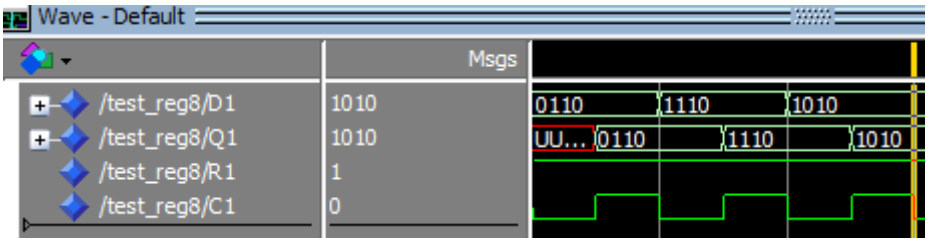
Ερώτημα 1

Κώδικας:

```
Ln#
 1  LIBRARY ieee;
 2  USE ieee.std_logic_1164.all;
 3
 4  entity test_reg8 is
 5  end test_reg8;
 6
 7  architecture test_b of test_reg8 is
 8
 9   signal D1: std_logic_vector(3 downto 0):= "0110";
10   signal Q1: std_logic_vector(3 downto 0);
11   signal R1: std_logic:= '1';
12   signal C1: std_logic;
13
14   component reg8 port(
15   D: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
16   Resetn, Clock: IN STD_LOGIC;
17   Q: OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
18
19  end component;
20
21  begin
22    R8: reg8 PORT MAP (D=> D1, Q=> Q1, Resetn=>R1, Clock=> C1);
23
24  Clock_process : PROCESS
25    BEGIN
26      C1 <= '0';
27      WAIT FOR 5 ps;
28      C1 <= '1';
29      WAIT FOR 5 ps;
30    END PROCESS;
31
32
33    Stimulus_process : PROCESS
34    BEGIN
35
36      WAIT FOR 10 ps;
37      D1 <= "1110";
38      WAIT FOR 10 ps;
39      D1 <= "1010";
40      WAIT FOR 10 ps;
41      WAIT;
42    END PROCESS;
43
44  END test_b;
45
```

Wave



Κώδικας TestBench:

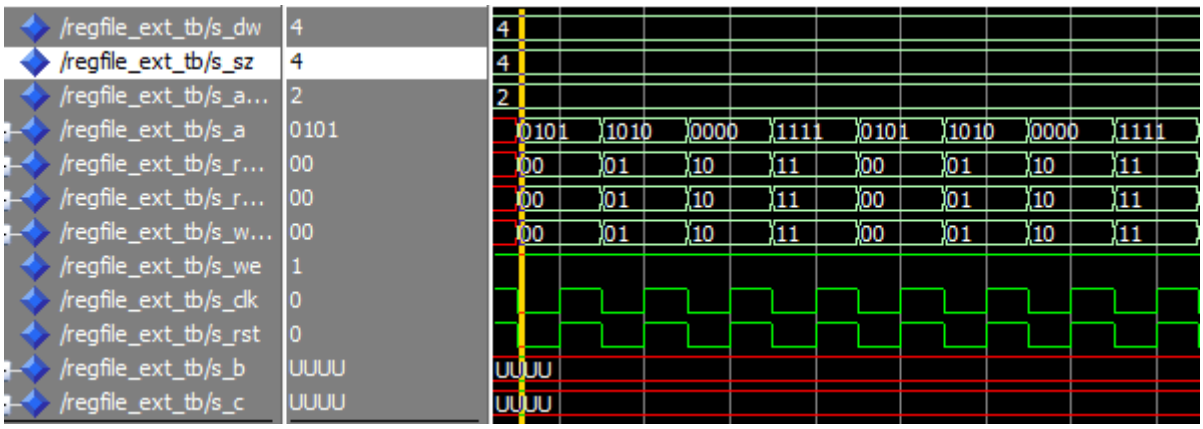
```
Ln#
 1  LIBRARY ieee;
 2  USE ieee.std_logic_1164.all;
 3
 4  entity test_reg8 is
 5  end test_reg8;
 6
 7  architecture test_b of test_reg8 is
 8
 9   signal D1: std_logic_vector(3 downto 0):= "0110";
10   signal Q1: std_logic_vector(3 downto 0);
11   signal R1: std_logic:= '1';
12   signal C1: std_logic;
13
14   component reg8 port(
15   D: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
16   Resetn, Clock: IN STD_LOGIC;
17   Q: OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
18
19  end component;
20
21  begin
22    R8: reg8 PORT MAP (D=> D1, Q=> Q1, Resetn=>R1, Clock=> C1);
23
24  Clock_process : PROCESS
25    BEGIN
26      C1 <= '0';
27      WAIT FOR 5 ps;
28      C1 <= '1';
29      WAIT FOR 5 ps;
30    END PROCESS;
31
32
33    Stimulus_process : PROCESS
34    BEGIN
35
36      WAIT FOR 10 ps;
37      D1 <= "1110";
38      WAIT FOR 10 ps;
39      D1 <= "1010";
40      WAIT FOR 10 ps;
41      WAIT;
42    END PROCESS;
43
44  END test_b;
45
```

Ερώτημα 3

Κώδικας:

```
library ieee;
use ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;
entity regfile is
 generic ( dw : natural := 4;
size : natural := 4;
adrw : natural := 2);
port ( A : in std_logic_vector(dw-1 downto 0);
rAddr1: in std_logic_vector(adrw-1 downto 0);
rAddr2: in std_logic_vector(adrw-1 downto 0);
wAddr : in std_logic_vector(adrw-1 downto 0);
we : in std_logic;
clk : in std_logic;
 reset : in std_logic;
 B : out std_logic_vector(dw-1 downto 0);
 C : out std_logic_vector(dw-1 downto 0));
end regfile;
architecture behavioral of regfile is
type regArray is array(0 to size-1) of std_logic_vector(dw-1 downto 0);
signal regfile : regArray;
begin
process(clk)
begin
 if (clk'event and clk='0') then
  if we='1' then
    regfile(to_integer(unsigned(Addr))) <= A;
  end if;
 end if;
 end process;
C <= regfile(to_integer(unsigned(Addr)));
end behavioral;
```

Wave



Κώδικας TestBench:

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

entity regfile_ext_tb is
end regfile_ext_tb;

architecture behav of regfile_ext_tb is

component regfile_ext is
generic (
 dw: natural := 4;
 sz: natural := 4;
 addrw: natural := 2
);
port (
 a: in std_logic_vector(dw-1 downto 0);
 raddr1: in std_logic_vector(addrw-1 downto 0);
 raddr2: in std_logic_vector(addrw-1 downto 0);
 waddr: in std_logic_vector(addrw-1 downto 0);
 we: in std_logic;
 clk: in std_logic;
 rst: in std_logic;
 b: out std_logic_vector(dw-1 downto 0);
 c: out std_logic_vector(dw-1 downto 0)
);
end component;

signal s_dw: natural := 4;
signal s_sz: natural := 4;
signal s_addrw: natural := 2;
signal s_a: std_logic_vector(s_dw-1 downto 0);
signal s_raddr1:std_logic_vector(s_addrw-1 downto 0);
signal s_raddr2:std_logic_vector(s_addrw-1 downto 0);
signal s_waddr: std_logic_vector(s_addrw-1 downto 0);
signal s_we: std_logic;
signal s_clk: std_logic;
signal s_rst: std_logic;
signal s_b: std_logic_vector(s_dw-1 downto 0);
signal s_c: std_logic_vector(s_dw-1 downto 0);

begin
 uut: regfile_ext port map (
   a => s_a,
   raddr1 => s_raddr1,
   raddr2 => s_raddr2,
   waddr => s_waddr,
   we => s_we,
   clk => s_clk,
   rst => s_rst,
   b => s_b,
   c => s_c
 );
```

```
process begin
  s_we <= '1';
  s_clk <= '1';
  s_rst <= '1';
  wait for 250 ns;

  s_we <= '1';
  s_clk <= '0';
  s_rst <= '0';
  s_raddr1 <= "00";
  s_raddr2 <= "00";
  s_waddr <= "00";
  s_a <= "0101";
  wait for 250 ns;

  s_we <= '1';
  s_clk <= '1';
  s_rst <= '1';
  wait for 250 ns;

  s_we <= '1';
  s_clk <= '0';
  s_rst <= '0';
  s_raddr1 <= "01";
  s_raddr2 <= "01";
  s_waddr <= "01";
  s_a <= "1010";
  wait for 250 ns;

  s_we <= '1';
  s_clk <= '1';
  s_rst <= '1';
  wait for 250 ns;

  s_we <= '1';
  s_clk <= '0';
  s_rst <= '0';
  s_raddr1 <= "10";
  s_raddr2 <= "10";
  s_waddr <= "10";
  s_a <= "0000";
  wait for 250 ns;


   s_we <= '1';
   s_clk <= '1';
   s_rst <= '1';
   wait for 250 ns;

   s_we <= '1';
   s_clk <= '0';
   s_rst <= '0';
   s_raddr1 <= "11";
   s_raddr2 <= "11";
   s_waddr <= "11";
   s_a <= "1111";
   wait for 250 ns;
 end process;
end behav;
```