ICMC - USP

Introduction to VHDL (part 2 of 2)

Prof. Vanderlei Bonato - vbonato@icmc.usp.br

Summary

- Data Types
- Assignments
- Data Conversions
- Operators
- Component Instantiation
- Bi-directional Pins
- Exercises

Data Types

Data types	Synthesizable values
BIT, BIT_VECTOR	'0', '1'
STD_LOGIC, STD_LOGIC_VECTOR	'X', '0', '1', 'Z' (resolved)
STD_ULOGIC, STD_ULOGIC_VECTOR	'X', '0', '1', 'Z' (unresolved)
BOOLEAN	True, False
NATURAL	From 0 to +2, 147, 483, 647
INTEGER	From −2,147,483,647 to +2,147,483,647
SIGNED	From −2,147,483,647 to +2,147,483,647
UNSIGNED	From 0 to +2,147,483,647
User-defined integer type	Subset of INTEGER
User-defined enumerated type	Collection enumerated by user
SUBTYPE	Subset of any type (pre- or user-defined)
ARRAY	Single-type collection of any type above
RECORD	Multiple-type collection of any types above

Dealing with Data Types

```
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
```

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
```

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 Assignments

```
------- 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


------ 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
```

DOWNTO and TO

```
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.

  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').
```

Bit Levels

- BIT (and BIT_VECTOR): 2-level logic ('0', '1')

- 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	

Most of the std_logic are intended for simulation only!

ULOGIC

- STD_ULOGIC (STD_ULOGIC_VECTOR): 9-level logic system introduced in the IEEE 1164 standard ('U', 'X', '0', '1', 'Z', 'W', 'L', 'H', '–').

- STD_LOGIC system described above is a *subtype* of STD_ULOGIC. The latter includes an extra logic value, 'U', which stands for unresolved. Thus, contrary to STD_LOGIC, conflicting logic levels are not automatically resolved here, so output wires should never be connected together directly. However, if two output wires are never supposed to be connected together, this logic system can be used to detect design errors.

SIGNED and UNSIGNED

- Their syntax similar to STD_LOGIC_VECTOR
- SIGNED and UNSIGNED are intended mainly for arithmetic operations
- Logic operations are not allowed

```
SIGNAL x: SIGNED (7 DOWNTO 0);
SIGNAL y: UNSIGNED (0 TO 3);
```

Data Conversion

- VHDL does not allow direct operations between data of different types
- Conversions are necessary
- Several data conversion functions can be found in the std_logic_arith package of IEEE library

std_logic_arith Conversion Functions

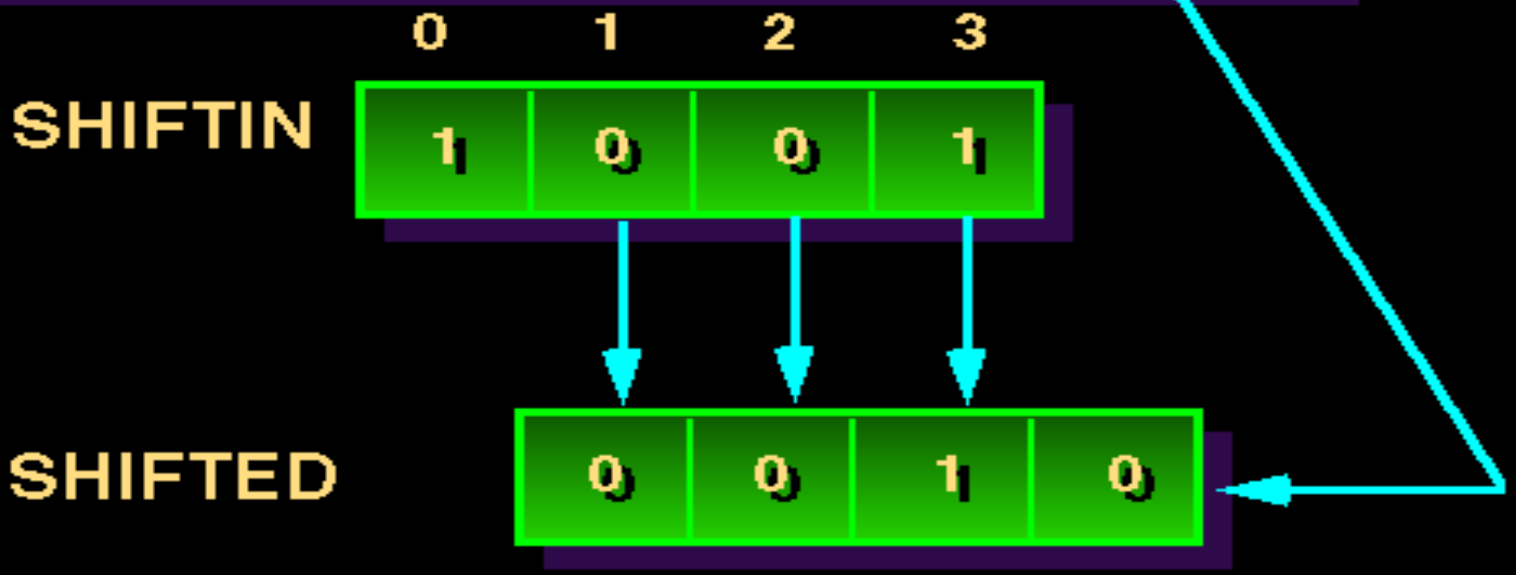
- conv_integer(p) : Converts a parameter p of type INTEGER, UNSIGNED, SIGNED, or STD_ULOGIC to an INTEGER value. Notice that STD_LOGIC_VECTOR is not included.
- conv_unsigned(p, b): Converts a parameter p of type INTEGER, UNSIGNED, SIGNED, or STD_ULOGIC to an UNSIGNED value with size b bits.
- conv_signed(p, b): Converts a parameter p of type INTEGER, UNSIGNED, SIGNED, or STD_ULOGIC to a SIGNED value with size b bits.
- conv_std_logic_vector(p, b): Converts a parameter p of type INTEGER, UNSIGNED, SIGNED, or STD_LOGIC to a STD_LOGIC_VECTOR value with size b bits.

Operators

Operator type	Operators	Data types
Assignment	<=, :=, =>	Any
Logical	NOT, AND, NAND, OR, NOR, XOR, XNOR	BIT, BIT_VECTOR, STD_LOGIC, STD_LOGIC_VECTOR, STD_ULOGIC, STD_ULOGIC_VECTOR
Arithmetic	+, −, *, /, ** (mod, rem, abs)♦	INTEGER, SIGNED, UNSIGNED
Comparison	=, /=, <, >, <=, >=	All above
Shift	sll, srl, sla, sra, rol, ror	BIT_VECTOR
Concatenation	&, (,,,)	Same as for logical operators, plus SIGNED and UNSIGNED

The concatentation operator &

```
VARIABLE shifted, shiftin : BIT_VECTOR (0 TO 3) :
....
shifted := shiftin (1 TO 3) & '0';
```



The exponentiation operator **

```
x := 5**5  — 5^5, OK
y := 0.5**3 — 0.5^3, OK
x := 4**0.5 — 4^0.5, bad
y := 0.5**(-2) — 0.5^(-2), OK
```

Component instantiation (Structural VHDL)

```
component fifo_cam is
        port(       data            : in STD_LOGIC_VECTOR (31 downto 0);
                    wrreq           : in STD_LOGIC ;
                    rdreq           : in STD_LOGIC ;
                    rdclk           : in STD_LOGIC ;
                    wrclk           : in STD_LOGIC ;
                    aclr            : in STD_LOGIC ;
                    q                 : out STD_LOGIC_VECTOR (31 downto 0);
                    rdempty         : out STD_LOGIC ;
                    wrfull          : out STD_LOGIC );
end component;

fifo: fifo_cam port map(pixel,'1',read_cs,clk_n,ready_pixel,aclr_fifo,readdata,waitrequest,fifofull);
```

Bidirectional pin

```
ENTITY proc_eld2 is
      PORT(clk, rst                 : in          STD_LOGIC;
                data                : inout       STD_LOGIC_VECTOR(7 downto 0);
                web_oeb             : buffer   STD_LOGIC;
                address             : out   STD_LOGIC_VECTOR(7 downto 0);
                pc_out, ir_out      : out   STD_LOGIC_VECTOR(7 downto 0);
                saida               : out STD_LOGIC_VECTOR(2 downto 0)
               );
END proc_eld2;
```

signal ACC : std_logic_vector(7 downto 0);

web_oeb <= ‘1'; --1 escreve e 0 lê da mem.
data <= ACC WHEN web_oeb='1' else "ZZZZZZZZ";

web_oeb <= ‘0'; --1 escreve e 0 lê da mem.
ACC <= data;

Tips

- **The ENTITY name and the file name must be the same**

- **Physical and time data types are not synthesizable for FPGAs**
  - **ohm, kohm**
  - **fs, ps, ns, um, ms, min, hr**

And more ...

- **Function**
  - **Produce a single return value**
  - **Requires a RETURN statement**
- **Procedure**
  - **Produce many output values**
  - **Do not require a RETURN statement**
- **Testbench**
  - **Generate stimulus for simulation**
  - **Compare output responses with expected values**

Implemente em VHDL os seguintes componentes

- FFs do tipo D, T e JK
- Registrador de deslocamento da direita para a esquerda
- Conversor de binário para display de 7 segmentos
- Crie um componente somador completo de 1 bit e instancie esse mesmo componente para formar um somador/subtrator de 8 bits do tipo ripple-carry. Considere que os números estão em complemento de 2; e para o controle da operação utilize C=0 para adição e C=1 para subtração. Indique também overflow. Utilize STD_LOGIC_VECTOR para os sinais de entrada e saída

References

- Pedroni, Volnei A. Circuit Design with VHDL, MIT Press, 2004

- DARPA/Tri-Services RASSP Program
  - http://www.vhdl.org/rassp/

- Brown, S. and Vranesic, Z.. Fundamentals of Digital Logic with VHDL Design, 2nd Ed., P. 939, 2005.