طراحی سیستم های دیجیتال
تکلیف سوم

استاد مروستی

علیرضا نعمتی

کسری رشیدفر

Table of Contents

Question1

Code

Dec_counter

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

```
entity dec_counter is
    port (
    clk, reset: in std_logic;
    en: in std_logic;
    q: out std_logic_vector(3 downto 0);
    pulse: out std_logic);
end dec_counter;

architecture arch of dec_counter is
    signal r_reg: unsigned(3 downto 0):= (others => '0');
    signal r_next: unsigned(3 downto 0):= (others => '0');
    constant TEN: integer:= 10;

    begin
        process(clk, reset)
        begin
            if(reset = '1') then
                r_reg <= (others => '0');
            elsif(clk'event and clk = '1') then
                r_reg <= r_next;
            end if;
        end process;

        process(en, r_reg)
        begin
            r_next <= r_reg;
            if(en = '1') then
                if r_reg = (TEN - 1) then
                    r_next <= (others => '0');
                else
                    r_next <= r_reg + 1;
                end if;
            end if;
        end process;

        q <= std_logic_vector(r_reg);
```

```
    pulse <= '1' when r_reg = (TEN - 1) else
             '0';

    end arch;
```

N_dec_counter

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


entity n_dec_counter is
  generic (N: natural:= 4);
  port (
  clk, reset: in std_logic;
  en: in std_logic;
  q_n: out std_logic_vector(0 to (4*N)-1);
  p: out std_logic
  );
end n_dec_counter;

architecture counter_arch of n_dec_counter is
  component dec_counter
    port(
    clk, reset: in std_logic;
    en: in std_logic;
    q: out std_logic_vector(3 downto 0);
    pulse: out std_logic);
  end component;

  signal p_n: std_logic_vector(0 to N):= (0 => '1',others => '0');
  signal tmp: std_logic_vector(0 to N);

  begin
    tmp(0) <= p_n(0);
```

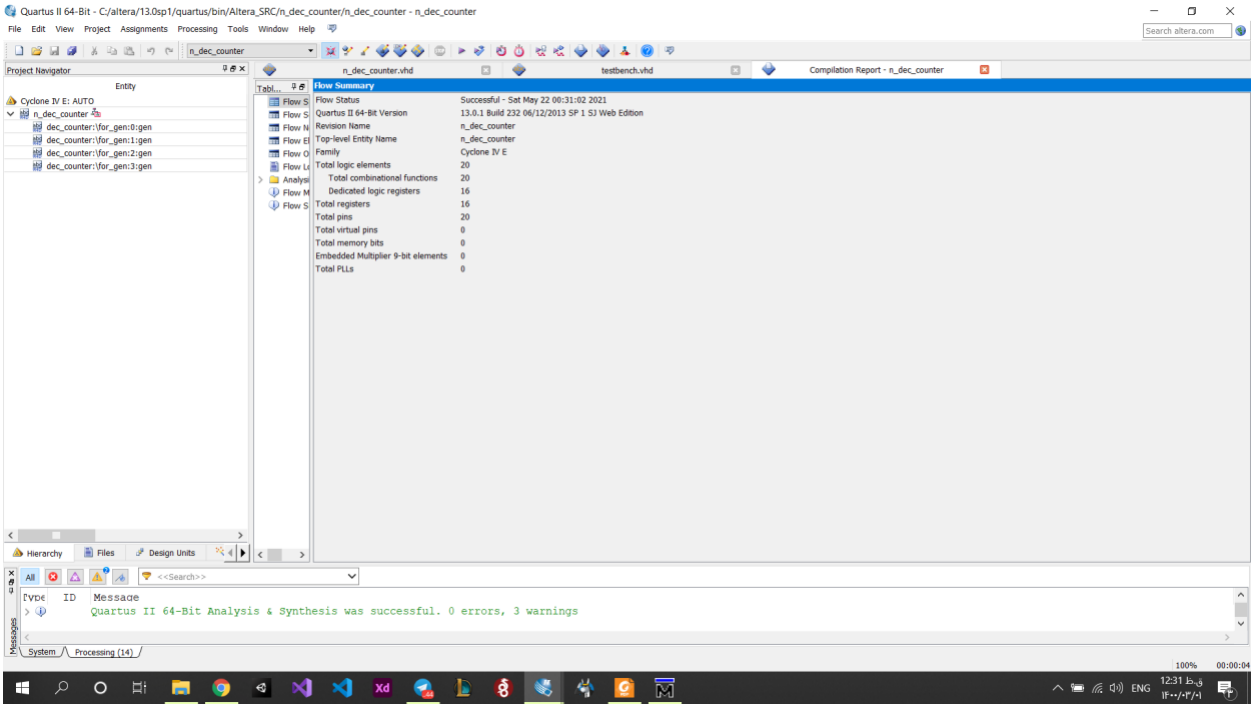
```
    for_gen: for i in 0 to N-1 generate
        gen: dec_counter
            port map (clk => clk, reset => reset, en => p_n(i), q => q_n(4*i to 4*i+3), pulse => p_n(i+1));
    end generate for_gen;

    p_gen: for j in 1 to N-1 generate
        tmp(j) <= p_n(j) and tmp(j-1);
    end generate p_gen;
    p <= tmp(N-1);
end counter_arch;
```

testbench

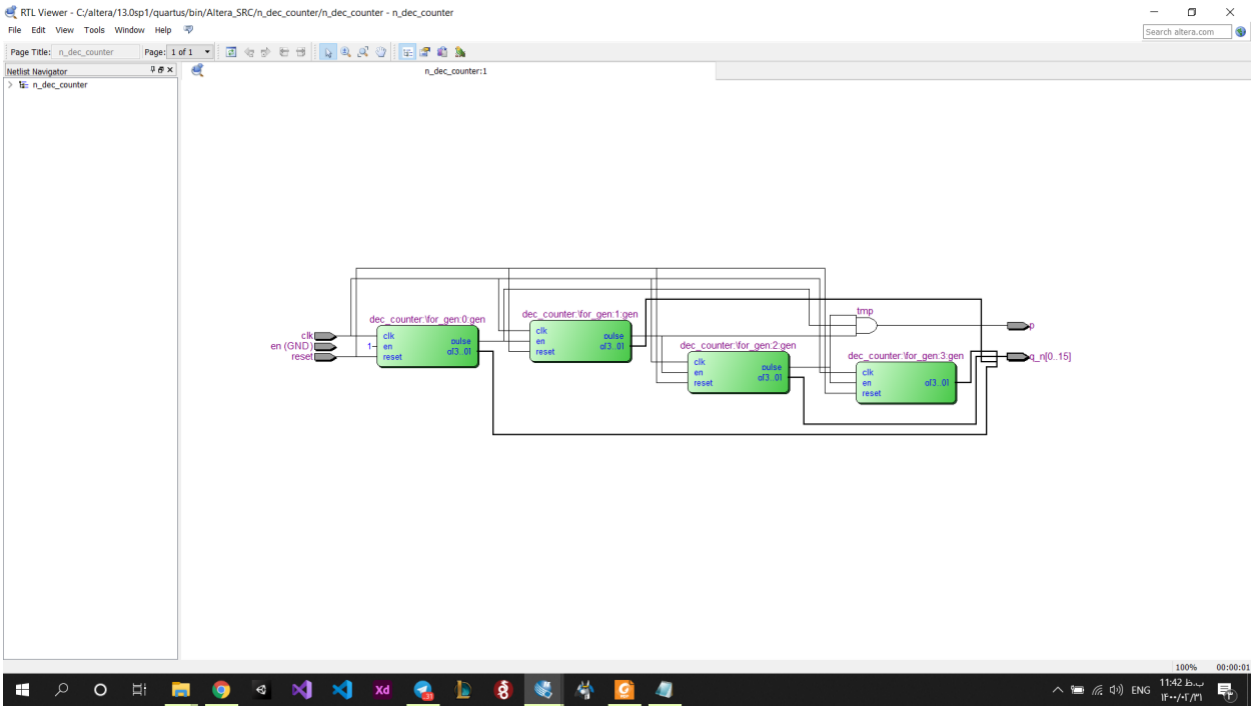
```
library ieee;
use ieee.std_logic_1164.all;

entity testbench is

end testbench;

architecture arch of testbench is

    component n_dec_counter is
        generic (N: natural);
        port (
            clk, reset: in std_logic;
            en: in std_logic;
            q_n: out std_logic_vector(4*N-1 downto 0);
            p: out std_logic);
    end component;

    constant N: natural:= 4;
    signal clk: std_logic:= '0';
    signal reset: std_logic:= '0';
    signal en: std_logic:= '1';
    signal q: std_logic_vector(4*N-1 downto 0);
```

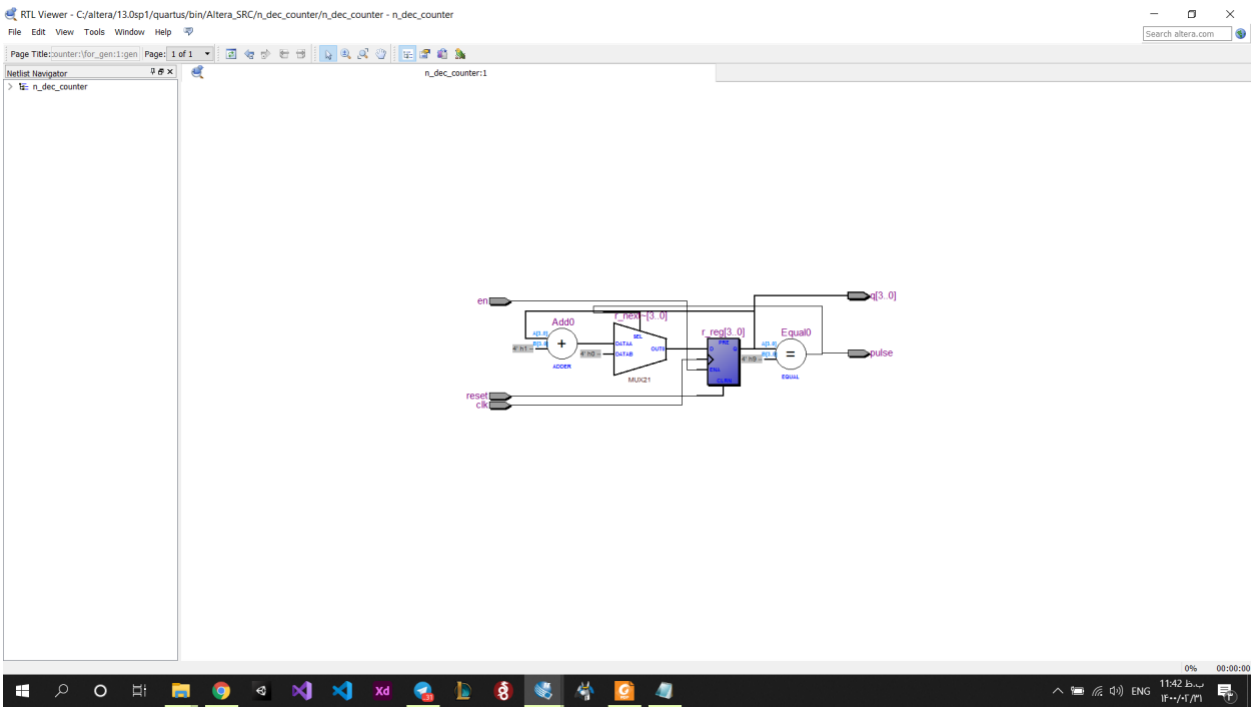
```
signal p: std_logic:= '0';

begin
    uut: n_dec_counter
        generic map (N => N)
        port map (
            clk => clk,
            reset => reset,
            en => en,
            q_n => q,
            p => p);

process
    begin
        clk <= '1';
        wait for 15 ns;
        clk <= '0';
        wait for 15 ns;
        clk <= '1';
        wait for 15 ns;
        clk <= '0';
        wait for 15 ns;
        clk <= '1';
        wait for 15 ns;
        clk <= '0';
        wait for 15 ns;
end process;

end arch;
```

Compilation report

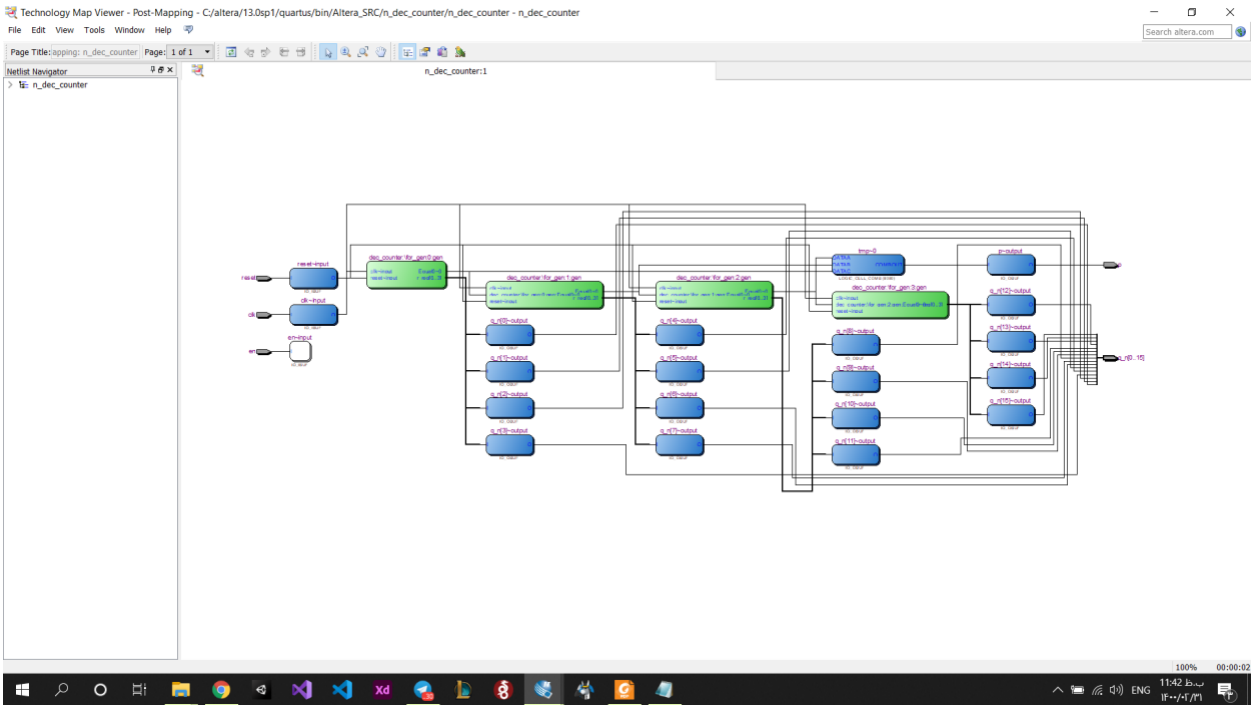

Quartus II 64-Bit - C:/altera/13.0sp1/quartus/bin/Altera_SRC/n_dec_counter/n_dec_counter - n_dec_counter
File Edit View Project Assignments Processing Tools Window Help
Project Navigator
Entity
Cyclone IV E: AUTO
n_dec_counter
dec_counter:\for_gen:0:gen
dec_counter:\for_gen:1:gen
dec_counter:\for_gen:2:gen
dec_counter:\for_gen:3:gen
n_dec_counter.vhd
testbench.vhd
Compilation Report - n_dec_counter
Flow Summary
Flow Status Successful - Sat May 22 00:31:02 2021
Quartus II 64-Bit Version 13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name n_dec_counter
Top-level Entity Name n_dec_counter
Family Cyclone IV E
Total logic elements 20
Total combinational functions 20
Dedicated logic registers 16
Total registers 16
Total pins 20
Total virtual pins 0
Total memory bits 0
Embedded Multiplier 9-bit elements 0
Total PLLs 0
Hierarchy Files Design Units
Type ID Message
Quartus II 64-Bit Analysis & Synthesis was successful. 0 errors, 3 warnings
System Processing (14)
Messages
100% 00:00:04

RTL


RTL Viewer - C:/altera/13.0sp1/quartus/bin/Altera_SRC/n_dec_counter/n_dec_counter - n_dec_counter
File Edit View Tools Window Help
Page Title: n_dec_counter
Page: 1 of 1
Netlist Navigator
n_dec_counter
n_dec_counter:1
clk
en (GND)
reset
dec_counter:\for_gen:0:gen
dec_counter:\for_gen:1:gen
dec_counter:\for_gen:2:gen
dec_counter:\for_gen:3:gen
tmp
q
q_n[0..15]
100% 00:00:01

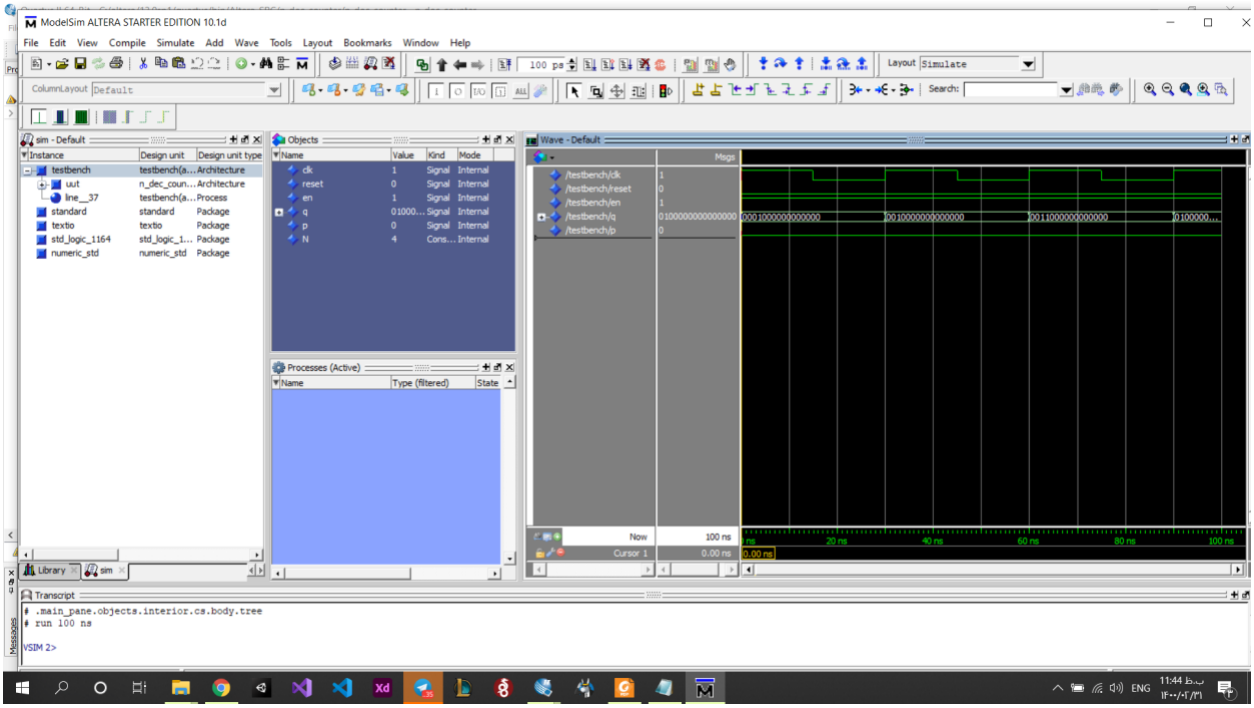


Post-Mapping



Simulation

اسکرین شات



توضیح عملکرد

در هر کلاک شمارنده یک مجموعه بیت نمایش داده میشود که هر 4 بیت آن نمایانگر یک رقم از 4 رقم عدد میباشد.

Question2

Code

Fifo controller

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

```
-- use IEEE.STD_LOGIC_UNSIGNED.all;
entity fifo_controller is
    generic(
        address_width : integer:= 2
    );
    port (
        clk, reset  : in std_logic;
        wr,rd       : in std_logic;
        full, empty : out std_logic;
        w_address,r_address : out std_logic_vector (address_width-1 downto 0) );
end fifo_controller;

architecture rtl of fifo_controller is
        signal reader_pointer : unsigned(address_width downto 0) := "000";
        signal writer_pointer : unsigned(address_width downto 0) := "000";
        -- signal reader_pointer : std_logic_vector(3 downto 0);
        -- signal writer_pointer : std_logic_vector(3 downto 0);
        signal full_fifo      : std_logic;
        signal empty_fifo     : std_logic;



    begin
       process(wr,clk,reset)
           begin
              if(reset = '1') then
                    -- reader_pointer <= "0000";
                    writer_pointer <= (others => '0');
                    full_fifo      <= '0';
                    -- empty_fifo     <= '0';
              elsif(clk'event and clk ='1') then
                 if(wr = '1')then
                      if((reader_pointer(1 downto 0) = writer_pointer(1 downto 0)) and (reader_pointer(2) /=
writer_pointer(2)))then
                             full_fifo <= '1';
                      elsif(writer_pointer = "111") then
                         writer_pointer <= "000";
                      else
```

```
                writer_pointer <= writer_pointer + 1;
            end if;
            if(full_fifo = '1') then
                full <= '1';
            else
                -- i don't know what todod
                -- maybe send w_address
                w_address <= std_logic_vector(writer_pointer(address_width-1 downto 0));
                -- w_address <= writer_pointer(2 downto 0);
            end if;
        end if;
    end if;
end process;

process(rd,clk,reset)
    begin
        if(reset = '1') then
        reader_pointer <= "000";
        -- writer_pointer <= "0000";
        -- full_fifo      <= '0';
        empty_fifo     <= '0';
        elsif(clk'event and clk ='1') then
            if(rd = '1') then
                if(reader_pointer = writer_pointer) then
                    empty_fifo <= '1';
                elsif(reader_pointer = "111") then
                    reader_pointer <= "000";
            else
                reader_pointer <= reader_pointer + 1;
            end if;
            if(empty_fifo = '1') then
                empty <= '1';
            else
                -- i don't know what todod
                -- maybe send r_address
                r_address <= std_logic_vector(reader_pointer(address_width-1 downto 0));
                -- r_address <= reader_pointer(2 downto 0);
```

```
                end if;
            end if;
        end if;
    end process;



end architecture;
```

Fifo buffer

```
library ieee;
use ieee.std_logic_1164.all;
entity fifo_buffer is
    generic (
        data_width    : integer:= 8;
        address_width : integer:= 2
    );
    port(
        clk,reset : in std_logic;
        wr_en : in std_logic;
        w_address,r_address : in std_logic_vector (address_width-1 downto 0);
        w_data : in std_logic_vector(data_width-1  downto 0);
        r_data : out std_logic_vector(data_width-1 downto 0)
    );
end fifo_buffer;

architecture rtl of fifo_buffer is
    constant BIT_ADDR : natural := address_width;
    constant BIT_DATA : natural := data_width;
    type reg_file_type is array (2**BIT_ADDR-1 downto 0) of
        std_logic_vector (BIT_DATA-1 downto 0);
    signal array_reg : reg_file_type;
```

```
signal array_next : reg_file_type;
signal en : std_logic_vector(2**BIT_ADDR-1 downto 0);
begin
process(clk,reset)
begin
   if(reset = '1') then
      array_reg(3) <= (others => '0');
      array_reg(2) <= (others => '0');
      array_reg(1) <= (others => '0');
      array_reg(0) <= (others => '0');
   elsif ( clk'event and clk='1') then
      array_reg(3) <= array_next(3);
      array_reg(2) <= array_next(2);
      array_reg(1) <= array_next(1);
      array_reg(0) <= array_next(0);
   end if;
end process;

process(array_reg ,en,w_data)
begin
      array_next(3) <= array_reg(3);
      array_next(2) <= array_reg(2);
      array_next(1) <= array_reg(1);
      array_next(0) <= array_reg(0);
      if en(3)='1' then
         array_next(3) <= w_data;
      end if;
      if en(2)='1' then
         array_next(2) <= w_data;
      end if;
      if en(1)='1' then
         array_next(1) <= w_data;
      end if;
      if en(0)='1' then
         array_next(0) <= w_data;
      end if;
end process;
```

```
    process(wr_en,w_address)
    begin
        if( wr_en = '0' ) then
            en <= (others => '0');
        else
            case w_address is
                when "00"   => en <= "0001" ;
                when "01"   => en <= "0010" ;
                when "10"   => en <= "0100" ;
                when others => en <= "1000" ;
            end case;
        end if;
    end process;

    with r_address select
                r_data <= array_reg(0) when "00",
                          array_reg(1) when "01",
                          array_reg(2) when "10",
                          array_reg(3) when others;
end rtl;
```

Fifo

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

entity fifo is
    generic(
        address_width : integer :=2;
        data_width : integer := 8
    );
    port(
        clk,reset : in std_logic;
```

```
        w_data : in std_logic_vector(data_width-1  downto 0);
        wr,rd       : in std_logic;
        r_data : out std_logic_vector(data_width-1 downto 0)
        full, empty : out std_logic
    );
end fifo;

architecture rtl of fifo is
    signal w_address : std_logic_vector(address_width-1 downto 0);
    signal r_address : std_logic_vector(address_width-1 downto 0);
    signal full_storage : std_logic ;
    begin
        controller : entity work.fifo_controller(rtl)
        generic map(address_width => address_width)
        port map(
            wr    => wr,
            rd    => rd,
            full  => full_storage,
            empty => empty,
            w_address => w_address,
            r_address => r_address,
            clk => clk,
            reset => reset
        );

        buffer_storage : entity work.fifo_buffer(rtl)
        generic map(
            data_width => data_width,
            address_width => address_width
        )
        port map(
            wr_en => full_storage and wr,
            w_address => w_address,
            r_address => r_address,
            w_data => w_data,
            r_data => r_data,
            clk => clk,
```

```
        reset => reset
    );
end architecture;
```

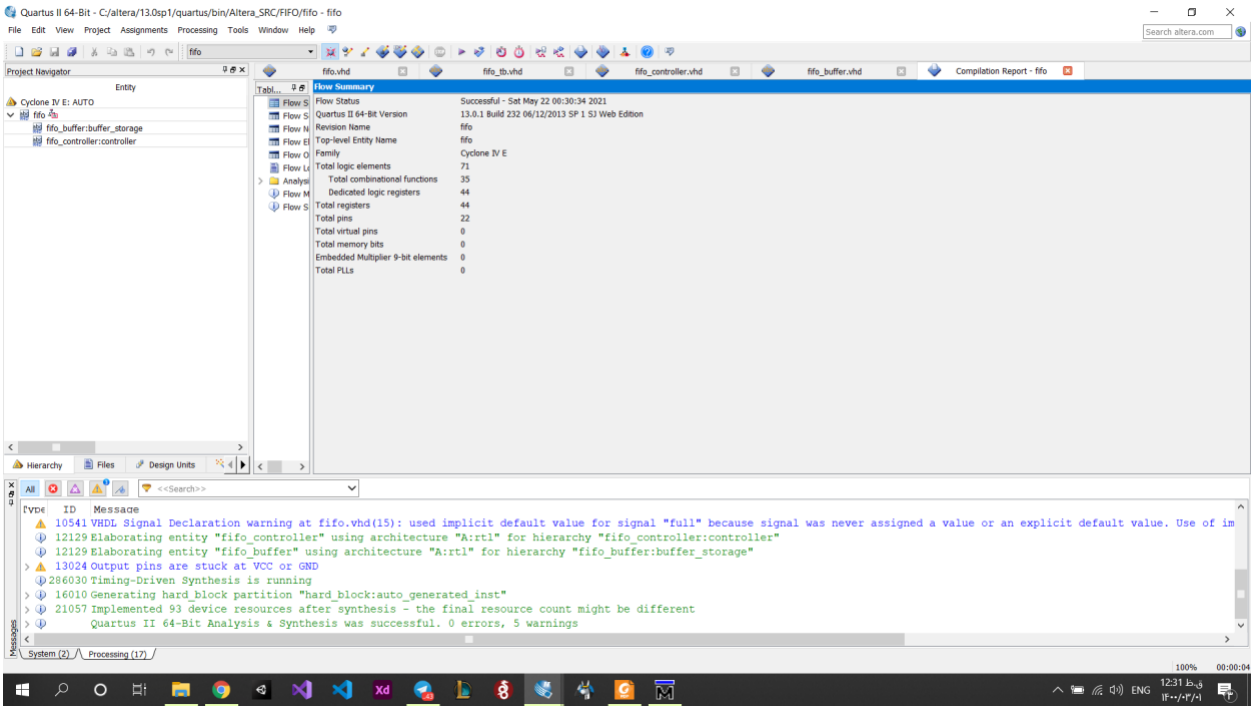
Fifo Test Bench

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

entity fifo_tb is
end entity;

architecture sim of fifo_tb is
    constant address_width: integer := 2;
    constant data_width: integer := 8;

    signal wr : std_logic := '0';
    signal rd : std_logic := '0';
    signal full: std_logic := '0';
    signal empty : std_logic := '1';
    signal w_address,r_address : std_logic_vector(address_width-1 downto 0) := (others => '0');
    signal w_data : std_logic_vector(data_width-1 downto 0) := (others => '0');
    signal r_data : std_logic_vector(data_width-1 downto 0);
    signal clk : std_logic := '1';
    signal reset : std_logic := '1';

    begin
        fifo_component : entity work.fifo(rtl)
        generic map(
            address_width => address_width;
            data_width => data_width

        )
        port map(
            clk => clk,
```

```
    reset => reset,
    w_data => w_data,
    wr    => wr,
    rd    => rd,
    r_data => r_data,
    full  => full,
    empty => empty
);
-- clock and data process
process is
begin
    wait for 20 ns ;
    clk <= '0';
    w_data <=  "00000001" ;
    wait for 20 ns;
    clk <= '1';
    w_data <=  "00000001" ;

end process;

--res
process is
begin
    reset <= '1';
    wait for 8 * 20 ns;
end process;

process is
begin
     wait for 20 ns;
     wr <= '1';
     rd <= '0';
     wait for 20 ns;
     wr <= '0';
     rd <= '1';
end process;
```

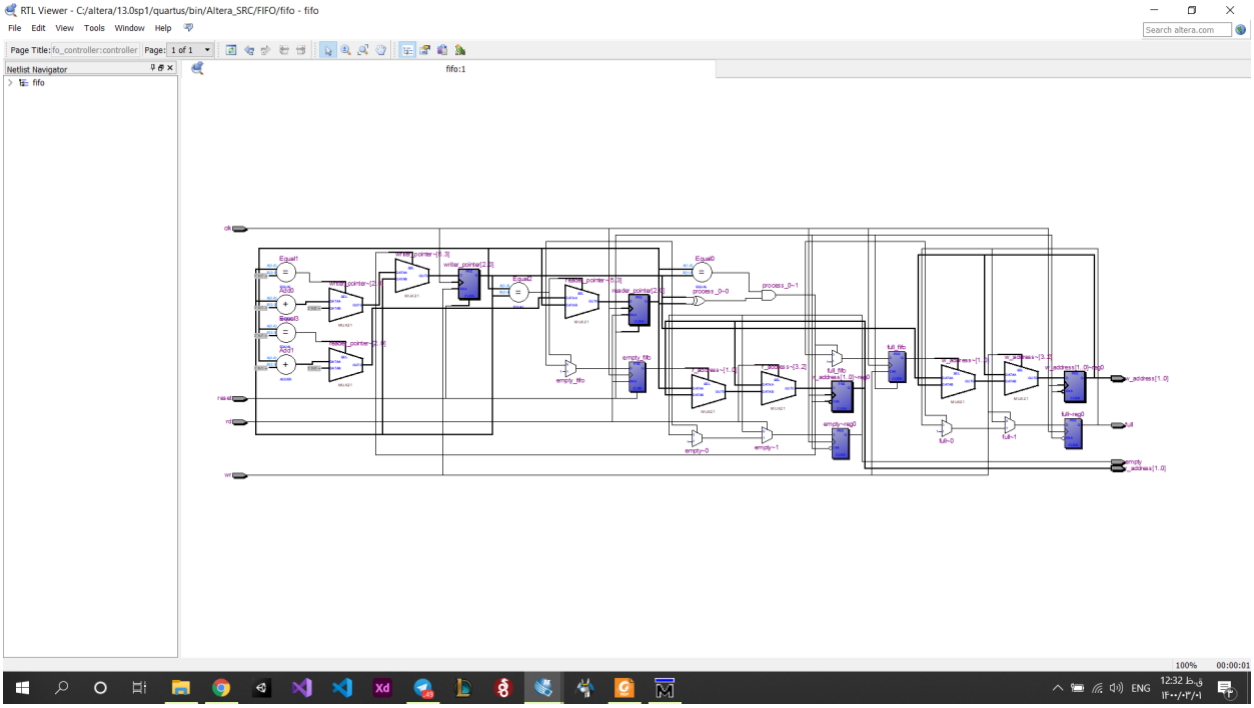
```
end architecture;
```

Compilation report

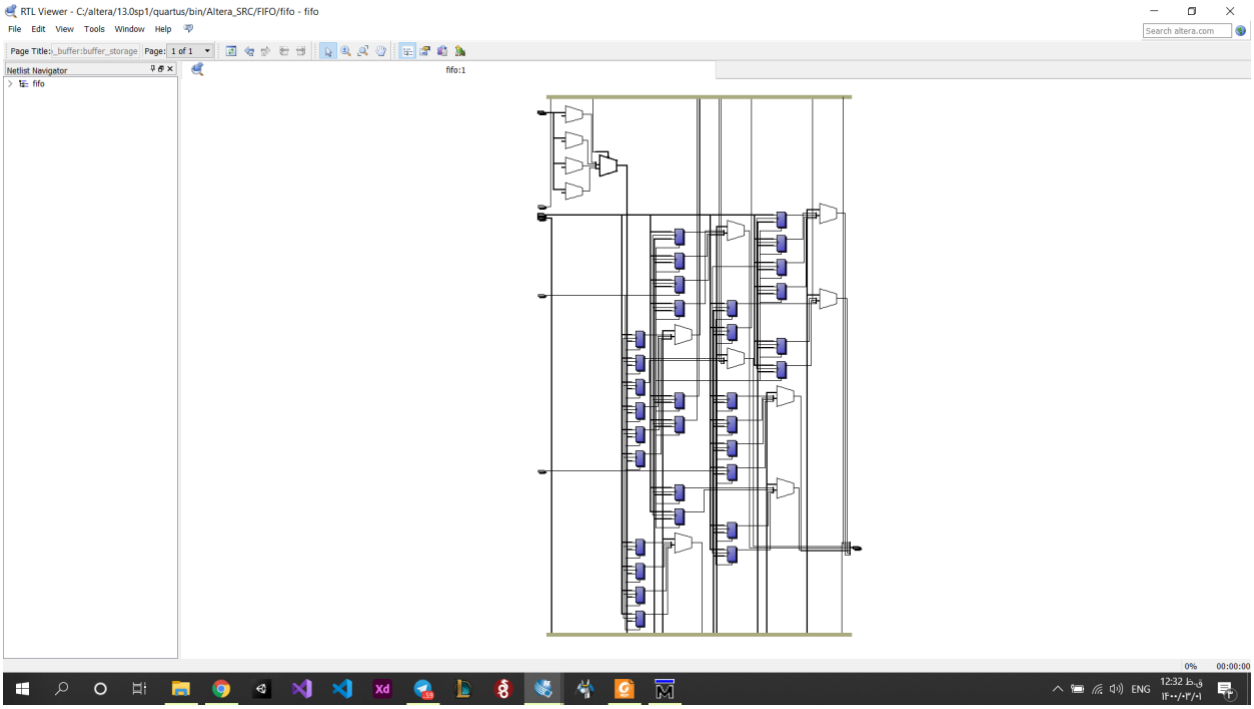


RTL

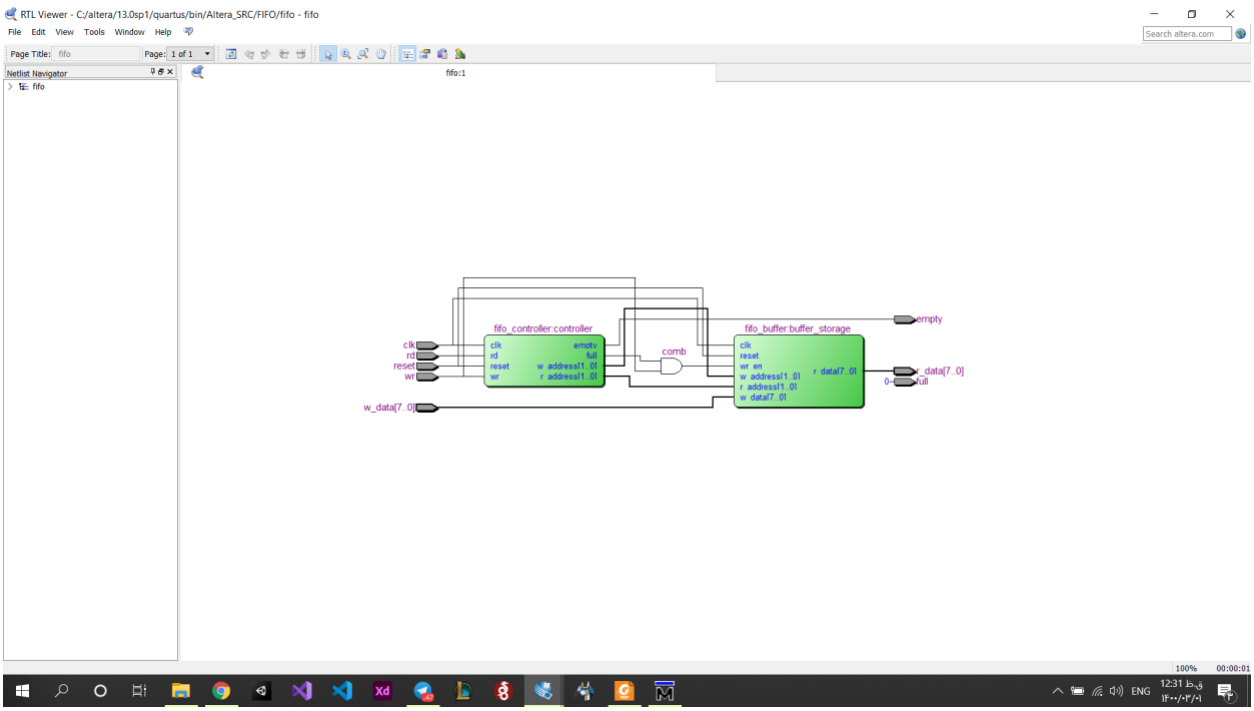
Fifo controller



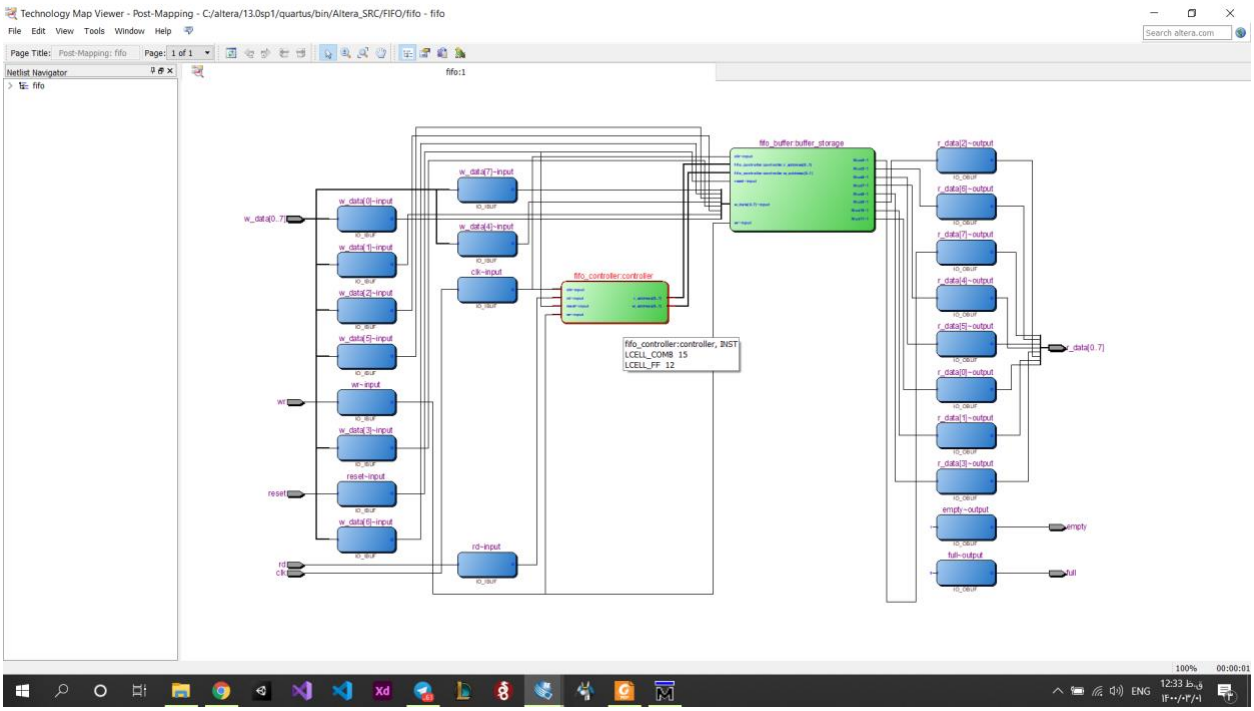
Fifo buffer



Fifo



Post-Mapping



Simulation

متاسفانه هر چه قدر تلاش کردیم هعی ارور میداد که error in loading design و در آخر هم متوجه نشدیم. بعدش از خود مدل سیم فایل ها رو اپلود کردیم و در مدل سیم کامپایل رو انجام دادیم. در مدل سیم در کامپایل ۲تا از فایلا ارور میداد در حالیکه در کوارتس این ارور ها رو نداشتیم.