جامعة نينوى
كلية هندسة الإلكترونيات

Circuit Design with VHDL

2

Hussein Aideen

VHDL> Data Types

- Pre-Defined Data Types:
  - **BIT** (and **BIT_VECTOR**): 2-level logic ('0', '1').

```
SIGNAL x: BIT;

-- x is declared as a one-digit signal of type BIT.

SIGNAL y: BIT_VECTOR (3 DOWNTO 0);

-- y is a 4-bit vector, with the leftmost bit being the MSB.

SIGNAL w: BIT_VECTOR (0 TO 7);

-- w is an 8-bit vector, with the rightmost bit being the MSB.
```

VHDL> Data Types

```
x <= '1';

-- x is a single-bit signal (as specified above), whose value is
-- '1'. Notice that single quotes (' ') are used for a single bit.

y <= "0111";

-- y is a 4-bit signal (as specified above), whose value is "0111"
-- (MSB='0'). Notice that double quotes (" ") are used for
-- vectors.

w <= "01110001";

-- w is an 8-bit signal, whose value is "01110001" (MSB='1').
```

VHDL> Data Types

- **STD_LOGIC** (and **STD_LOGIC_VECTOR**):

8-valued logic system introduced in the IEEE 1164 standard.

- 'X' Forcing Unknown (synthesizable unknown)
- '0' Forcing Low (synthesizable logic '1')
- '1' Forcing High (synthesizable logic '0')
- 'Z' High impedance (synthesizable tri-state buffer)
- 'W' Weak unknown
- 'L' Weak low
- 'H' Weak high
- '–' Don't care

VHDL> Data Types

```
SIGNAL x: STD_LOGIC;

-- x is declared as a one-digit (scalar) signal of type STD_LOGIC.

SIGNAL y: STD_LOGIC_VECTOR (3 DOWNTO 0) := "0001";

-- y is declared as a 4-bit vector, with the leftmost bit being
-- the MSB. The initial value (optional) of y is "0001". Notice
-- that the ":=" operator is used to establish the initial value.
```

VHDL> Data Types

- **BOOLEAN**: True, False.
- **INTEGER**: 32-bit integers (from -2,147,483,648 to +2,147,483,647).
- **NATURAL**: Non-negative integers (from 0 to +2,147,483,647).
- **SIGNED and UNSIGNED**: data types defined in the std_logic_arith package of the ieee library. They have the appearance of STD_LOGIC_VECTOR, but accept arithmetic operations, which are typical of INTEGER data types.
- **REAL**: Real numbers ranging from -1.0E38 to +1.0E38. Not synthesizable.
- **Physical literals**: Used to inform physical quantities, like time, voltage, etc. Useful in simulations. Not synthesizable.

VHDL> Data Types

- Examples:

```
x0 <= '0'; -- bit, std_logic, or std_ulogic value '0'

x1 <= "00011111"; -- bit_vector, std_logic_vector,
                  -- std_ulogic_vector, signed, or unsigned

x2 <= "0001_1111"; -- underscore allowed to ease visualization

x3 <= "101111" -- binary representation of decimal 47
```

VHDL> Data Types

- Examples:

```
x4 <= B"101111" -- binary representation of decimal 47
x5 <= O"57" -- octal representation of decimal 47
x6 <= X"2F" -- hexadecimal representation of decimal 47

n <= 1200; -- integer

m <= 1_200; -- integer, underscore allowed

IF ready THEN... -- Boolean, executed if ready=TRUE

y <= 1.2E-5; -- real, not synthesizable

q <= d after 10 ns; -- physical, not synthesizable
```

VHDL> Data Types

```
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 (same scalar type: BIT)
b(0) <= a; -- legal (same scalar type: BIT)
c <= d(5); -- legal (same scalar type: STD_LOGIC)
d(0) <= c; -- legal (same scalar type: STD_LOGIC)

a <= c; -- illegal (type mismatch: BIT x STD_LOGIC)
b <= d; -- illegal (type mismatch: BIT_VECTOR x
        -- STD_LOGIC_VECTOR)
e <= b; -- illegal (type mismatch: INTEGER x BIT_VECTOR)
e <= d; -- illegal (type mismatch: INTEGER x
        -- STD_LOGIC_VECTOR)
```

Ex: Write VHDL code to design 8-3 encoder (use if statement)

```
1  library ieee;
2      use ieee.std_logic_1164.all;
3
4  entity encoder_using_if is
5      port (
6          enable     :in  std_logic;                          --  Enable for the encoder
7          encoder_in :in  std_logic_vector (7 downto 0); --  16-bit Input
8          binary_out :out std_logic_vector (2 downto 0)    --  4 bit binary Output
9
10     );
11 end entity;
12
13 architecture behavior of encoder_using_if is
14
15 begin
16     process (enable, encoder_in) begin
17         binary_out <= "00";
18         if (enable = '1') then
19             if (encoder_in = X"02") then binary_out <= "001"; end if;
20             if (encoder_in = X"04") then binary_out <= "010"; end if;
21             if (encoder_in = X"08") then binary_out <= "011"; end if;
22             if (encoder_in = X"10") then binary_out <= "100"; end if;
23             if (encoder_in = X"20") then binary_out <= "101"; end if;
24             if (encoder_in = X"40") then binary_out <= "110"; end if;
25             if (encoder_in = X"80") then binary_out <= "111"; end if;
26
27         end if;
28     end process;
29 end architecture;
```

VHDL> Data Types

- User-Defined Data Types:
  - VHDL also allows the user to define his/her own data types.

```
TYPE integer IS RANGE -2147483647 TO +2147483647;
-- This is indeed the pre-defined type INTEGER.

TYPE natural IS RANGE 0 TO +2147483647;
-- This is indeed the pre-defined type NATURAL.

TYPE my_integer IS RANGE -32 TO 32;
-- A user-defined subset of integers.

TYPE student_grade IS RANGE 0 TO 100;
-- A user-defined subset of integers or naturals.
```

VHDL> Data Types

```
TYPE bit IS ('0', '1');
-- This is indeed the pre-defined type BIT

TYPE my_logic IS ('0', '1', 'Z');
-- A user-defined subset of std_logic.

TYPE state IS (idle, forward,
               backward, stop);
-- An enumerated data type, typical of
--    finite state machines.

TYPE color IS (red, green, blue, white);
-- Another enumerated data type.
```

VHDL> Data Types

- Sub-Types:
  - The main reason for using a subtype rather than specifying a new type is that, though operations between data of different types are not allowed, they are allowed between a subtype and its corresponding base type.

```
SUBTYPE natural IS INTEGER RANGE 0 TO INTEGER'HIGH;
-- As expected, NATURAL is a subtype (subset) of INTEGER.

SUBTYPE my_logic IS STD_LOGIC RANGE '0' TO 'Z';
-- Recall that STD_LOGIC=('X','0','1','Z','W','L','H','-').
 -- Therefore, my_logic=('0','1','Z').

SUBTYPE my_color IS color RANGE red TO blue;
-- Since color=(red, green, blue, white), then
-- my_color=(red, green, blue).

SUBTYPE small_integer IS INTEGER RANGE -32 TO 32;
-- A subtype of INTEGER.
```

VHDL> Data Types

- Signed and Unsigned Data Types:
  - defined in the *std_logic_arith* package of the *ieee* library.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;    -- extra package necessary
...
SIGNAL a: IN SIGNED (7 DOWNTO 0);
SIGNAL b: IN SIGNED (7 DOWNTO 0);
SIGNAL x: OUT SIGNED (7 DOWNTO 0);
...
v <= a + b;       -- legal (arithmetic operation OK)
w <= a AND b;     -- illegal (logical operation not OK)
```

VHDL> Data Types

Example: Legal and illegal operations with std_logic_vector.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;    -- no extra package required
...
SIGNAL a: IN STD_LOGIC_VECTOR (7 DOWNTO 0);
SIGNAL b: IN STD_LOGIC_VECTOR (7 DOWNTO 0);
SIGNAL x: OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
...
v <= a + b;       -- illegal (arithmetic operation not OK)
w <= a AND b;     -- legal (logical operation OK)
```

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;    -- extra package included

SIGNAL a: IN STD_LOGIC_VECTOR (7 DOWNTO 0);
SIGNAL b: IN STD_LOGIC_VECTOR (7 DOWNTO 0);
SIGNAL x: OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
...
v <= a + b;       -- legal (arithmetic operation OK), unsigned
w <= a AND b;     -- legal (logical operation OK)
```