دانشکده کامپیوتر

آزمایشگاه معماری کامپیوتر
آزمایش ششم

دکتر محبتی

عرشیا آرین نژاد، حوریه سبزواری، الناز رضایی

اردیبهشت ۱۴۰۱

هدف آزمایش:

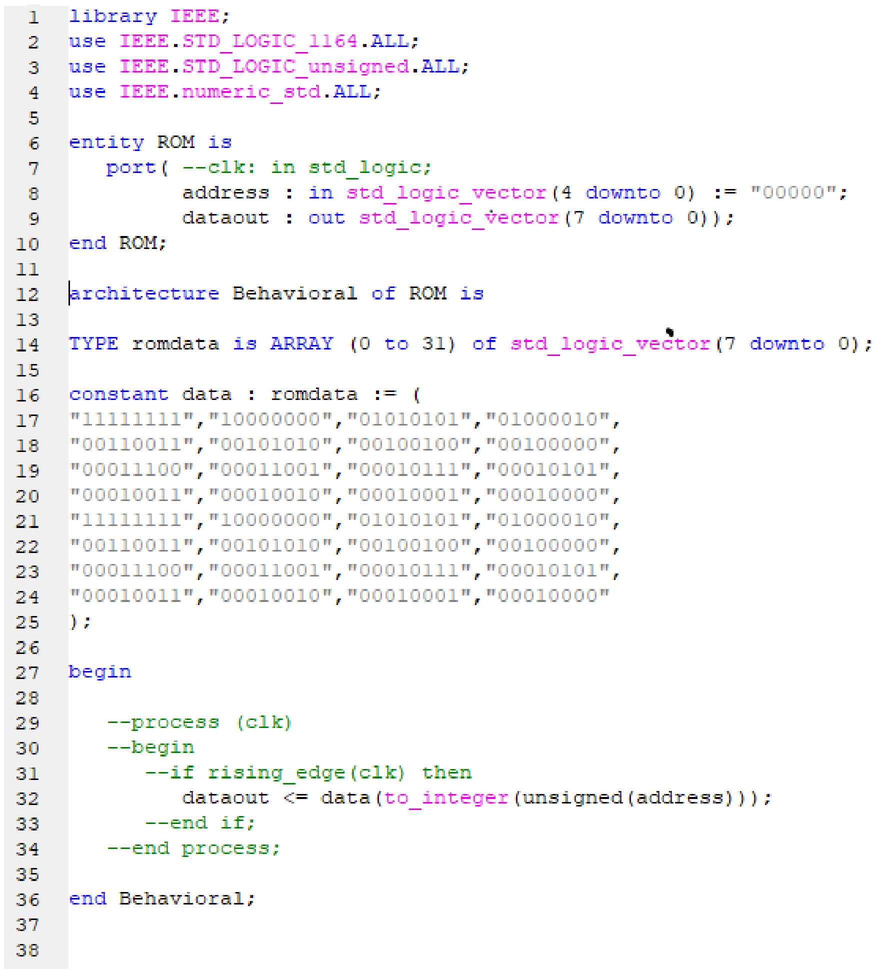
RAM یا Memory Access Random گونه‌ای از حافظه برای ذخیره‌سازی داده‌هاست که اجازه می‌دهد فایل‌ها در مدت زمانی کوتاه نوشته و خوانده شوند؛ بدون اینکه در این خواندن و نوشتن تقدم و تأخر زمانی اهمیتی داشته باشد. حافظه رم به دلیل سرعت بالای آن در خواندن و نوشتن از سایر حافظه‌ها، از جمله دیسک سخت است.

ROM مخفف Memory Read-Only است.ROM برخلاف RAM پایدار است و حتی وقتی کامپیوتر خاموش شود، محتوای رام باقی می‌ماند. رام، چیپ کامپیوتری است که تقریبا همه کامپیوترها مقدار کمی از آن را برای Firmware Boot دارند. فریمور بوت حاوی چند کیلوبایت کد است که به کامپیوتر می‌گوید هنگام روشن شدن چه کار باید انجام دهد مثلا شناسایی سخت افزار و لود کردن سیستم عامل روی رم. در PC، به فریور بوت، بایوس می‌گویند.

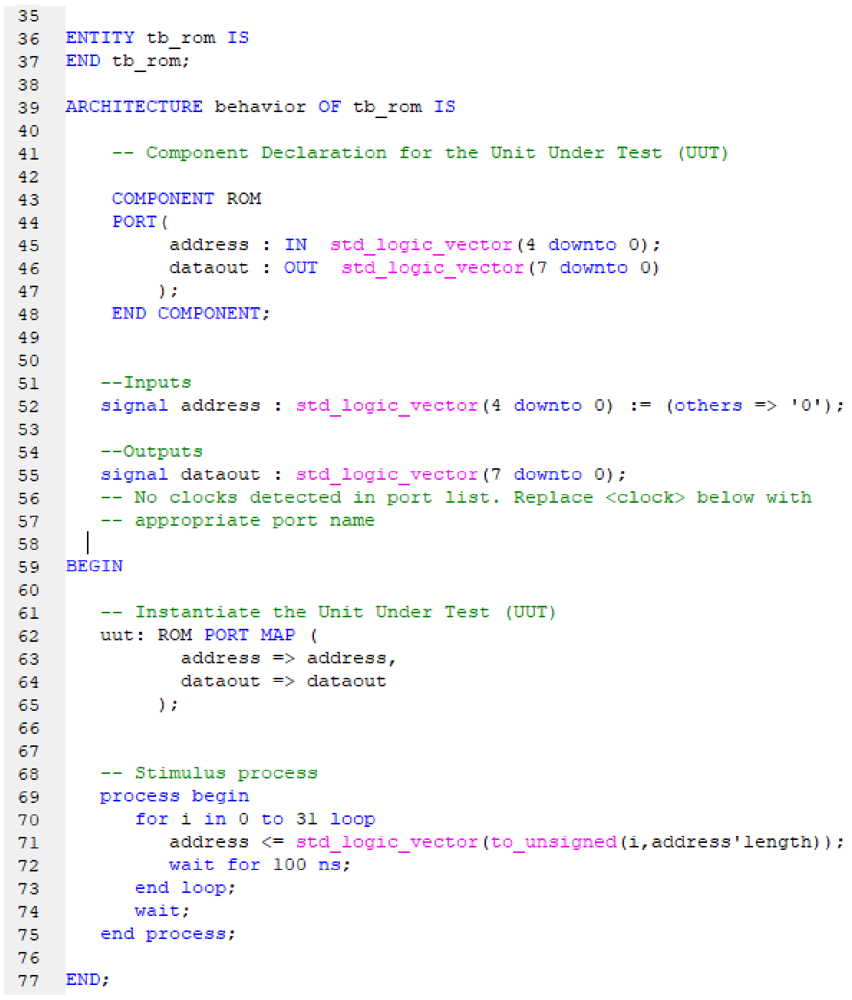
در این آزمایش قصد داریم با پیاده سازی RAM و ROM ها محتویات درون آن‌ها را نمایش دهیم.

ROM:

پیاده‌سازی مدارها با استفاده از زبان‌های توصیف:

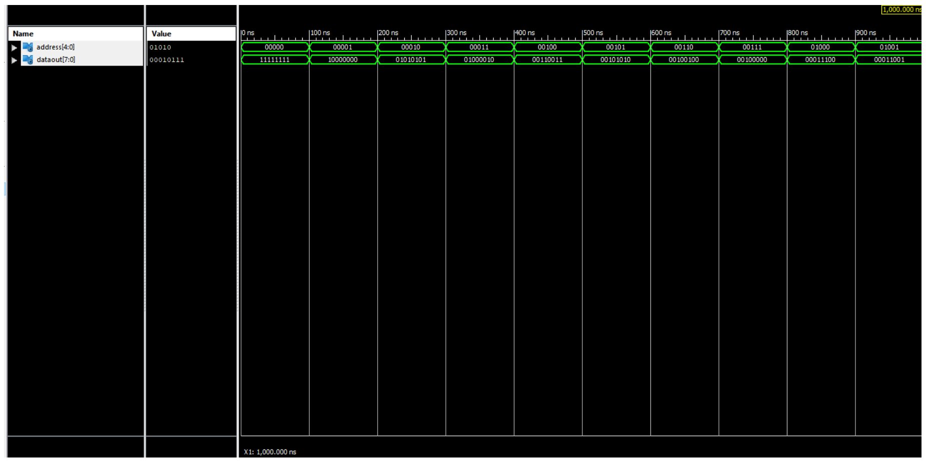


```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;
use IEEE.numeric_std.ALL;

entity ROM is
   port( --clk: in std_logic;
         address : in std_logic_vector(4 downto 0)  := "00000";
         dataout : out std_logic_vector(7 downto 0));
end ROM;

architecture Behavioral of ROM is

TYPE romdata is ARRAY (0 to 31) of std_logic_vector(7 downto 0);

constant data : romdata := (
"11111111","10000000","01010101","01000010",
"00110011","00101010","00100100","00100000",
"00011100","00011001","00010111","00010101",
"00010011","00010010","00010001","00010000",
"11111111","10000000","01010101","01000010",
"00110011","00101010","00100100","00100000",
"00011100","00011001","00010111","00010101",
"00010011","00010010","00010001","00010000"
);

begin

   --process (clk)
   --begin
      --if rising_edge(clk) then
         dataout <= data(to_integer(unsigned(address)));
      --end if;
   --end process;

end Behavioral;
```



```vhdl

ENTITY tb_rom IS
END tb_rom;

ARCHITECTURE behavior OF tb_rom IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT ROM
    PORT(
         address : IN  std_logic_vector(4 downto 0);
         dataout : OUT  std_logic_vector(7 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal address : std_logic_vector(4 downto 0) := (others => '0');

   --Outputs
   signal dataout : std_logic_vector(7 downto 0);
   -- No clocks detected in port list. Replace <clock> below with
   -- appropriate port name

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: ROM PORT MAP (
          address => address,
          dataout => dataout
        );


   -- Stimulus process
   process begin
      for i in 0 to 31 loop
         address <= std_logic_vector(to_unsigned(i,address'length));
         wait for 100 ns;
      end loop;
      wait;
   end process;

END;
```

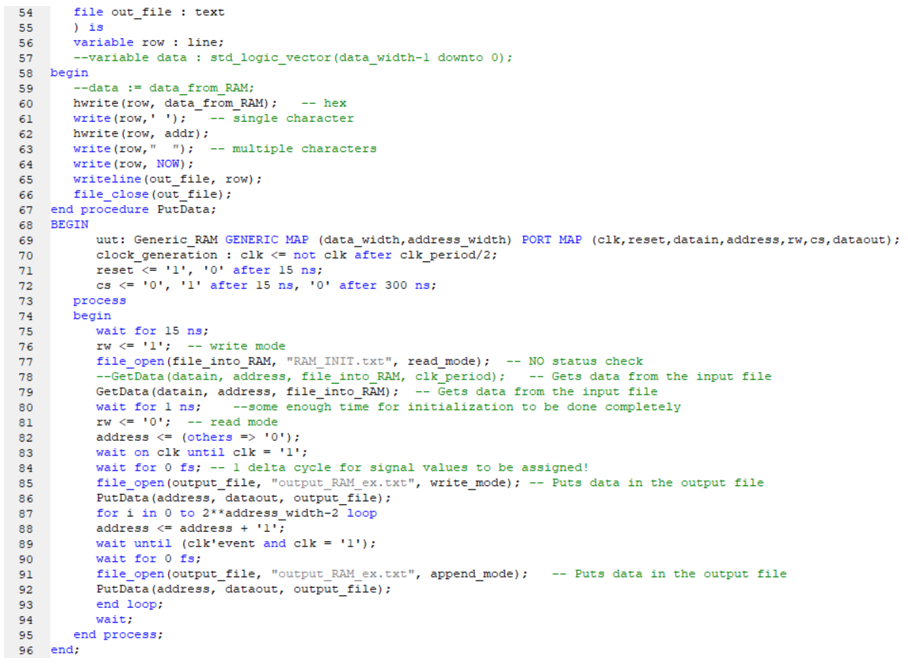
نتایج خروجی:



```
Name            Value      0 ns      100 ns    200 ns    300 ns    400 ns    500 ns    600 ns    700 ns    800 ns    900 ns    1,000.000 ns
address[4:0]    01010      00000     00001     00010     00011     00100     00101     00110     00111     01000     01001
dataout[7:0]    00010111   11111111  10000000  01010101  01000010  00110011  00101010  00100100  00100000  00011100  00011001

X1: 1,000.000 ns
```

RAM:

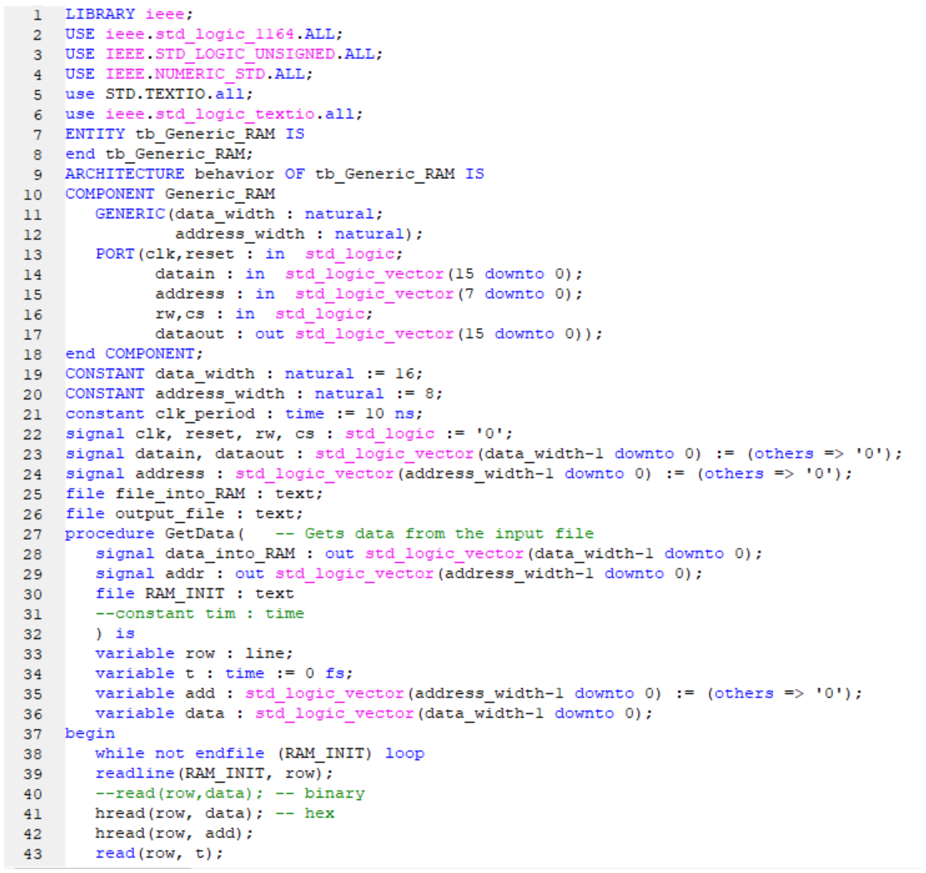
پیاده‌سازی مدارها با استفاده از زبان‌های توصیف:



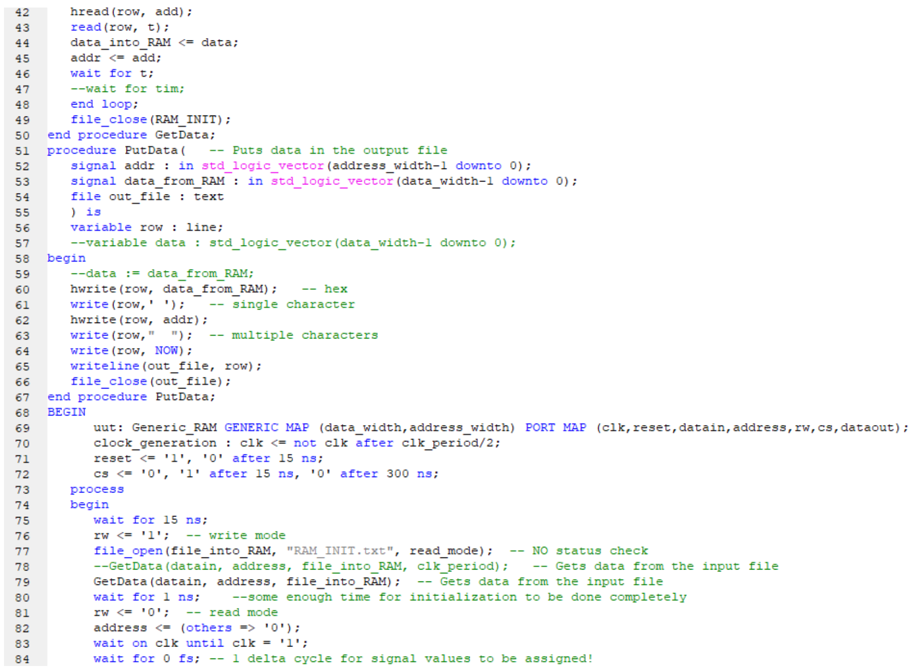
```vhdl
   file out_file : text
   ) is
   variable row : line;
   --variable data : std_logic_vector(data_width-1 downto 0);
begin
   --data := data_from_RAM;
   hwrite(row, data_from_RAM);   -- hex
   write(row,' ');   -- single character
   hwrite(row, addr);
   write(row,"  ");  -- multiple characters
   write(row, NOW);
   writeline(out_file, row);
   file_close(out_file);
end procedure PutData;
BEGIN
      uut: Generic_RAM GENERIC MAP (data_width,address_width) PORT MAP (clk,reset,datain,address,rw,cs,dataout);
      clock_generation : clk <= not clk after clk_period/2;
      reset <= '1', '0' after 15 ns;
      cs <= '0', '1' after 15 ns, '0' after 300 ns;
   process
   begin
      wait for 15 ns;
      rw <= '1';  -- write mode
      file_open(file_into_RAM, "RAM_INIT.txt", read_mode);  -- NO status check
      --GetData(datain, address, file_into_RAM, clk_period);   -- Gets data from the input file
      GetData(datain, address, file_into_RAM);  -- Gets data from the input file
      wait for 1 ns;    --some enough time for initialization to be done completely
      rw <= '0';  -- read mode
      address <= (others => '0');
      wait on clk until clk = '1';
      wait for 0 fs; -- 1 delta cycle for signal values to be assigned!
      file_open(output_file, "output_RAM_ex.txt", write_mode); -- Puts data in the output file
      PutData(address, dataout, output_file);
      for i in 0 to 2**address_width-2 loop
      address <= address + '1';
      wait until (clk'event and clk = '1');
      wait for 0 fs;
      file_open(output_file, "output_RAM_ex.txt", append_mode);   -- Puts data in the output file
      PutData(address, dataout, output_file);
      end loop;
      wait;
   end process;
end;
```

Test bench:

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

entity Generic_RAM is
   generic(
      data_width : natural := 16;
      address_width : natural := 8);
   port(
      clk : in  std_logic;
      reset : in  std_logic;
      datain : in  std_logic_vector(data_width-1 downto 0);
      address : in  std_logic_vector(address_width-1 downto 0);
      rw : in  std_logic;
      cs : in  std_logic;
      dataout : out std_logic_vector(data_width-1 downto 0)
   );
end entity Generic_RAM;
architecture Behavioral of Generic_RAM is
type mem_array is array (2**address_width-1 downto 0) of std_logic_vector(data_width-1 downto 0);
begin
   process (clk)
   variable memory : mem_array;
   begin
      if rising_edge(clk) then
         if reset = '1' then
            for i in memory'range loop
               memory(i) := (others => '0');
            end loop;
            dataout <= (others => '0');
         elsif cs = '1' then
            if rw = '1' then --write in memory "RAM_INIT.txt" the address
               memory(to_integer(unsigned(address))) := datain;
               dataout <= datain;
            else              --read from memory "RAM_INIT.txt" the address
               dataout <= memory(to_integer(unsigned(address)));
            end if;
         end if;
      end if;
   end process;
end Behavioral;
```

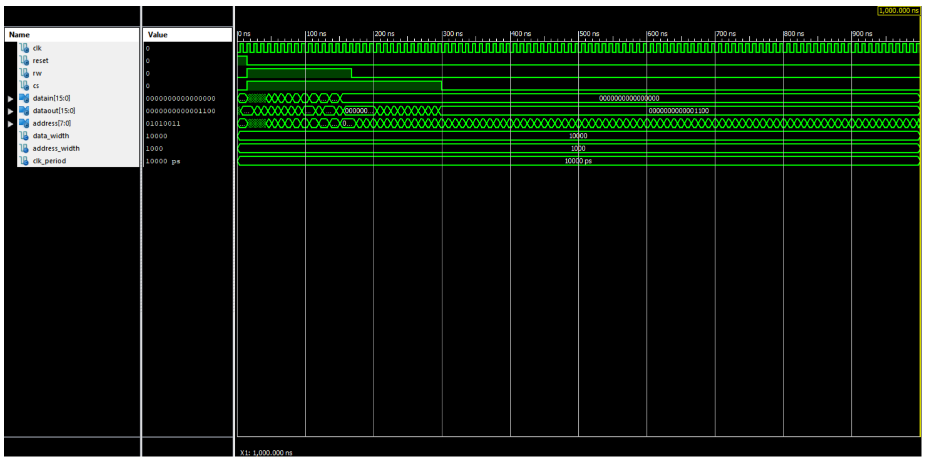


```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE IEEE.NUMERIC_STD.ALL;
use STD.TEXTIO.all;
use ieee.std_logic_textio.all;
ENTITY tb_Generic_RAM IS
end tb_Generic_RAM;
ARCHITECTURE behavior OF tb_Generic_RAM IS
COMPONENT Generic_RAM
   GENERIC(data_width : natural;
           address_width : natural);
   PORT(clk,reset : in  std_logic;
         datain : in  std_logic_vector(15 downto 0);
         address : in  std_logic_vector(7 downto 0);
         rw,cs : in  std_logic;
         dataout : out std_logic_vector(15 downto 0));
end COMPONENT;
CONSTANT data_width : natural := 16;
CONSTANT address_width : natural := 8;
constant clk_period : time := 10 ns;
signal clk, reset, rw, cs : std_logic := '0';
signal datain, dataout : std_logic_vector(data_width-1 downto 0) := (others => '0');
signal address : std_logic_vector(address_width-1 downto 0) := (others => '0');
file file_into_RAM : text;
file output_file : text;
procedure GetData(   -- Gets data from the input file
   signal data_into_RAM : out std_logic_vector(data_width-1 downto 0);
   signal addr : out std_logic_vector(address_width-1 downto 0);
   file RAM_INIT : text
   --constant tim : time
   ) is
   variable row : line;
   variable t : time := 0 fs;
   variable add : std_logic_vector(address_width-1 downto 0) := (others => '0');
   variable data : std_logic_vector(data_width-1 downto 0);
begin
   while not endfile (RAM_INIT) loop
   readline(RAM_INIT, row);
   --read(row,data); -- binary
   hread(row, data); -- hex
   hread(row, add);
   read(row, t);
```



```
   hread(row, add);
   read(row, t);
   data_into_RAM <= data;
   addr <= add;
   wait for t;
   --wait for tim;
   end loop;
   file_close(RAM_INIT);
end procedure GetData;
procedure PutData(   -- Puts data in the output file
   signal addr : in std_logic_vector(address_width-1 downto 0);
   signal data_from_RAM : in std_logic_vector(data_width-1 downto 0);
   file out_file : text
   ) is
   variable row : line;
   --variable data : std_logic_vector(data_width-1 downto 0);
begin
   --data := data_from_RAM;
   hwrite(row, data_from_RAM);   -- hex
   write(row,' ');   -- single character
   hwrite(row, addr);
   write(row,"  ");  -- multiple characters
   write(row, NOW);
   writeline(out_file, row);
   file_close(out_file);
end procedure PutData;
BEGIN
      uut: Generic_RAM GENERIC MAP (data_width,address_width) PORT MAP (clk,reset,datain,address,rw,cs,dataout);
      clock_generation : clk <= not clk after clk_period/2;
      reset <= '1', '0' after 15 ns;
      cs <= '0', '1' after 15 ns, '0' after 300 ns;
   process
   begin
      wait for 15 ns;
      rw <= '1';  -- write mode
      file_open(file_into_RAM, "RAM_INIT.txt", read_mode);  -- NO status check
      --GetData(datain, address, file_into_RAM, clk_period);   -- Gets data from the input file
      GetData(datain, address, file_into_RAM);  -- Gets data from the input file
      wait for 1 ns;    --some enough time for initialization to be done completely
      rw <= '0';  -- read mode
      address <= (others => '0');
      wait on clk until clk = '1';
      wait for 0 fs; -- 1 delta cycle for signal values to be assigned!
```

نمونه خروجی:



Name	Value
clk	0
reset	0
rw	0
cs	0
datain[15:0]	0000000000000000
dataout[15:0]	0000000000001100
address[7:0]	01010011
data_width	10000
address_width	1000
clk_period	10000 ps