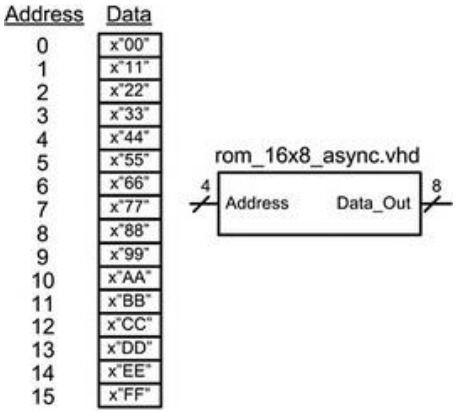
6. Меморије

Задатак 10.4.1
Написати VHDL модел понашања за 16×8 асинхрону ROM меморијски систем приказан на слици 10.22. Систем треба да садржи информације дате у меморијској мапи. Креирати тест бенч за симулацију модела читањем са сваке од 16 јединствених адреса и посматрањем излаза Data_Out да би се проверило да ли меморијски систем садржи информације у меморијској мапи.



Слика. 10.22 16×8 асинхрони ROM блок дијаграм

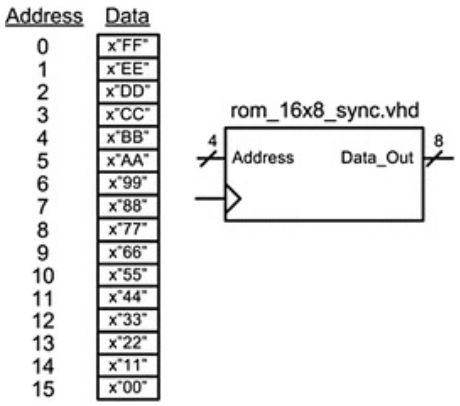
Решење задатка 10.4.1

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

entity rom_16x8_async is
      port (address  : in std_logic_vector(3 downto 0);
            data_out : out std_logic_vector(7 downto 0));
end entity;

architecture rom_16x8_async_arch of rom_16x8_async is
      type ROM_type is array (0 to 15) of std_logic_vector(7 downto 0);

      constant ROM : ROM_type := (0  => x”00”,
                                  1  => x”11”,
                                  2  => x”22”,
                                  3  => x”33”,
                                  4  => x”44”,
                                  5  => x”55”,
                                  6  => x”66”,
                                  7  => x”77”,
                                  8  => x”88”,
                                  9  => x”99”,
                                  10 => x”AA”,
                                  11 => x”BB”,
                                  12 => x”CC”,
                                  13 => x”DD”,
                                  14 => x”EE”,
                                  15 => x”FF”);
      begin

            data_out <= ROM(to_integer(unsigned(address)));
end architecture;

entity rom_16x8_async_TB is
end entity;
```

```
architecture rom_16x8_async_TB_arch of rom_16x8_async_TB is
      component rom_16x8_async
      port (address  : in std_logic_vector(3 downto 0);
            data_out : out std_logic_vector(7 downto 0));
      end component;
            signal address_TB  : std_logic_vector(3 downto 0);
            signal data_out_TB : std_logic_vector(7 downto 0);
      begin
      DUT1: rom_16x8_async port map (address => address_TB,
                                     data_out => data_out_TB);

      STIMULUS : process
      signal i : integer;
      begin
            for i in 0 to 15 loop
                  address_TB = std_logic_vector(to_unsigned(i,4));
                  report "adress= " & address_TB'image &
                         " data_out= " & data_out_TB'image;
            end loop;
      end process;
end architecture;
```

Задатак 10.4.2

Написати VHDL модел понашања за 16×8 синхрону ROM меморијски систем приказан на слици 10.23. Систем треба да садржи информације дате у меморијској мапи. Креирати тест бенч за симулацију модела читањем са сваке од 16 јединствених адреса и посматрањем излаза Data_Out да би се проверило да ли меморијски систем садржи информације у меморијској мапи.



Слика. 10.22 16×8 синхрони ROM блок дијаграм

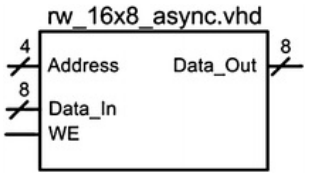
Решење задатка 10.4.2

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

entity rom_16x8_sync is
      port (clock    : in std_logic;
            address  : in std_logic_vector(3 downto 0);
            data_out : out std_logic_vector(7 downto 0));
end entity;

architecture rom_16x8_sync_arch of rom_16x8_sync is
      type ROM_type is array (0 to 15) of std_logic_vector(7 downto 0);

      constant ROM : ROM_type := (0  => x"FF",
                                  1  => x"EE",
                                  2  => x"DD",
                                  3  => x"CC",
```

```
                                    4  => x"BB",
                                    5  => x"AA",
                                    6  => x"99",
                                    7  => x"88",
                                    8  => x"77",
                                    9  => x"66",
                                    10 => x"55",
                                    11 => x"44",
                                    12 => x"33",
                                    13 => x"22",
                                    14 => x"11",
                                    15 => x"00");
      begin

            MEMORY : process (clock)
                  begin
                        if (clock'event and clock='1') then
                              data_out <= ROM(to_integer(unsigned(address)));
                        end if;
            end process;

end architecture;

entity rom_16x8_sync_TB is
end entity;

architecture rom_16x8_sync_TB_arch of rom_16x8_sync_TB is
      component rom_16x8_sync
      port (clock : std_logic;
            address  : in std_logic_vector(3 downto 0);
            data_out : out std_logic_vector(7 downto 0));
      end component;
            signal clock_TB    : std_logic;
            signal address_TB  : std_logic_vector(3 downto 0);
            signal data_out_TB : std_logic_vector(7 downto 0);
      begin
      DUT1: rom_16x8_sync port map (clock => clock_TB,
                                    address => address_TB,
                                    data_out => data_out_TB);

      CLOCK_PROCESS : process
      begin
            clock_TB <= '0'; wait for 5 ns;
            clock_TB <= '1'; wait for 5 ns;
      end process;

      STIMULUS : process
      signal i : integer;
      begin
            for i in 0 to 15 loop
                  wait until rising_edge(clock_TB);
                  address_TB = std_logic_vector(to_unsigned(i,4));
                  wait until rising_edge(clock_TB);
                  report "adress= " & address_TB'image &
                         " data_out= " & data_out_TB'image;
            end loop;
      end process;
end architecture;
```

Задатак 10.4.3

Написати VHDL модел понашања за 16×8 асинхрону читај/упиши (R/W) меморијски систем приказан на слици 10.24. Креирати тест бенч за симулацију модела. Тест бенч треба да чита са свих меморијских локација да би проверио да оне нису иницијализоване. Затим, тест бенч треба да упише јединствене информације у сваку локацију (адресу). На крају, тест бенч треба да прочита сваку локацију како би верификовао да је информација која је уписана сачувана и да може да се прочита из меморије.



Слика. 10.24 16×8 асинхрони R/W меморијски блок дијаграм

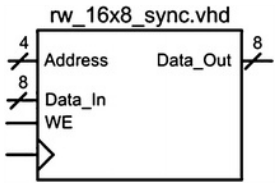
Решење задатка 10.4.3

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

entity rw_16x8_async is
      port (address  : in std_logic_vector(3 downto 0);
            data_in  : in std_logic_vector(7 downto 0));
            WE       : in std_logic;
            data_out : out std_logic_vector(7 downto 0));
end entity;

architecture rw_16x8_async_arch of rw_16x8_async is
      type RW_type is array (0 to 15) of std_logic_vector(7 downto 0);
      signal RW : RW_type;

      begin

            MEMORY : process (address, WE, data_in)
                  begin
                        if (WE = '1') then
                              RW(to_integer(unsigned(address))) <= data_in;
                        else
                              data_out <= RW(to_integer(unsigned(address)));
                        end if;
            end process;

end architecture;

entity rw_16x8_async_TB is
end entity;

architecture rw_16x8_async_TB_arch of rw_16x8_async_TB is
      component rw_16x8_async
      port (address  : in std_logic_vector(3 downto 0);
            data_in  : in std_logic_vector(7 downto 0));
            WE       : in std_logic;
            data_out : out std_logic_vector(7 downto 0));
      end component;
            signal address_TB  : std_logic_vector(3 downto 0);
            signal data_in_TB  : std_logic_vector(7 downto 0);
            WE_TB              : std_logic;
            signal data_out_TB : std_logic_vector(7 downto 0);
      begin
      DUT1: rw_16x8_sync port map (address => address_TB,
                                   data_in => data_in_TB,
```

```
                                    WE => WE_TB,
                                    data_out => data_out_TB);

      STIMULUS : process
      signal i : integer;
      begin
            WE_TB <= 0;
            for i in 0 to 15 loop
                  report "adress= " & address_TB'image &
                        " data_out= " & data_out_TB'image;
            end loop;
            wait for 40 ns;
            WE_TB <= 1;
            for i in 0 to 15 loop
                  address_TB <= std_logic_vector(to_unsigned(i,4));
                  data_in_TB <= std_logic_vector(to_unsigned(i,8));
            end loop;
            wait for 120 ns;
            WE_TB <= 0;
            for i in 0 to 15 loop
                  report "adress= " & address_TB'image &
                        " data_out= " & data_out_TB'image;
            end loop;
            wait for 200 ns;
      end process;
end architecture;
```

Задатак 10.4.4

Написати VHDL модел понашања за 16×8 синхрону читај/упиши (R/W) меморијски систем приказан на слици 10.25. Креирати тест бенч за симулацију модела. Тест бенч треба да чита са свих меморијских локација да би проверио да оне нису иницијализоване. Затим, тест бенч треба да упише јединствене информације у сваку локацију (адресу). На крају, тест бенч треба да прочита сваку локацију како би верификовао да је информација која је уписана сачувана и да може да се прочита из меморије.



Слика. 10.24 16×8 синхрони R/W меморијски блок дијаграм

Решење задатка 10.4.3

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

entity rw_16x8_sync is
      port (clock    : in std_logic;
            address  : in std_logic_vector(3 downto 0);
            data_in  : in std_logic_vector(7 downto 0));
            WE       : in std_logic;
            data_out : out std_logic_vector(7 downto 0));
end entity;

architecture rw_16x8_sync_arch of rw_16x8_sync is
      type RW_type is array (0 to 15) of std_logic_vector(7 downto 0);
      signal RW : RW_type;

      begin

            MEMORY : process (clock)
```

```
            begin
                  if (clock'event and clock='1') then
                        if (WE = '1') then
                              RW(to_integer(unsigned(address))) <= data_in;
                        else
                              data_out <= RW(to_integer(unsigned(address)));
                        end if;
                  end if;
            end process;

end architecture;

entity rw_16x8_sync_TB is
end entity;

architecture rw_16x8_sync_TB_arch of rw_16x8_sync_TB is
      component rw_16x8_sync
      port (clcock   : in std_logic;
            address  : in std_logic_vector(3 downto 0);
            data_in  : in std_logic_vector(7 downto 0);
            WE       : in std_logic;
            data_out : out std_logic_vector(7 downto 0));
      end component;
            signal clock_TB    : std_logic;
            signal address_TB  : std_logic_vector(3 downto 0);
            signal data_in_TB  : std_logic_vector(7 downto 0);
            WE_TB              : std_logic;
            signal data_out_TB : std_logic_vector(7 downto 0);
      begin
      DUT1: rw_16x8_sync port map (clock => clock_TB,
                                   address => address_TB,
                                   data_in => data_in_TB,
                                   WE => WE_TB,
                                   data_out => data_out_TB);
      CLOCK_PROCESS : process
      begin
            clock_TB <= '0'; wait for 5 ns;
            clock_TB <= '1'; wait for 5 ns;
      end process;

      STIMULUS : process
      signal i : integer;
      begin
            WE_TB <= 0;
            for i in 0 to 15 loop
                  wait until rising_edge(clock_TB);
                  address_TB = std_logic_vector(to_unsigned(i,4));
                  wait until rising_edge(clock_TB);
                  report "adress= " & address_TB'image &
                   " data_out= " & data_out_TB'image;
            end loop;
            wait for 80 ns;
            WE_TB <= 1;
            for i in 0 to 15 loop
                  wait until rising_edge(clock_TB);
                  address_TB <= std_logic_vector(to_unsigned(i,4));
                  wait until rising_edge(clock_TB);
                  data_in_TB <= std_logic_vector(to_unsigned(i,8));
            end loop;
            wait for 80 ns;
            WE_TB <= 0;
            for i in 0 to 15 loop
                  wait until rising_edge(clock_TB);
                  address_TB = std_logic_vector(to_unsigned(i,4));
                  wait until rising_edge(clock_TB);
```

```
                report "adress= " & address_TB'image &
                 " data_out= " & data_out_TB'image;
            end loop;
            wait for 80 ns;
      end process;

end architecture;
```