جامعة نينوى
كلية هندسة الإلكترونيات

Circuit Design with VHDL 4

Textbook: Volnei A. Pedroni

Submitted By: Hussein Aideen

VHDL> Operators

- **Logical operators:**
- perform logical operations.
- Data types: BIT, STD_LOGIC, STD_ULOGIC, BIT_VECTOR, STD_LOGIC_VECTOR, or STD_ULOGIC_VECTOR

- NOT
  - AND
  - OR
  - NAND
  - NOR
  - XOR
  - XNOR

Examples:

```
y <= NOT a AND b;        -- (a'.b)
y <= NOT (a AND b);      -- (a.b)'
y <= a NAND b;           -- (a.b)'
```

VHDL> Operators

- **Arithmetic Operators:**
- perform arithmetic operations
- data types: INTEGER, SIGNED, UNSIGNED, or REAL
- With *std_logic_signed* or *std_logic_unsigned* package: STD_LOGIC_VECTOR.

+	Addition	
–	Subtraction	
*	Multiplication	
/	Division	only power of two dividers
**	Exponentiation	only static values of base and exponent are accepted

VHDL> Operators

- N bit adder circuit:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

------------------------------------------------------

entity ADDER is

generic(n: natural :=2);
port( A: in std_logic_vector(n-1 downto 0);
   B: in std_logic_vector(n-1 downto 0);
   carry:   out std_logic;
   sum:  out std_logic_vector(n-1 downto 0)
);

end ADDER;
```

VHDL> Operators

- N bit adder circuit:

```
architecture behv of ADDER is

-- define a temparary signal to store the result

signal result: std_logic_vector(n downto 0);

begin

    -- the 3rd bit should be carry

    result <= ('0' & A)+('0' & B);
    sum <= result(n-1 downto 0);
    carry <= result(n);

end behv;
```

VHDL> Operators

- **Comparison Operators:**
- Used for making comparisons.
- Data types: any.

=	Equal to
/=	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

VHDL> Operators

N bit comparator:

```
library ieee;
use ieee.std_logic_1164.all;

-----------------------------------------------------

entity Comparator is

generic(n: natural :=2);
port( A: in std_logic_vector(n-1 downto 0);
   B: in std_logic_vector(n-1 downto 0);
   less:    out std_logic;
   equal:      out std_logic;
   greater: out std_logic
);
end Comparator;

-----------------------------------------------------
```

VHDL> Operators

```
architecture behv of Comparator is

begin

    process(A,B)
    begin
        if (A<B) then
       less <= '1';
       equal <= '0';
       greater <= '0';
   elsif (A=B) then
       less <= '0';
       equal <= '1';
       greater <= '0';
   else
       less <= '0';
       equal <= '0';
       greater <= '1';
   end if;
    end process;

end behv;
```

VHDL> Operators

- **Shift Operators:**
  - sll Shift left logic – positions on the right are filled with '0's
  - srl Shift right logic – positions on the left are filled with '0's
- Syntax:

⟨left operand⟩ ⟨shift operation⟩ ⟨right operand⟩

BIT_VECTOR

INTEGER

sll, srl, sla, sra, rol, ror

VHDL> Data Attributes

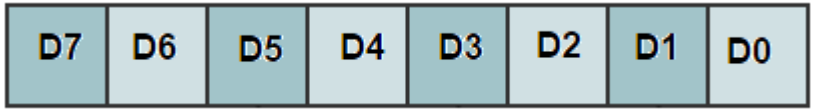
- The pre-defined, synthesizable data attributes are the following:
- d'LOW: Returns lower array index
- d'HIGH: Returns upper array index
- d'LEFT: Returns leftmost array index
- d'RIGHT: Returns rightmost array index
- d'LENGTH: Returns vector size
- d'RANGE: Returns vector range
- d'REVERSE_RANGE: Returns vector range in reverse order

VHDL> Data Attributes

Example: Consider the following signal:

```
SIGNAL d : STD_LOGIC_VECTOR (7 DOWNTO 0);
```

Then:

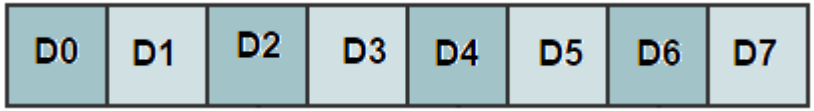



```
d'LOW=0, d'HIGH=7, d'LEFT=7, d'RIGHT=0, d'LENGTH=8,
d'RANGE=(7 downto 0), d'REVERSE_RANGE=(0 to 7).
```

Example: Consider the following signal:

```
SIGNAL x: STD_LOGIC_VECTOR (0 TO 7);
```

Then all four LOOP statements below are synthesizable and equivalent.




```
FOR i IN RANGE (0 TO 7) LOOP ...
FOR i IN x'RANGE LOOP ...
FOR i IN RANGE (x'LOW TO x'HIGH) LOOP ...
FOR i IN RANGE (0 TO x'LENGTH-1) LOOP ...
```

VHDL> Signal Attributes

- Let us consider a signal s. Then:
  - s'EVENT: Returns true when an event occurs on s.
  - s'STABLE: Returns true if no event has occurred on s.

```
IF (clk'EVENT AND clk='1')...          -- EVENT attribute used
                                       -- with IF
IF (NOT clk'STABLE AND clk='1')...     -- STABLE attribute used
                                       -- with IF
WAIT UNTIL (clk'EVENT AND clk='1');    -- EVENT attribute used
                                       -- with WAIT
IF RISING_EDGE(clk)...                 -- call to a function
```

VHDL> User-Defined Attributes

- VHDL also allows the construction of user defined attributes.

```
ATTRIBUTE attribute_name: attribute_type;

ATTRIBUTE attribute_name OF target_name: class IS value;
```

```
 -- declaration
ATTRIBUTE number_of_inputs: INTEGER;
 -- specification
ATTRIBUTE number_of_inputs OF nand3: SIGNAL IS 3;
...
 -- attribute call, returns 3
inputs <= nand3'number_of_pins;
```