Operatori

- Gli operatori permettono di formare espressioni complesse, e.g. :

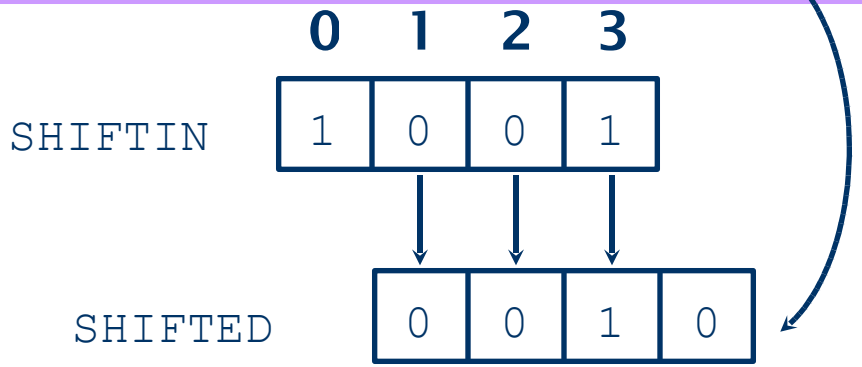
```
res <= a AND NOT(B) OR NOT(a) AND b;
```

- Precedenze in ordine decrescente:
  - Miscellaneous operators: **, abs, not
  - Multiplication operators: *, /, mod, rem
  - Sign operator: +, -
  - Addition operators: +, -, &
  - Shift operators: sll, srl, sla, sra, rol, ror
  - Relational operators: =, /=, <, <=, >, >=
  - Logical operators: AND, OR, NAND, NOR, XOR, XNOR

Operatori: Esempi

- Concatenazione: &

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



- **Elevamento a potenza ****

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

Files

- Il VHDL definisce gli oggetti file, dei tipi di dato associati e delle operazioni

```
TYPE file_type IS FILE OF type_mark;
PROCEDURE READ(FILE identifier : file_type; value : OUT type_mark);
PROCEDURE WRITE(FILE identifier : file_type; value : IN type_mark);
FUNCTION ENDFILE(FILE identifier : file_type) RETURN BOOLEAN;
```

- File declarations

```
FILE identifier : file_type [[OPEN mode] IS "file_name"];
```

- Il VHDL usa il package TEXTIO della library STD per manipolare file di testo

```
USE STD.TEXTIO.ALL;
```

Procedure del package TEXTIO

- TEXTIO definisce un tipo di dato LINE su cui eseguire operazioni di read e write

- TEXTIO definisce un tipo di datoTEXT per I FILE da utilizzare con I file ASCII
- Le procedure definited sono:
  - Readline(f,k)
    - legge una line dal file f e la mette nel buffer k
  - Read(k,v,)
    - Legge un valore v da k
  - Write(k,v,...)
    - Scrive il valore v sulla LINE k
  - Writeline(f,k)
    - Scrive la LINE k sul file f
  - Endfile(f) restituisce TRUE alla fine del file f

Testbenches

- Il Testbench è il top level del sistema da simulare
  - La Entity non contiene PORT
  - Instanzia tutti i componenti che formano il sistema
- I Testbenches possono :
  - generare gli stimoli per la simulazione:
  - Applicare gli stimoli all' Entity da testare
  - Confrontare gli output con i valori attesi
- Usano costrutti non sintetizzabili

Testbench:esempio 1

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

entity registro_8bit is
   port(
        clock,reset:in std_logic;
        din:in std_logic_vector(7 downto 0);
        dout:out std_logic_vector(7 downto 0));
end registro_8bit;

architecture arch of registro_8bit is
    begin
        process(clock,reset)
            begin
                if reset='1' then
                    dout<=(others=>'0');
                elsif clock='1' and clock'event then
                    dout<=din;
                end if;
        end process;
end arch;
```

Testbench:esempio 1

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

entity registro_8bit_tb is
end registro_8bit_tb;

architecture TB_ARCHITECTURE of
        registro_8bit_tb is
        -- Component declaration of the tested unit
        component registro_8bit
        port(
                clock : in std_logic;
                reset : in std_logic;
                din : in std_logic_vector(7 downto 0);
                dout : out std_logic_vector(7
        downto0) );
        end component;

        -- Stimulus signals - signals mapped to the
        input and inout ports of tested entity
        signal clock : std_logic:='0';
        signal reset : std_logic;
        signal din : std_logic_vector(7 downto 0);
        -- Observed signals –
        signal dout : std_logic_vector(7 downto 0);
constant clock_period:time:=10 ns;

begin
        -- Unit Under Test port map
        UUT : registro_8bit
                port map (
                        clock => clock,
                        reset => reset,
                        din => din,
                        dout => dout
                );
        reset<='1','0' after 100 ns;
        clock<= not(clock) after clock_period/2;
        din<="00000001", "00000010" after 298 ns,
        "00000011" after 597 ns;

end TB_ARCHITECTURE;
```

Testbenches: esempio 2

```
PACKAGE count_types IS
    SUBTYPE bit8 is INTEGER RANGE 0 to 255;
END count_types;

LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE work.count_types.all;

ENTITY count IS
    PORT (clk : IN std_logic;
          ld : IN std_logic;
          up_dwn : IN std_logic;
          clk_en : IN std_logic;
          din : IN bit8;
          qout : INOUT bit8);
END count;

ARCHITECTURE synthesis OF count IS
SIGNAL count_val : bit8;
BEGIN
    PROCESS(ld, up_dwn, din, qout)
        BEGIN
            IF ld = '1' THEN count_val <= din;
            ELSIF up_dwn = '1' THEN
                IF (qout >= 255) THEN
                count_val <= 0;
                ELSE
                count_val <= count_val + 1;
                END IF;
            ELSE
                IF (qout <= 0) THEN
                count_val <= 255;
                ELSE
                count_val <= count_val - 1;
                END IF;
            END IF;
END PROCESS;
    PROCESS
            BEGIN
            WAIT UNTIL clk'EVENT AND clk = '1';
            IF clk_en = '1' THEN
            qout <= count_val;
            END IF;
    END PROCESS;
END synthesis
```

Testbenches: esempio 2

```
ENTITY testbench IS END;

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE std.textio.ALL;
USE WORK.count_types.all;
ARCHITECTURE stimonly OF testbench IS
      COMPONENT count
            PORT (clk : IN std_logic;
                  ld : IN std_logic;
                  up_dwn : IN std_logic;
                  clk_en : IN std_logic;
                  din : IN bit8;
                  qout : INOUT bit8);
END COMPONENT;
SIGNAL clk, ld, up_dwn, clk_en : std_logic;
SIGNAL qout, din : bit8;
BEGIN
-- instantiate the component
uut: count PORT MAP(clk => clk, ld => ld, up_dwn
      => up_dwn, clk_en => clk_en, din => din, qout
      => qout);
```

```
            -- provide stimulus and check the result
test: PROCESS
VARIABLE tmpclk, tmpld, tmpup_dwn, tmpclk_en :
std_logic;
VARIABLE tmpdin : integer;
FILE vector_file : text IS IN “counter.txt”;
VARIABLE l : line;
VARIABLE space : character;
BEGIN
WHILE NOT endfile(vector_file) LOOP
readline(vector_file, l); -- read line
read(l,tmpclk); -- read clk value
read(l,tmpld); -- read ld value
read(l,tmpclk_en); -- read clk_en value
read(l, tmpup_dwn); -- read up_dwn value
read(l, space); --- skip a space
read(l, tmpdin);
Clk<=tmpclk_en; ld <= tmpld; up_dwn <=
      tmpup_dwn; clk_en <= tmpclk_en; din <=
      tmpdin;
WAIT FOR 10 ns;
END LOOP;
END PROCESS;
END;
```

Simulazione VHDL

Vlib:

Creates a new design library.

Syntax

vlib [<logical_name>] <physical_name>

Arguments

<logical_name>

Specifies the logical name of the library to be created. If not specified, the logical name of the library is assumed to be the same as the physical name.

<physical_name>

The name of the library index file with the full path and extension LIB.

Example:

Assuming that the current working design folder is *c:\my_designs*. Each of the commands:

vlib test "c:\my_designs\test\test.lib"

vlib "c:\my_designs\test\test.lib"

vlib "c:\my_designs\test“

creates the library *c:\my_designs\test\test.lib* with logical name *test*.

Simulazione VHDL

Invokes the VHDL compiler. Use this command to compile the selected VHDL source file.

Syntax

vcom [-f <filename>] [-work <library_name>] [-reorder] [...]

Arguments

-f <filename>

Specifies an optional text file containing arguments for the acom command. You can use this switch instead of specifying the arguments directly in the command line.

-work <library_name>

Specifies the default working library for the compiled file(s). If omitted, the current active library (from the current active design) is assumed.

-refresh <library_name>

-reorder

Runs the compiler in the *Compile with File Reorder* mode.

<filename ... >

Specifies the location and name(s) of the source file(s) to compile.

Simulazione VHDL

Vsim:

Invokes the simulator and initializes the process of simulation. You can set the simulator to elaborate the model and initialize simulation from the design library.

You can specify a configuration or an entity/architecture pair for simulation

Syntax

vsim [-lib <library>] [-sdfmin | -sdftyp | -sdfmax <region>=<sdf_filename>] [-g<generic_name=value>] <configuration> | <entity> [<architecture>]

Arguments

-lib <library>

Specifies the logical name of the library (currently set as active) that contains the top-level design unit. If omitted, the current working library is assumed.

-g<generic_name=value>

Defines a value for all generics in the project with the specified name that have not received explicit values in generic maps.

Simulazione VHDL

-sdfmax

Annotates VITAL cells in the specified region with maximum timing values from the SDF file.

-sdfmin

Annotates VITAL cells in the specified region with minimum timing values from the SDF file.

-sdftyp

Annotates VITAL cells in the specified region with typical timing values from the SDF file.

[<region>=]<sdf_filename>

Specifies the design region into which timing data is to be loaded from the SDF file specified b
the <sdf_filename> parameter.

Esempio:

vsim work. registro_8bit_tb (TB_ARCHITECTURE)

add wave test_bench/*

run 20 ms