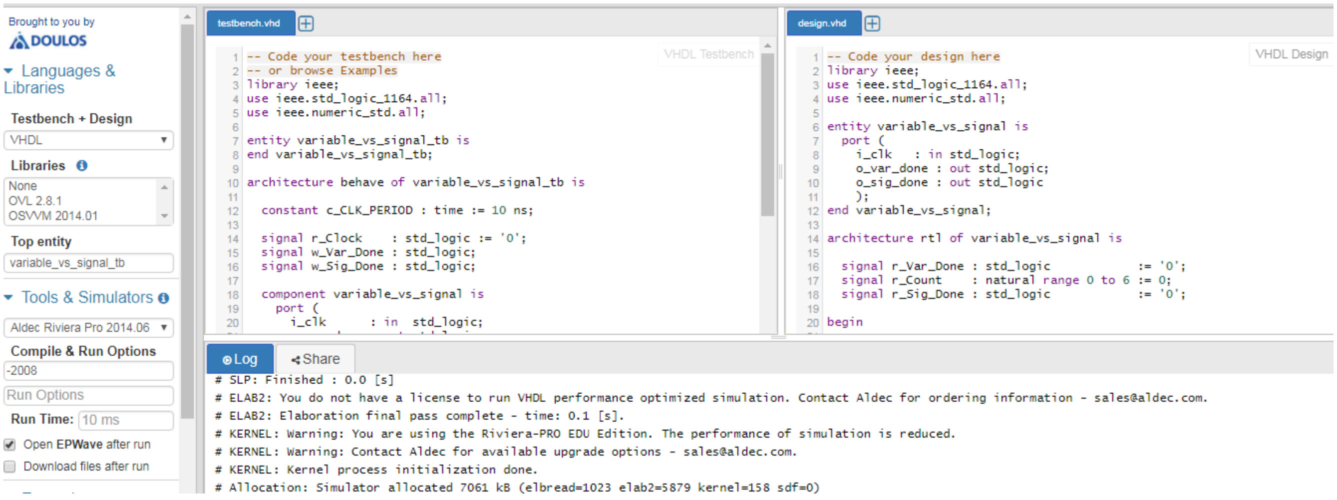
Primer 1 VHDL – varijable i signali



Testbench + Design: VHDL
Libraries: None, OVL 2.8.1, OSVVM 2014.01
Top entity: variable_vs_signal_tb
Tools & Simulators: Aldec Riviera Pro 2014.06
Compile & Run Options: -2008
Run Options
Run Time: 10 ms
☑ Open EPWave after run
☐ Download files after run

testbench.vhd

```
-- Code your testbench here
-- or browse Examples
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity variable_vs_signal_tb is
end variable_vs_signal_tb;

architecture behave of variable_vs_signal_tb is

  constant c_CLK_PERIOD : time := 10 ns;

  signal r_Clock    : std_logic := '0';
  signal w_Var_Done : std_logic;
  signal w_Sig_Done : std_logic;

  component variable_vs_signal is
    port (
      i_clk      : in  std_logic;
```

design.vhd

```
-- Code your design here
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity variable_vs_signal is
  port (
    i_clk   : in std_logic;
    o_var_done : out std_logic;
    o_sig_done : out std_logic
    );
end variable_vs_signal;

architecture rtl of variable_vs_signal is

  signal r_Var_Done : std_logic            := '0';
  signal r_Count    : natural range 0 to 6 := 0;
  signal r_Sig_Done : std_logic            := '0';

begin
```

```
# SLP: Finished : 0.0 [s]
# ELAB2: You do not have a license to run VHDL performance optimized simulation. Contact Aldec for ordering information - sales@aldec.com.
# ELAB2: Elaboration final pass complete - time: 0.1 [s].
# KERNEL: Warning: You are using the Riviera-PRO EDU Edition. The performance of simulation is reduced.
# KERNEL: Warning: Contact Aldec for available upgrade options - sales@aldec.com.
# KERNEL: Kernel process initialization done.
# Allocation: Simulator allocated 7061 kB (elbread=1023 elab2=5879 kernel=158 sdf=0)
```

design.vhd

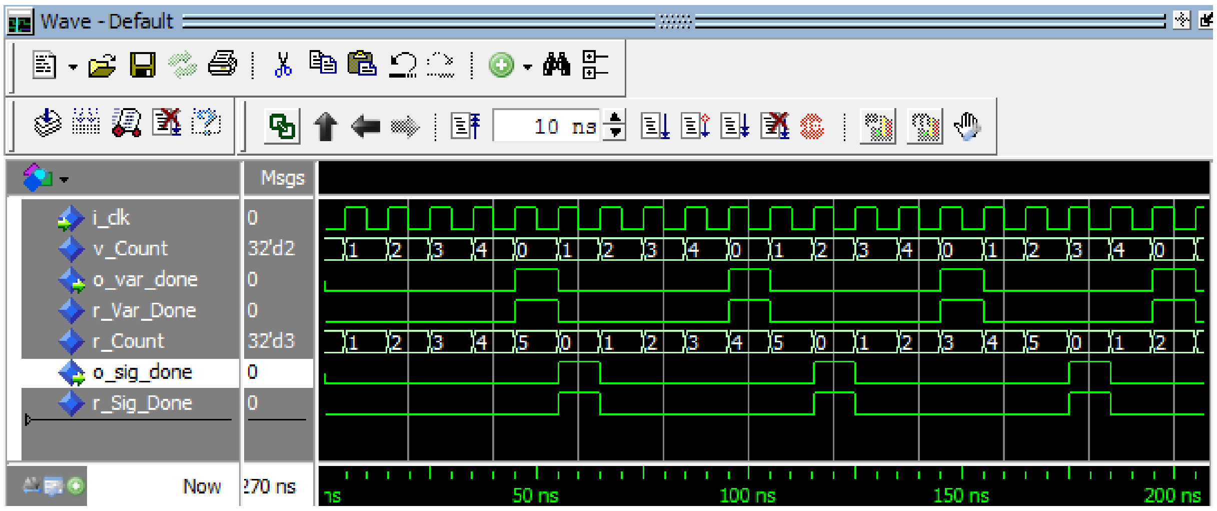
```
-- Code your design here
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity variable_vs_signal is
  port (
    i_clk   : in std_logic;
    o_var_done : out std_logic;
    o_sig_done : out std_logic
    );
end variable_vs_signal;

architecture rtl of variable_vs_signal is

  signal r_Var_Done : std_logic            := '0';
  signal r_Count    : natural range 0 to 6 := 0;
  signal r_Sig_Done : std_logic            := '0';

begin

  VAR_VS_SIG : process (i_clk)
    variable v_Count : natural range 0 to 5 := 0;
  begin
    if rising_edge(i_clk) then
      v_Count := v_Count + 1;        -- Variable
      r_Count <= r_Count + 1;        -- Signal

      -- Variable Checking
```

```vhdl
      if v_Count = 5 then
        r_Var_Done <= '1';
        v_Count := 0;
      else
        r_Var_Done <= '0';
      end if;

      -- Signal Checking
      if r_Count = 5 then
        r_Sig_Done <= '1';
        r_Count    <= 0;
      else
        r_Sig_Done <= '0';
      end if;

    end if;
  end process VAR_VS_SIG;

  o_var_done <= r_Var_Done;
  o_sig_done <= r_Sig_Done;

end rtl;
```

testbench.vhd

```vhdl
-- Code your testbench here
-- or browse Examples
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity variable_vs_signal_tb is
end variable_vs_signal_tb;

architecture behave of variable_vs_signal_tb is

  constant c_CLK_PERIOD : time := 10 ns;

  signal r_Clock    : std_logic := '0';
  signal w_Var_Done : std_logic;
  signal w_Sig_Done : std_logic;

  component variable_vs_signal is
    port (
      i_clk      : in  std_logic;
      o_var_done : out std_logic;
      o_sig_done : out std_logic
      );
  end component variable_vs_signal;

begin

  r_Clock <= not r_Clock after c_CLK_PERIOD/2;
  variable_vs_signal_inst : variable_vs_signal
    port map (
      i_clk      => r_Clock,
      o_var_done => w_Var_Done,
      o_sig_done => w_Sig_Done
      );

end behave;
```

SIMULACIJA



Primer 2 – množenje matrica

Skladištenje matrica (naravno hardcoded vrednosti za `numcols1, numcols2, numcols3...)`

```
type t11 is array (0 to numcols1-1) of unsigned(15 downto 0);
type t1 is array (0 to numrows1-1) of t11;
type t22 is array (0 to numcols2-1) of unsigned(15 downto 0);
type t2 is array (0 to numrows2-1) of t22;
type t33 is array (0 to numcols3-1) of unsigned(31 downto 0);
type t3 is array (0 to numrows3-1) of t33;
```

Na primer, hardcoded verzija:

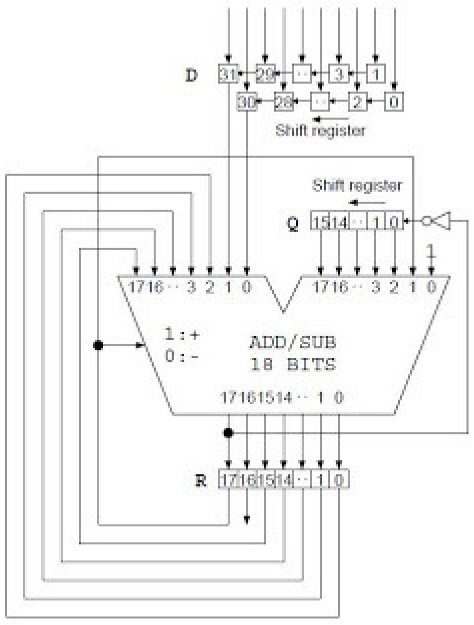
```
type t11 is array (0 to 2) of unsigned(15 downto 0);
type t1 is array (0 to 3) of t11;
type t22 is array (0 to 4) of unsigned(15 downto 0);
type t2 is array (0 to 2) of t22;
type t33 is array (0 to 4) of unsigned(31 downto 0);
type t3 is array (0 to 3) of t33;
```

Algoritam za množenje matrica:

```
function  matmul  ( a : t1; b:t2 ) return t3 is
variable i,j,k : integer:=0;
variable prod : t3:=(others => (others => (others => '0')));
begin
for i in 0 to numrows1-1 loop
for j in 0 to numcols2-1 loop
for k in 0 to numcols1-1 loop
   prod(i)(j) := prod(i)(j) + (a(i)(k) * b(k)(j));
end loop;
end loop;
end loop;
return prod;
end matmul;
```

Primer 3 – numerički recept za nalaženje kvadratnog korena

https://drive.google.com/file/d/1bhOr-KdjjQjOQ13TKr4b6WiEq42trPQj/view



design.vhd

```
library IEEE;
use IEEE.std_logic_1164.all;
use ieee.numeric_std.all;    --  UNSIGNED

function  sqrt  ( d : UNSIGNED ) return UNSIGNED is
variable a : unsigned(31 downto 0):=d;
variable q : unsigned(15 downto 0):=(others => '0');  --resenje.
variable left,right,r : unsigned(17 downto 0):=(others => '0');
variable i : integer:=0;

begin
for i in 0 to 15 loop
right(0):='1';
right(1):=r(17);
right(17 downto 2):=q;
left(1 downto 0):=a(31 downto 30);
left(17 downto 2):=r(15 downto 0);
a(31 downto 2):=a(29 downto 0);  --shift za 2 bita
if ( r(17) = '1') then
r := left + right;
else
r := left - right;
end if;
q(15 downto 1) := q(14 downto 0);
q(0) := not r(17);
end loop;
return q;

end sqrt;
```

Driver koji testira gornju funkciju

```
--primer koriscenja funkcije
signal a : unsigned(31 downto 0) :="00000000000000000000000000110010
";  --50
signal b : unsigned(15 downto 0) :=(others => '0');
b <= sqrt ( a );  --poziv funkcije
--b ce sadrzati vrednosti "00000111" ( tj. vrednost 7) cim je

--operacija obavljena
```