OF adder IS
BEGIN
    result <= CONV_INTEGER(op1 + op2);
END maxp1d;
```

Lenguajes de Descripción de Hardware

SUBTIPOS

- Se declaran cuando se desea un nuevo tipo que es una versión restringida del tipo original.
- Los subtipos siguen conservando el mismo tipo que el original. No son necesarias funciones de conversión para operarlos.

```
SUBTYPE natural IS integer RANGE 0 TO integer'high ;
SUBTYPE positive IS integer RANGE 1 TO integer'high;
```

Lenguajes de Descripción de Hardware