VHDL – TD1

1 Syntaxe VHDL

(V. Pedroni, “Circuit Design with VHDL ”, MIT Press)

Déterminer les instructions VHDL légales et illégales.

1.1 Legal and illegal operations between data of different 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)
```

1.2 Example: Legal and illegal operations between types and subtypes.

```
SUBTYPE my_logic IS STD_LOGIC RANGE '0' TO '1';
SIGNAL a: BIT;
SIGNAL b: STD_LOGIC;
SIGNAL c: my_logic;
...
b <= a; --illegal (type mismatch: BIT versus STD_LOGIC)
b <= c; --legal (same "base" type: STD_LOGIC)
```

1.3 Legal and illegal array assignments.

The assignments in this example are based on the following type definitions and signal declarations:

```
TYPE row IS ARRAY (7 DOWNTO 0) OF STD_LOGIC; -- 1D array
TYPE array1 IS ARRAY (0 TO 3) OF row; -- 1Dx1D array
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 array
SIGNAL x: row;
SIGNAL y: array1;
SIGNAL v: array2;
SIGNAL w: array3;
```

```
--------- Scalar assignments: ---------------
-- The scalar (single bit) assignments below are all legal,
-- because the "base" (scalar) type is STD_LOGIC for all signals
-- (x,y,v,w).
x(0) <= y(1)(2);  -- notice two pairs of parenthesis
                  -- (y is 1Dx1D)
x(1) <= v(2)(3);  -- two pairs of parenthesis (v is 1Dx1D)
x(2) <= w(2,1);   -- a single pair of parenthesis (w is 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);

--------- Vector assignments: ---------------------
x <= y(0);  -- legal (same data types: ROW)
x <= v(1);  -- illegal (type mismatch: ROW x
            -- STD_LOGIC_VECTOR)
x <= w(2);  -- illegal (w must have 2D index)
x <= w(2, 2 DOWNTO 0);  -- illegal (type mismatch: ROW x
                        -- STD_LOGIC)
v(0) <= w(2, 2 DOWNTO 0);     -- illegal (mismatch: STD_LOGIC_VECTOR
                              -- x STD_LOGIC)
v(0) <= w(2);     -- illegal (w must have 2D index)
y(1) <= v(3);     -- illegal (type mismatch: ROW x
                  -- STD_LOGIC_VECTOR)
y(1)(7 DOWNTO 3) <= x(4 DOWNTO 0);  -- legal (same type,
                                    -- same size)
v(1)(7 DOWNTO 3) <= v(2)(4 DOWNTO 0);     -- legal (same type,
                                          -- same size)
w(1, 5 DOWNTO 1) <= v(2)(4 DOWNTO 0); -- illegal (type mismatch)
```

1.4 Legal and illegal operations with signed/unsigned data types.

```
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)
```

1.5 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)
```

1.6 Legal and illegal operations with subsets.

```
TYPE long IS INTEGER RANGE -100 TO 100;
TYPE short IS INTEGER RANGE -10 TO 10;
SIGNAL x : short;
SIGNAL y : long;
...
y <= 2*x + 5; -- error, type mismatch
y <= long(2*x + 5); -- OK, result converted into type long
```

The legal and illegal assignments presented next are based on the following type definitions and signal declarations:

```
TYPE byte IS ARRAY (7 DOWNTO 0) OF STD_LOGIC; -- 1D array
TYPE mem1 IS ARRAY (0 TO 3, 7 DOWNTO 0) OF STD_LOGIC; -- 2D array
TYPE mem2 IS ARRAY (0 TO 3) OF byte; -- 1Dx1D array
TYPE mem3 IS ARRAY (0 TO 3) OF STD_LOGIC_VECTOR(0 TO 7); -- 1Dx1D array
SIGNAL a: STD_LOGIC; -- scalar signal
SIGNAL b: BIT; -- scalar signal
SIGNAL x: byte; -- 1D signal
SIGNAL y: STD_LOGIC_VECTOR (7 DOWNTO 0); -- 1D signal
SIGNAL v: BIT_VECTOR (3 DOWNTO 0); -- 1D signal
SIGNAL z: STD_LOGIC_VECTOR (x'HIGH DOWNTO 0); -- 1D signal
SIGNAL w1: mem1; -- 2D signal
SIGNAL w2: mem2; -- 1Dx1D signal
SIGNAL w3: mem3; -- 1Dx1D signal

-------- Legal scalar assignments: ---------------------
x(2) <= a; -- same types (STD_LOGIC), correct indexing
y(0) <= x(0); -- same types (STD_LOGIC), correct indexing
z(7) <= x(5); -- same types (STD_LOGIC), correct indexing
b <= v(3); -- same types (BIT), correct indexing
w1(0,0) <= x(3); -- same types (STD_LOGIC), correct indexing
w1(2,5) <= y(7); -- same types (STD_LOGIC), correct indexing
w2(0)(0) <= x(2); -- same types (STD_LOGIC), correct indexing
w2(2)(5) <= y(7); -- same types (STD_LOGIC), correct indexing
w1(2,5) <= w2(3)(7); -- same types (STD_LOGIC), correct indexing

------- Illegal scalar assignments: --------------------
b <= a; -- type mismatch (BIT x STD_LOGIC)
w1(0)(2) <= x(2); -- index of w1 must be 2D
w2(2,0) <= a; -- index of w2 must be 1Dx1D

------- Legal vector assignments: ----------------------
x <= "11111110";
y <= ('1','1','1','1','1','1','0','Z');
z <= "11111" & "000";
x <= (OTHERS => '1');
y <= (7 =>'0', 1 =>'0', OTHERS => '1');
z <= y;
y(2 DOWNTO 0) <= z(6 DOWNTO 4);
w2(0)(7 DOWNTO 0) <= "11110000";
w3(2) <= y;
z <= w3(1);
z(5 DOWNTO 0) <= w3(1)(2 TO 7);
w3(1) <= "00000000";
```

```
w3(1) <= (OTHERS => '0');
w2 <= ((OTHERS=>'0'),(OTHERS=>'0'),(OTHERS=>'0'),(OTHERS=>'0'));
w3 <= ("11111100", ('0','0','0','0','Z','Z','Z','Z',),
      (OTHERS=>'0'), (OTHERS=>'0'));
w1 <= ((OTHERS=>'Z'), "11110000" ,"11110000", (OTHERS=>'0'));

------ Illegal array assignments: ----------------------
x <= y; -- type mismatch
y(5 TO 7) <= z(6 DOWNTO 0); -- wrong direction of y
w1 <= (OTHERS => '1'); -- w1 is a 2D array
w1(0, 7 DOWNTO 0) <="11111111"; -- w1 is a 2D array
w2 <= (OTHERS => 'Z'); -- w2 is a 1Dx1D array
w2(0, 7 DOWNTO 0) <= "11110000"; -- index should be 1Dx1D
```

2 Attributes

(P. Ashenden, “Systems on Silicon”, Morgan Kaufmann Publishers)

2.1 Attributes of Scalar Types

```
TYPE Resistance IS RANGE 0 TO 1E9
      UNITS
            ohm;
            kohm = 1000 ohm;
            Mohm = 1000 kohm;
      END UNITS Resistance;

TYPE Set_index_range IS RANGE 21 DOWNTO 11;
TYPE Logic_level is (unknown,low,undriven,high);

Resistance'LEFT = 0 ohm
Resistance'RIGHT = 1E9 ohm
Resistance'LOW = 0 ohm
Resistance'HIGH = 1E9 ohm
Resistance'ASCENDING = true
Resistance'IMAGE(2 kohm) = "2000 ohm"
Resistance'VALUE("5 Mohm") = 5_000_000 ohm

Set_index_range'LEFT = 21
Set_index_range'RIGHT = 11
Set_index_range'LOW = 11
Set_index_range'HIGH = 21
Set_index_range'ASCENDING = false
Set_index_range'IMAGE(14) = "14"
Set_index_range'VALUE("20") = 20

Logic_Level'LEFT = unknown
Logic_Level'RIGHT = high
Logic_Level'LOW = unknown
Logic_Level'HIGH = high
Logic_Level'ASCENDING = true
Logic_Level'IMAGE(undriven) = "undriven"
Logic_Level'VALUE("low") = low
```

2.2 Attributes of arrays.

```
TYPE A IS ARRAY (1 TO 4, 31 DOWNTO 0) OF BOOLEAN;

A'LEFT(1) = 1
A'LOW(1) = 1
A'RIGHT(2) = 0
A'HIGH(2) = 31
A'LENGTH(1) = 4
A'LENGTH(2) = 32
A'RANGE(1) = 1 TO 4
A'REVERSE_RANGE(2) = 0 TO 31
A'ASCENDING(1) = true
A'ASCENDING(2) = false
```