Tipos de datos en VHDL

Sistemas Digitales Avanzados

Universidad Técnica Particular de Loja

Prof: Diego Barragán Guerrero

Oct. 2014 - Feb. 2015

Tipos de datos pre definidos en paquetes

- Estándar: IEEE 1076 y 1164.
- Son definidos en paquetes y librerías.
- Paquete **standard** de libr. std: bit, boolen, integer y real.
- Paquete **std_logic_1164** de libr. IEEE: std_logic y std_ulogic.
- Paquete **std_logic_arith** de libr. IEEE: *signed* y *unsigned*.
  - conv_integer(p)
  - conv_unsigned(p, b)
  - conv_signed(p, b)
  - conv_std_logic_vector(p, b)
- Paquete **std_logic_signed** y **std_logic_unsigned** de libr. IEEE: permite operaciones con datos std_logic_vector considerando que sea un dato *signed* o *unsigned*, respectivamente [2].

Paquete IEEE numeric_std

Añade tipo de dato *unsigned* y *signed*. Permite operaciones aritméticas y de relación.

Código VHDL

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all; -- invocar nuevo paquete
```

Cuadro: Funciones de conversión de datos [1].

Desde	Hacia	Función de conversión
unsigned, signed	std_logic_vector	std_logic_vector(a)
signed, std_logic_vector	unsigned	unsigned(a)
unsigned, std_logic_vector	signed	signed(a)
unsigned, signed	integer	to_integer(a)
natural	unsigned	to_unsigned(a, size)
integer	signed	to_signed(a, size)

BIT y BIT_VECTOR

Dos niveles lógicos: '0' y '1'; Ejemplos:

Código VHDL

```
SIGNAL x: BIT;
-- x es declarada como una señal tipo bit (un dígito).
SIGNAL y: BIT_VECTOR (3 DOWNTO 0);
-- y es un vector de 4 bits, con el MSB correspondiente al bit más
-- a la izquierda.
SIGNAL w: BIT_VECTOR (0 TO 7);
-- w es un vector de 8 bits, con el MSB correspondiente al bit más
-- a la derecha.

x<='1';
-- Para un bit usar comilla simple.
y<="0111";
-- Para un vector de bits se usa doble comilla.
w<="01110001";
```

Std_logic (y std_logic_vector)

Cuadro: IEEE 1164

Tipo	**Std_logic**
U	Uninitialized (Sin inicializar)*
X	Forcing Unknown (Forzar valor desconocido)*
0	Forcing 0 (Forzar un cero)*
1	Forcing 1 (Forzar un uno)*
Z	High Impedance (Alta impedancia)
W	Weak Unknown (Valor débil desconocido)
L	Weak 0 (Cero débil)

* Sintetizables.

Std_logic (y std_logic_vector)

Código VHDL

```
SIGNAL x: STD_LOGIC;
-- x es declarada como señal de un dígito del tipo std_logic.

SIGNAL y: STD_LOGIC_VECTOR (3 DOWNTO 0) := "0001";
-- y es declarada como un vector de 4 bits (MSB= bit más a la izquierda).
-- El valor "0001" es opcional. Se establece con el operador :=
```

Std_logic (y std_logic_vector)

¿Qué sucede cuando dos datos diferentes se conectan a un mismo cable?

Cuadro: Resolución de lógica

	X	**0**	**1**	**Z**	**W**	**L**	**H**	-
X	X	X	X	X	X	X	X	X
0	X	0	X	0	0	0	0	X
1	X	X	1	1	1	1	1	X
Z	X	0	1	Z	W	L	H	X
W	X	0	1	W	W	W	W	X
L	X	0	1	L	W	L	W	X
H	X	0	1	H	W	W	H	X
-	X	X	X	X	X	X	X	X

Resto de tipos de datos

- **Boolean**: True, False.
- **Integer**: entero de 32 bits (de -2,147,483,647 a + 2,147,483,647).
- **Natural**: numeros reales en rango -1.0E38 hasta +1.0E38. No es sintetizable.
- **Literales físicos**: tiempo, voltaje, etc. No sintetizable.
- **Signed y unsigned**:
  - Definidos en el paquete *std_logic_arith*.
  - Definidos en el paquete *numeric_std*.
  - Misma apariencia que los datos *std_logic_vector*.
  - Acepta operaciones aritméticas.
- ¿Por qué la librería *numeric_std* es preferida sobre *std_logic_arith* y las otras?

Paquetes VHDL: numeric_std, std_logic_arith

- Ambos definen tipos de datos *unsigned* y *signed* basados en tipos *std_logic*.
- Definen operaciones como: +, -, *, /, abs, <, >, etc.
- *std_logic_arith* fue desarrollando antes que *numeric_std*.
- *numeric_std* es preferido en la actualidad porque es estandarizado por IEEE.
- Use siempre el paquete *numeric_std* cuando necesite realizar operaciones matemática en datos *std_logic*.

Use uno u otro paquete, pero NUNCA ambos

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
--
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
```

Ejemplos

Código VHDL

```
x0 <= '0';           -- bit, std_logic, o std_ulogic value '0'
x1 <= "00011111";    -- bit_vector, std_logic_vector,
                     -- std_ulogic_vector, signed, or unsigned
x2 <= "0001_1111";   -- Guión bajo permite facilidad en visualización.
x3 <= "101111"       -- Representación binaria del 47.
x4 <= B"101111"      -- Representación binaria del 47.
x5 <= O"57"          -- Representación octal del 47.
x6 <= X"2F"          -- Representación hexadecimal del 47.
n<=1200;             -- Entero
m<=1_200;            -- Entero, guión bajo permitido.
IF ready THEN...     -- Boolean, ejecutado si ready=TRUE.
y<=1.2E-5;           -- real, no sintetizable.
q<=dafter 10 ns;     -- physical, no sintetizable.
```

Operaciones permitidas y no permitidas

Código VHDL

```
SIGNAL a: BIT;
SIGNAL b: BIT_VECTOR(7 DOWNTO 0);
SIGNAL c: STD_LOGIC;
SIGNAL d: STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL e: INTEGER RANGE 0 TO 255;
...
a<=b(5);   -- legal (mismo tipo escalar: BIT)
b(0) <= a; -- legal (mismo tipo escalar: BIT)
c<=d(5);   -- legal (mismo tipo escalar: STD_LOGIC)
d(0) <= c; -- legal (mismo tipo escalar: STD_LOGIC)
a<=c;      -- ilegal (discordancia de tipos: BIT x STD_LOGIC)
b<=d;      -- ilegal (discordancia de tipos: BIT_VECTOR x
           -- STD_LOGIC_VECTOR)
e<=b;      -- ilegal (discordancia de tipos: INTEGER x BIT_VECTOR)
e<=d;      -- ilegal (discordancia de tipos: INTEGER x
           -- STD_LOGIC_VECTOR)
```

Tipos de datos definidos por el usuario: enteros.

Código VHDL

```
TYPE integer IS RANGE -2147483647 TO +2147483647;
-- Definición del tipo de dato entero (integer).

TYPE natural IS RANGE 0 TO +2147483647;
-- Definición del tipo de dato natural.

TYPE my_integer IS RANGE -32 TO 32;
-- Subconjunto de enteros definido por el usuario.

TYPE student_grade IS RANGE 0 TO 100;
-- Subconjunto de enteros naturales definido por el usuario.
```

Tipos de datos definidos por el usuario: enumerados.

Código VHDL

```
TYPE bit IS ('0', '1');
-- Definición del tipo de dato BIT.

TYPE my_logic IS ('0', '1', 'Z');
-- Subconjunto definido por el usuario de datos std_logic.

TYPE bit_vector IS ARRAY (NATURAL RANGE <>) OF BIT;
-- Definición del tipo de dato BIT_VECTOR.
-- RANGE <>: indica rango sin restricción.
-- NATURAL RANGE <>: restringe al rango de los números naturales.

TYPE state IS (idle, forward, backward, stop);
-- Tipo de dato enumerado (tipo para máquinas de estado).

TYPE color IS (red, green, blue, white);
-- Tipo de dato enumerado.
```

Arreglos

Arreglo: colección de objetos del mismo tipo.

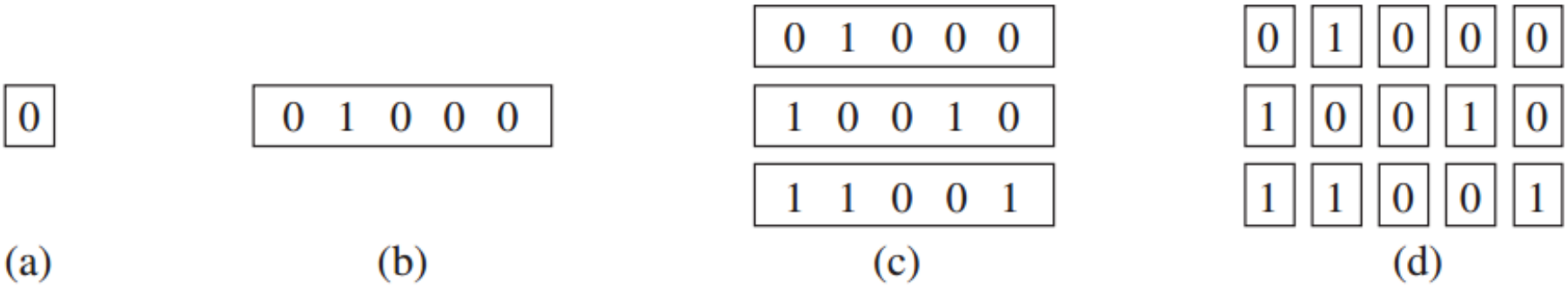

Figura: Arreglos de datos: (a) Escalar, (b) 1D, (c) 1Dx1D y (d) 2D

Definición de un arreglo

Código VHDL

```
TYPE type_name IS ARRAY (specification) OF data_type;

SIGNAL signal_name: type_name [:= initial_value];
CONSTANT signal_name: type_name [:= initial_value];
VARIABLE signal_name: type_name [:= initial_value];
```

Código VHDL

```
TYPE matrix IS ARRAY (0 TO 3) OF STD_LOGIC_VECTOR(7 DOWNTO 0);
```

Ejemplo

- Arreglo que contiene cuatro vectores.
- Cada vector con ocho bits.
- Esto define un vector de 1Dx1D.
- Cada vector se denominará *row*.
- El arreglo completo se denominará *matrix*.
- El bit más a la izquierda de cada vector es el MSB.
- La fila superior es la fila 0.

Código VHDL

```
TYPE row IS ARRAY (7 downto 0) OF STD_LOGIC;-- arreglo 1D
TYPE matrix IS ARRAY (0 TO 3) OF row;        -- arreglo 1Dx1D
SIGNAL x: matrix;
--Otra forma de hacer lo mismo del código mostrado arriba
TYPE matrix IS ARRAY (0 TO 3) OF STD_LOGIC_VECTOR(7 DOWNTO 0);
```

Arreglo 2D

Construcción basada en escalares, no en vectores.

Código VHDL

```
TYPE matrix2D IS ARRAY (0 TO 3, 7 DOWNTO 0) OF STD_LOGIC;
```

Ejemplo: inicialización de arreglo.

Código VHDL

```
... :="0001";-- Para arreglo 1D
... :=('0','0','0','1');-- Para arreglo 1D
... :=(('0','1','1','1'),('1','1','1','0'));-- Para arreglo
                                              -- 1Dx1D
                                              -- o arreglo 2D
```

Asignaciones legales e ilegales en arreglos

Código VHDL

```
TYPE row IS ARRAY (7 DOWNTO 0) OF STD_LOGIC;--1D
TYPE array1 IS ARRAY (0 TO 3) OF row;--1Dx1D

TYPE array2 IS ARRAY (0 TO 3) OF STD_LOGIC_VECTOR (7 DOWNTO 0);--1Dx1D
TYPE array3 IS ARRAY (0 TO 3, 7 DOWNTO 0) OF STD_LOGIC;--2D

SIGNAL x: row;
SIGNAL y: array1;
SIGNAL v: array2;
SIGNAL w: array3;
```

Asignaciones en escalares

Código VHDL

```
x(0)    <= y(1)(2); -- uso de paréntesis (y es 1Dx1D)
x(1)    <= v(2)(3); -- uso de paréntesis (v es 1Dx1D)
x(2)    <= w(2,1);  -- uso de paréntesis (w es 2D)
y(1)(1) <= x(6);
y(2)(0) <= v(0)(0);
y(0)(0) <= w(3,3);
w(1,1)  <= x(7);
w(3,0)  <= v(0)(3);
```

Asignaciones en vectores

Código VHDL

```
x <= y(0); -- legal (mismo tipo de datos: ROW)
x <= v(1); -- ilegal (ROW x STD_LOGIC_VECTOR)
x <= w(2); -- ilegal (w deber tener índice 2D)
x <= w(2, 2 DOWNTO 0);    -- ilegal (ROW x STD_LOGIC)
v(0) <= w(2, 2 DOWNTO 0); -- ilegal
v(0) <= w(2); -- illegal (w deber tener índice 2D)
y(1) <= v(3); -- illegal (ROW x  STD_LOGIC_VECTOR)
y(1)(7 DOWNTO 3) <= x(4 DOWNTO 0);    -- legal
v(1)(7 DOWNTO 3) <= v(2)(4 DOWNTO 0); -- legal
w(1, 5 DOWNTO 1) <= v(2)(4 DOWNTO 0); -- ilegal
```

Tipos de datos: *signed* y *unsigned*

Son tipos de datos definidos en paquetes *std_logic_arith* y *numeric_std*.

Código VHDL

```
SIGNAL x: SIGNED (7 DOWNTO 0); -- Positivo o negativo
SIGNAL y: UNSIGNED (0 TO 3);   -- Siempre mayor que 0.
```

Unsigned

- 0101_2 representa el 5_{10} decimal.
- 1101_2 representa el 13_{10} decimal.

Signed (complemento A2)

- 0101_2 representa el 5_{10} decimal.
- 1101_2 representa el el -3_{10} decimal.

Estos tipos de datos SON para ejecutar operaciones matemáticas y de relación. NO para operaciones lógicas.

Operación matemática con *std_logic_arith*(1/2)

Código VHDL

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
USE ieee.std_logic_arith.all; -- Paquete extra
--
entity arreglos is
     port( a,b: IN SIGNED (7 DOWNTO 0);
               v: OUT SIGNED (7 DOWNTO 0));
end arreglos;
--
architecture Behavioral of arreglos is
signal w: std_logic_vector (7 downto 0);
begin
     v <= a + b; --  legal (Operación aritmética OK)
     w <= a AND b; -- ilegal (Operación lógica NO OK)
end Behavioral;
```

Operación matemática con *std_logic_arith*(2/2)

Código VHDL

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
USE ieee.std_logic_arith.all; -- Paquete extra
--
entity arreglos is
     port( a,b: IN SIGNED (7 DOWNTO 0);
               v: OUT SIGNED (7 DOWNTO 0));
end arreglos;
--
architecture Behavioral of arreglos is
signal w: std_logic_vector (7 downto 0);
begin
     v <= a + b; --  legal (Operación aritmética OK)
     w <= conv_std_logic_vector(a,8) AND conv_std_logic_vector(b,8);
end Behavioral;
```

Operación matemática con *numeric_std*(1/2)

Código VHDL

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
USE ieee.numeric_std.all; -- Paquete extra
--
entity arreglos is
      port(   a:  IN SIGNED (7 DOWNTO 0);
              b:  IN SIGNED (7 DOWNTO 0);
              v: OUT SIGNED (7 DOWNTO 0));
end arreglos;
--
architecture Behavioral of arreglos is
signal w: std_logic_vector (7 downto 0);
begin
     v <= a + b; -- legal (Operación aritmética OK)
     w<=a AND b; -- ilegal (Operación lógica NO OK)
end Behavioral;
```

Operación matemática con *numeric_std*(2/2)

Código VHDL

```
use IEEE.STD_LOGIC_1164.ALL;
USE ieee.numeric_std.all; -- Paquete extra
--
entity arreglos is
port( a,b:  IN SIGNED (7 DOWNTO 0);
        v: OUT SIGNED (7 DOWNTO 0));
end arreglos;
--
architecture Behavioral of arreglos is
signal w: std_logic_vector (7 downto 0);
begin
     v <= a + b;   -- legal (Operación aritmética OK)
     w <= std_logic_vector(a) AND std_logic_vector(b);
end Behavioral;
```

Bibliografía

[1] P.P. Chu.
FPGA Prototyping by VHDL Examples: Xilinx Spartan-3 Version.
Wiley, 2008.

[2] Volnei A. Pedroni.
Circuit Design with VHDL.
MIT Press, Cambridge, MA, USA, 2004.