گزارشکار آزمایش ۷

: RAM

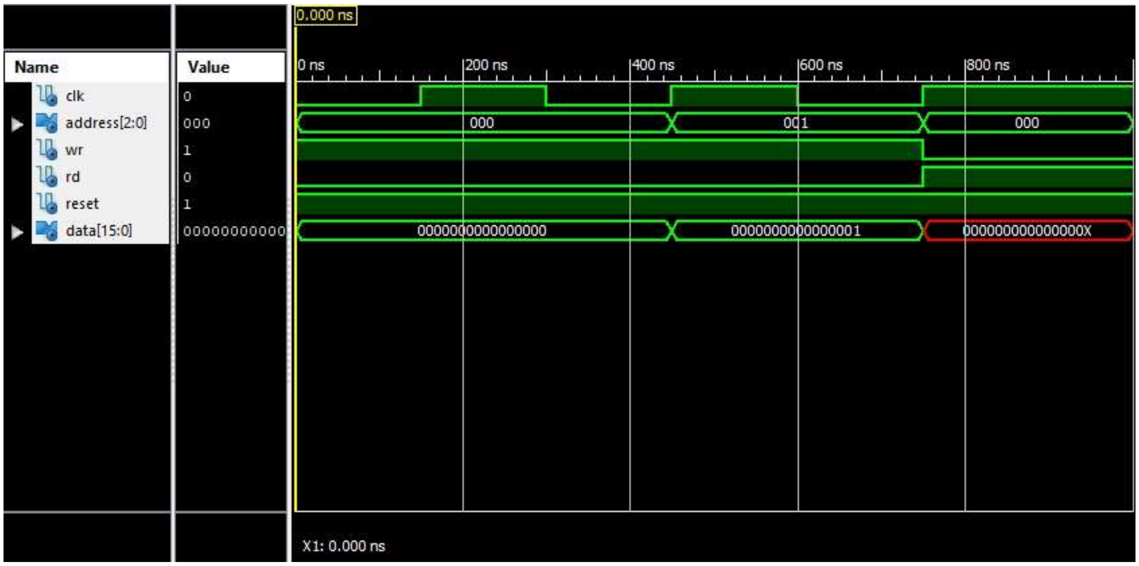
```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use ieee.math_real.all;

use ieee.Numeric_Std.all;

use ieee.std_logic_unsigned.all;

entity RAM is

 generic (

    W :integer := 16;   -- data width

   C :integer := 3   -- addr width

     );

port (

        clk     :in    std_logic;                              -- Clock Input

        address :in    std_logic_vector (C-1 downto 0);         -- address Input 3 bit

        data    :inout std_logic_vector (W-1 downto 0);         -- input or output data

        WR      :in    std_logic;                              -- Write Enable

        RD      :in    std_logic;                              -- Read Enable

        reset   :in    std_logic

     );

end RAM;

architecture rt1 of RAM is

-- Internal variables

    signal data_out :std_logic_vector (W-1 downto 0);

    constant D :integer := 2**C;    -- ram size

    type RAM is array (0 to D-1 )of std_logic_vector (W-1 downto 0);

     signal RAM_1: RAM := (others=>(others=>'0'));

begin

     data <= data_out when (reset ='1' and RD ='1') else(others =>'Z');

            process (reset,clk )
        begin
        --  RESET_state:
         if(reset = '0' ) then
        -- some code --
         end if;
   -- RAM_1 Write Block
   -- Write Operation : When wr = 1, reset = 1
      if (rising_edge(clk)) then
        if (reset = '1' and WR = '1') then
          RAM_1(conv_integer(address)) <= data;
        end if;
           end if;
   -- RAM_1 Read Block
   -- Read Operation : When rd = 1, reset = 1
                       if (rising_edge(clk)) then
        if (reset = '1' and RD = '1') then
           data_out <=  RAM_1(conv_integer(address));
        end if;
           end if;
               end process;
end rt1;
```

: RAM_tb

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY RAM_tb IS
END RAM_tb;

ARCHITECTURE test OF RAM_tb IS
```

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

   COMPONENT RAM

   PORT(

       clk : IN  std_logic;

       address : IN  std_logic_vector(2 downto 0);

        data  :inout std_logic_vector (15 downto 0);

       WR : IN  std_logic;

       RD : IN  std_logic;

       reset : IN  std_logic

      );

   END COMPONENT;

  --Inputs

  signal clk : std_logic;

  signal address : std_logic_vector(2 downto 0);

  signal WR : std_logic ;

  signal RD : std_logic ;

  signal reset : std_logic ;

          signal data : std_logic_vector(15 downto 0);

BEGIN

          -- Instantiate the Unit Under Test (UUT)

  uut: RAM PORT MAP (

        clk => clk,

        address => address,

        data=> data,

        WR => WR,

        RD => RD,

        reset => reset

      );

 clk <='0','1' after 150 ns , '0' after 300 ns , '1' after 450 ns ,'0' after 600 ns, '1' after 750 ns;
 address <= "000" ,"001" after 450 ns ,"000" after 750 ns;

 data <= "0000000000000000" ,"0000000000000001" after 450 ns;

 WR <= '1' ,'0' after 750 ns;

 RD <= '0' , '1' after 750 ns;

 reset <= '1';

END;
```

خروجی شبیه سازی RAM_tb :



: RAM_DUAL

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.math_real.all;
use ieee.Numeric_Std.all;
use ieee.std_logic_unsigned.all;
entity RAM_DUAL is
 generic (
   W :integer := 16;   -- data width
   C :integer := 3   -- addr width
     );
port (
        clk     :in    std_logic;                              -- Clock Input
        address :in    std_logic_vector (C-1 downto 0);         -- address Input 3 bit
        datai  :in std_logic_vector (W-1 downto 0);            -- input data
        WR      :in    std_logic;                              -- Write Enable
        RD      :in    std_logic;                              -- Read Enable
       reset   :in    std_logic;
       datao  :out std_logic_vector (W-1 downto 0)              -- output data
     );
end RAM_DUAL;
architecture rt1 of RAM_DUAL is
-- Internal variables
      signal data_out :std_logic_vector (W-1 downto 0);
     constant D :integer := 2**C;    -- ram size
     type RAM_DUAL is array (0 to D-1 )of std_logic_vector (W-1 downto 0);
     signal RAM_DUAL_1: RAM_DUAL := (others=>(others=>'0'));
begin
     datao <= data_out when (reset ='1' and RD ='1') else(others =>'X');
```

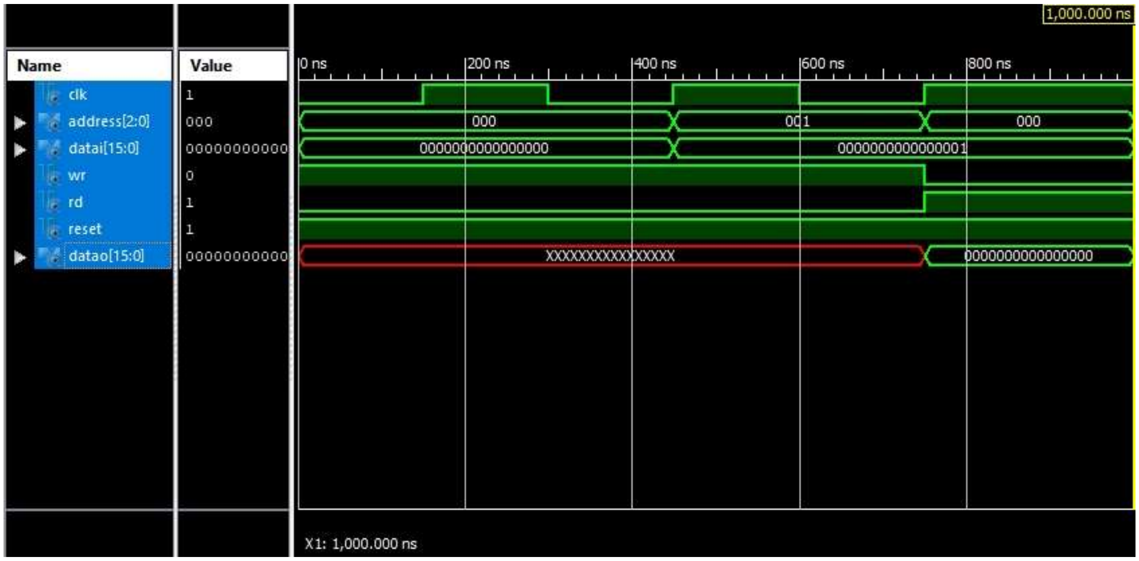
```
       process (reset,clk )
       begin
       --  RESET_state:
        if(reset = '0') then
        ------ some code -------------
        end if;
   -- RAM_1 Write Block
   -- Write Operation : When wr = 1, reset = 1
     if (rising_edge(clk)) then
        if (reset = '1' and WR = '1') then
          RAM_DUAL_1(conv_integer(address)) <= datai;
        end if;
              end if;
   -- RAM_1 Read Block
   -- Read Operation : When rd = 1, reset = 1
if (rising_edge(clk)) then
        if (reset = '1' and RD = '1') then
           data_out <=  RAM_DUAL_1(conv_integer(address));
        end if;
          end if;
              end process;
end rt1;
```

: RAM_DUAL_tb

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY RAM_DUAL_tb IS
END RAM_DUAL_tb;
```

```
ARCHITECTURE test OF RAM_DUAL_tb IS
  -- Component Declaration for the Unit Under Test (UUT)
  COMPONENT RAM_DUAL
  PORT(
     clk : IN  std_logic;
     address : IN  std_logic_vector(2 downto 0);
     datai : IN  std_logic_vector(15 downto 0);
     WR : IN  std_logic;
     RD : IN  std_logic;
     reset : IN  std_logic;
                    datao : OUT  std_logic_vector(15 downto 0)
    );
  END COMPONENT;
 --Inputs
 signal clk : std_logic ;
 signal address : std_logic_vector(2 downto 0);
 signal datai : std_logic_vector(15 downto 0);
 signal WR : std_logic ;
 signal RD : std_logic ;
 signal reset : std_logic;
        --Outputs
 signal datao : std_logic_vector(15 downto 0);
BEGIN
       -- Instantiate the Unit Under Test (UUT)
 uut: RAM_DUAL PORT MAP (
     clk => clk,
     address => address,
     datai => datai,
     datao => datao,
        WR => WR,

        RD => RD,

        reset => reset

      );

   clk <='0','1' after 150 ns , '0' after 300 ns , '1' after 450 ns ,'0' after 600 ns, '1' after 750 ns;

   address <= "000" ,"001" after 450 ns ,"000" after 750 ns;

   datai <= "0000000000000000" ,"0000000000000001" after 450 ns;

   WR <= '1' ,'0' after 750 ns;

   RD <= '0' , '1' after 750 ns;

   reset <= '1';

END;
```

خروجی شبیه سازی RAM_DUAL_tb :



Name	Value
clk	1
address[2:0]	000
datai[15:0]	00000000000
wr	0
rd	1
reset	1
datao[15:0]	00000000000

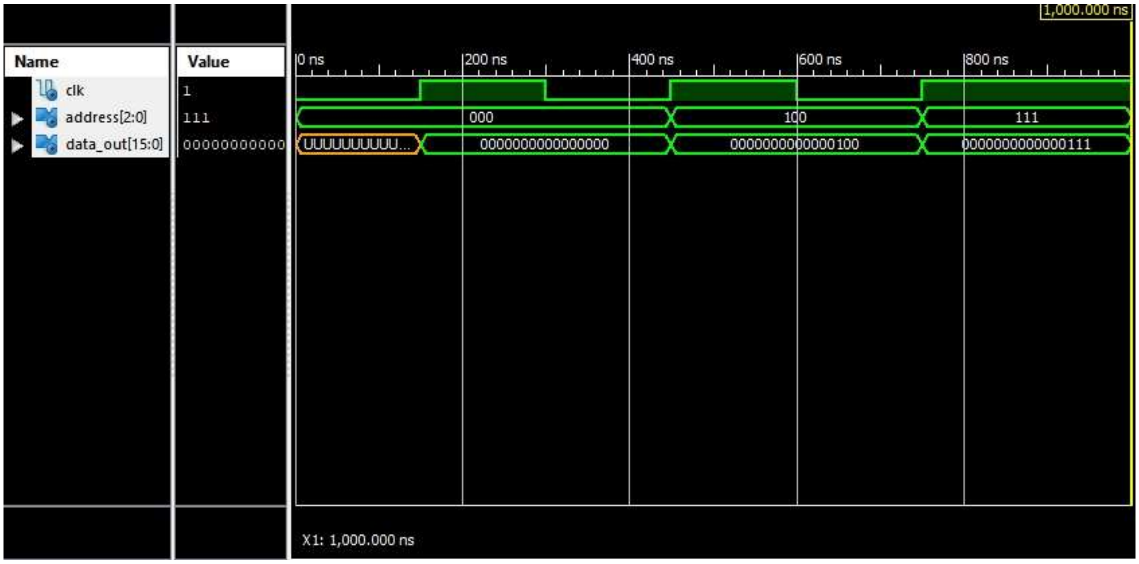
: ROM

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE ieee.std_logic_unsigned.all;
entity ROM is
port ( clk : in std_logic ;
    address : in std_logic_vector( 2 downto 0 );
    data_out : out std_logic_vector( 15 downto 0 )) ;
end entity ;
architecture arch of ROM is
        type ROM is array ( 0 to 7 ) of std_logic_vector( 15 downto 0 ) ;
        constant ROM_1 : ROM := (
        0 => "0000000000000000" ,
        1 => "0000000000000001" ,
        2 => "0000000000000010" ,
        3 => "0000000000000011" ,
        4 => "0000000000000100" ,
        5 => "0000000000000101" ,
        6 => "0000000000000110" ,
        7 => "0000000000000111" ) ;
        begin
        process(clk)
                begin
                        if( clk'event and clk = '1' ) then
                                data_out <= ROM_1(conv_integer(address)) ;
                end if ;
end process ;
end architecture ;
```

: ROM_tb

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY ROM_tb IS
END ROM_tb;
ARCHITECTURE behavior OF ROM_tb IS
  -- Component Declaration for the Unit Under Test (UUT)
  COMPONENT ROM
  PORT(
     clk : IN  std_logic;
     address : IN  std_logic_vector(2 downto 0);
     data_out : OUT  std_logic_vector(15 downto 0)
    );
  END COMPONENT;
 --Inputs
 signal clk : std_logic ;
 signal address : std_logic_vector(2 downto 0) := (others => '0');
--Outputs
 signal data_out : std_logic_vector(15 downto 0);
BEGIN
       -- Instantiate the Unit Under Test (UUT)
 uut: ROM PORT MAP (
      clk => clk,
      address => address,
      data_out => data_out
    );
clk <= '0' ,'1' after 150 ns , '0' after 300 ns , '1' after 450 ns , '0' after 600 ns , '1' after 750 ns;
address <="000" ,"100" after 450 ns , "111" after 750 ns;
END;
```

خروجی شبیه سازی ROM_tb :


1,000.000 ns
Name
Value
0 ns
200 ns
400 ns
600 ns
800 ns
clk
1
address[2:0]
111
000
100
111
data_out[15:0]
00000000000
UUUUUUUUUU...
0000000000000000
0000000000000100
0000000000000111
X1: 1,000.000 ns