**طراحی سیستم های دیجیتال**

**تکلیف سوم**

**استاد مروستی**

**علیرضا نعمتی**

**کسری رشیدفر**

Table of Contents

[Question1 2](#_Toc72537312)

[Code 2](#_Toc72537313)

[Dec\_counter 2](#_Toc72537314)

[N\_dec\_counter 3](#_Toc72537315)

[testbench 4](#_Toc72537316)

[Compilation report 6](#_Toc72537317)

[RTL 6](#_Toc72537318)

[Post-Mapping 7](#_Toc72537319)

[Simulation 8](#_Toc72537320)

[اسکرین شات 8](#_Toc72537321)

[توضیح عملکرد 9](#_Toc72537322)

[Question2 9](#_Toc72537323)

[Code 9](#_Toc72537324)

[Fifo controller 9](#_Toc72537325)

[Fifo buffer 11](#_Toc72537326)

[Fifo 13](#_Toc72537327)

[Fifo Test Bench 14](#_Toc72537328)

[Compilation report 16](#_Toc72537329)

[RTL 17](#_Toc72537330)

[Fifo controller 17](#_Toc72537331)

[Fifo buffer 17](#_Toc72537332)

[Fifo 18](#_Toc72537333)

[Post-Mapping 18](#_Toc72537334)

[Simulation 19](#_Toc72537335)

# 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;

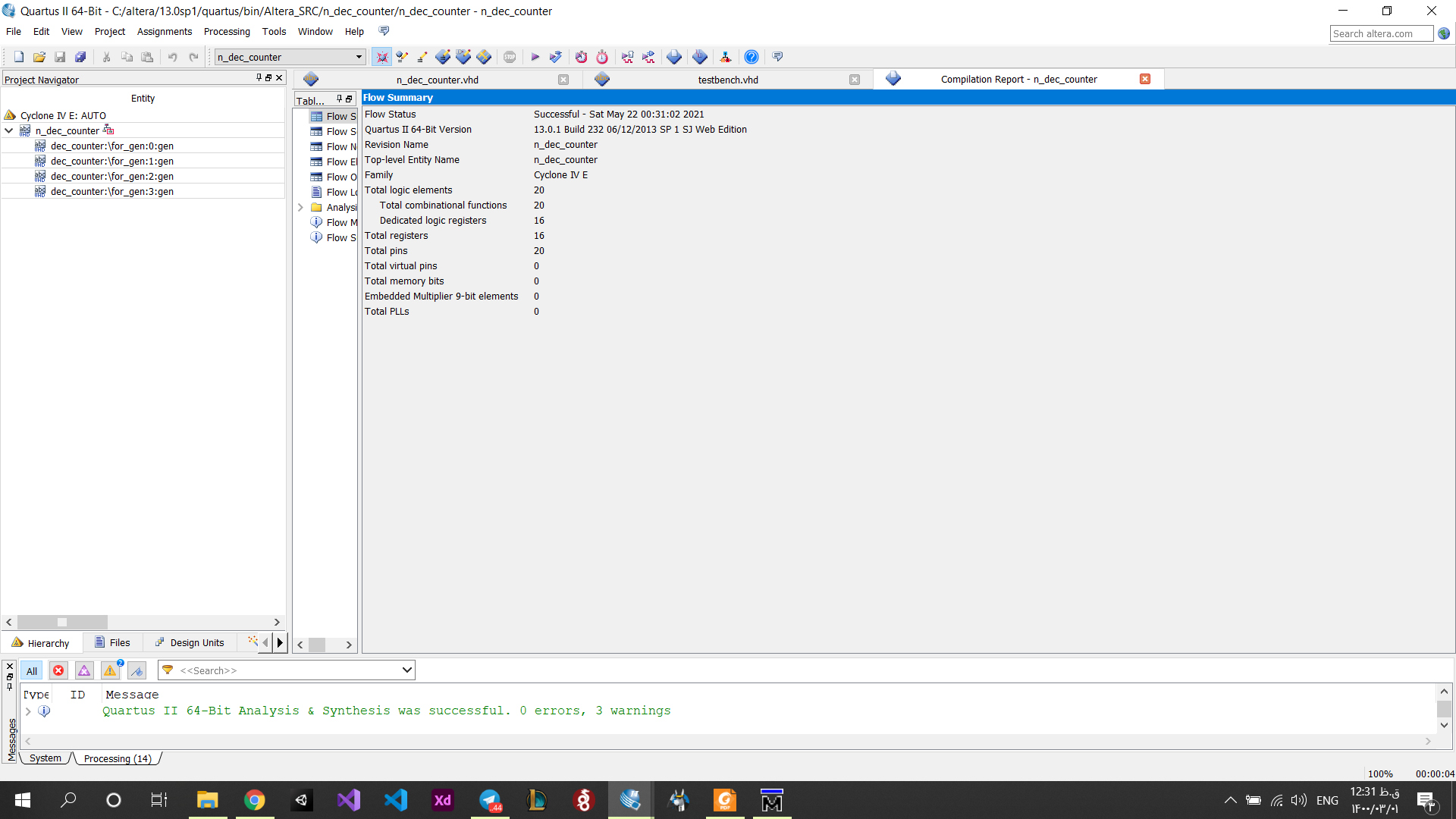
clk <= '0';

wait for 15 ns;

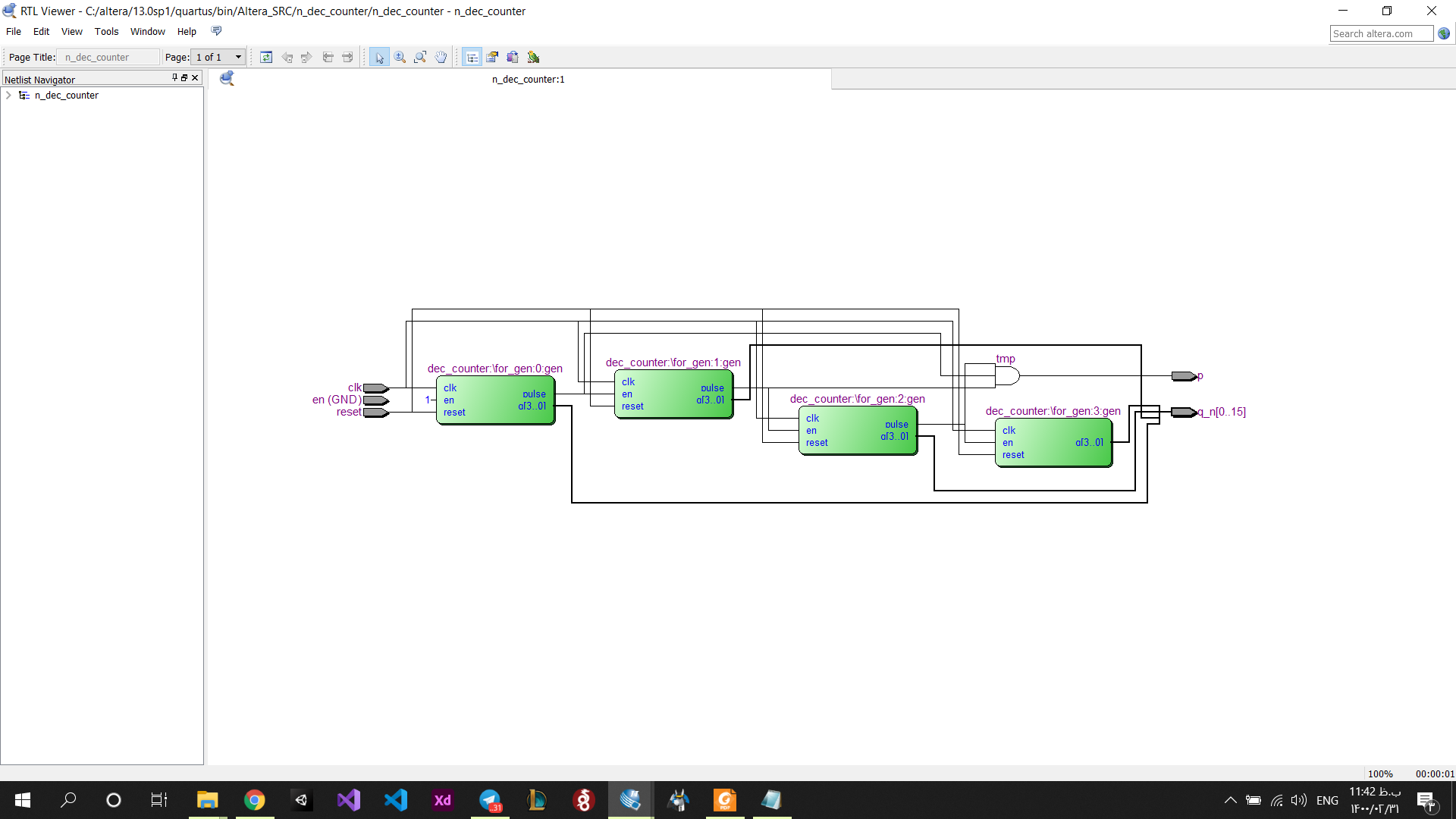
end process;

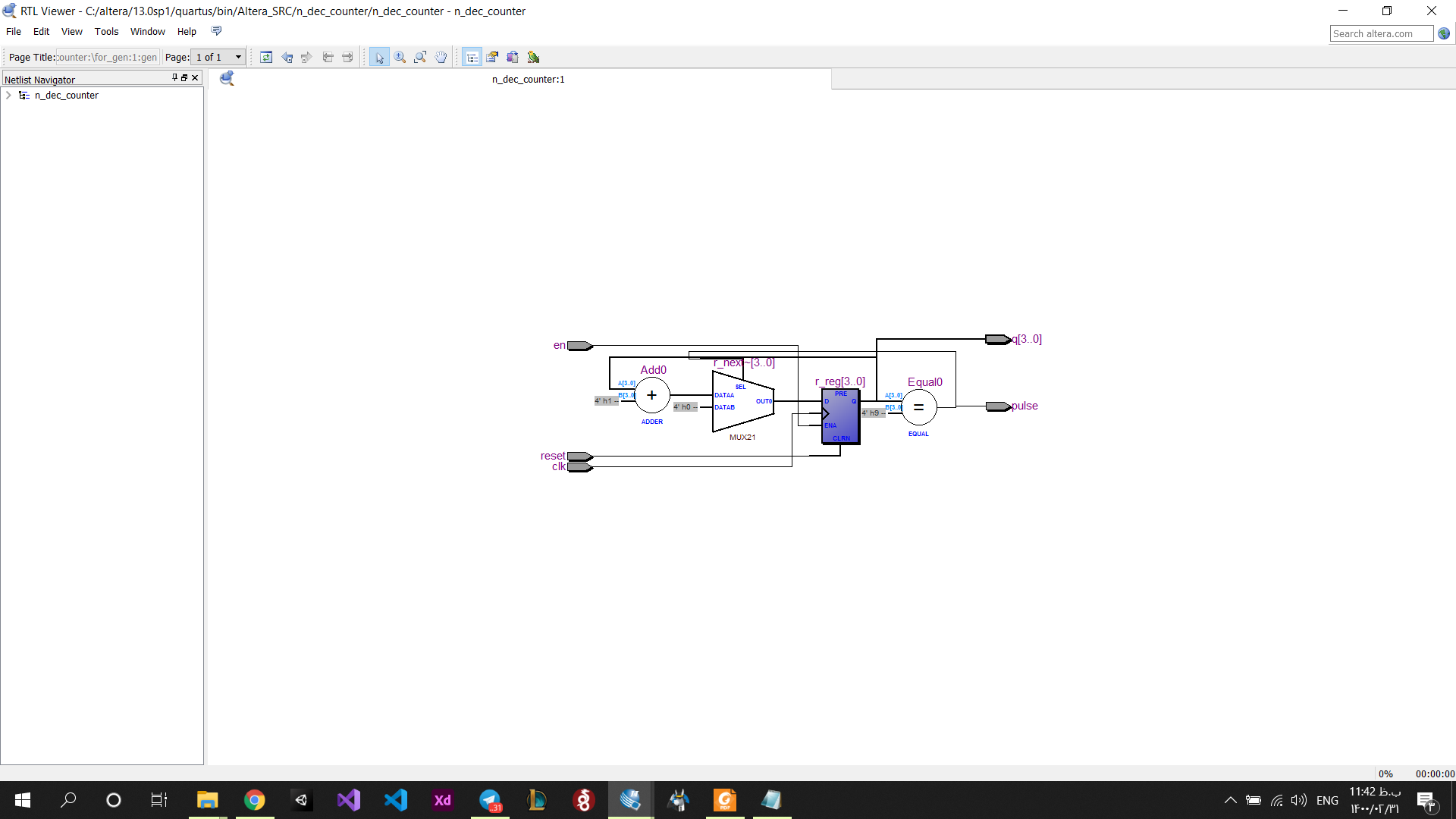
end arch;

## Compilation report

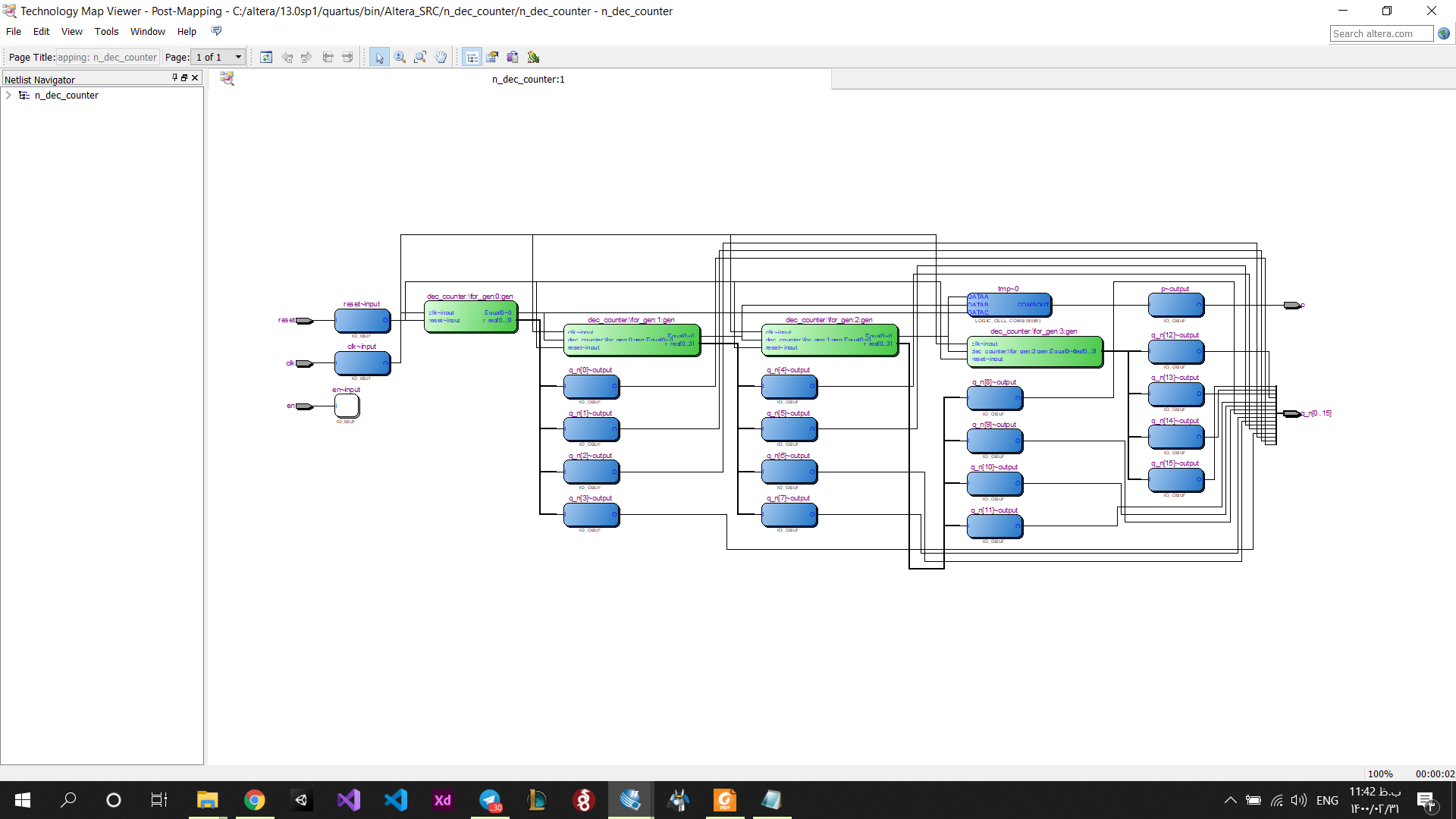
****

## RTL

****

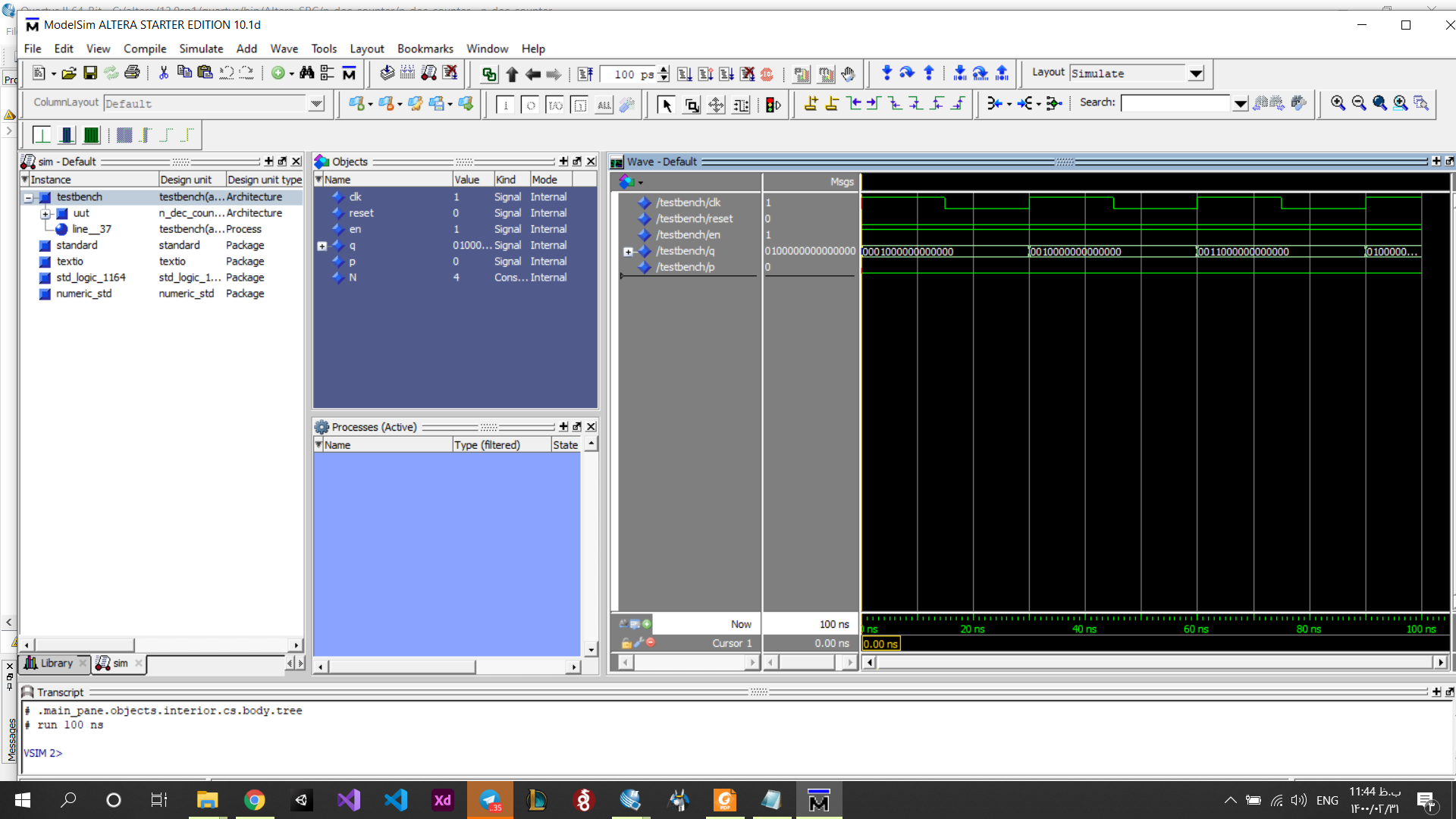
****

## 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;

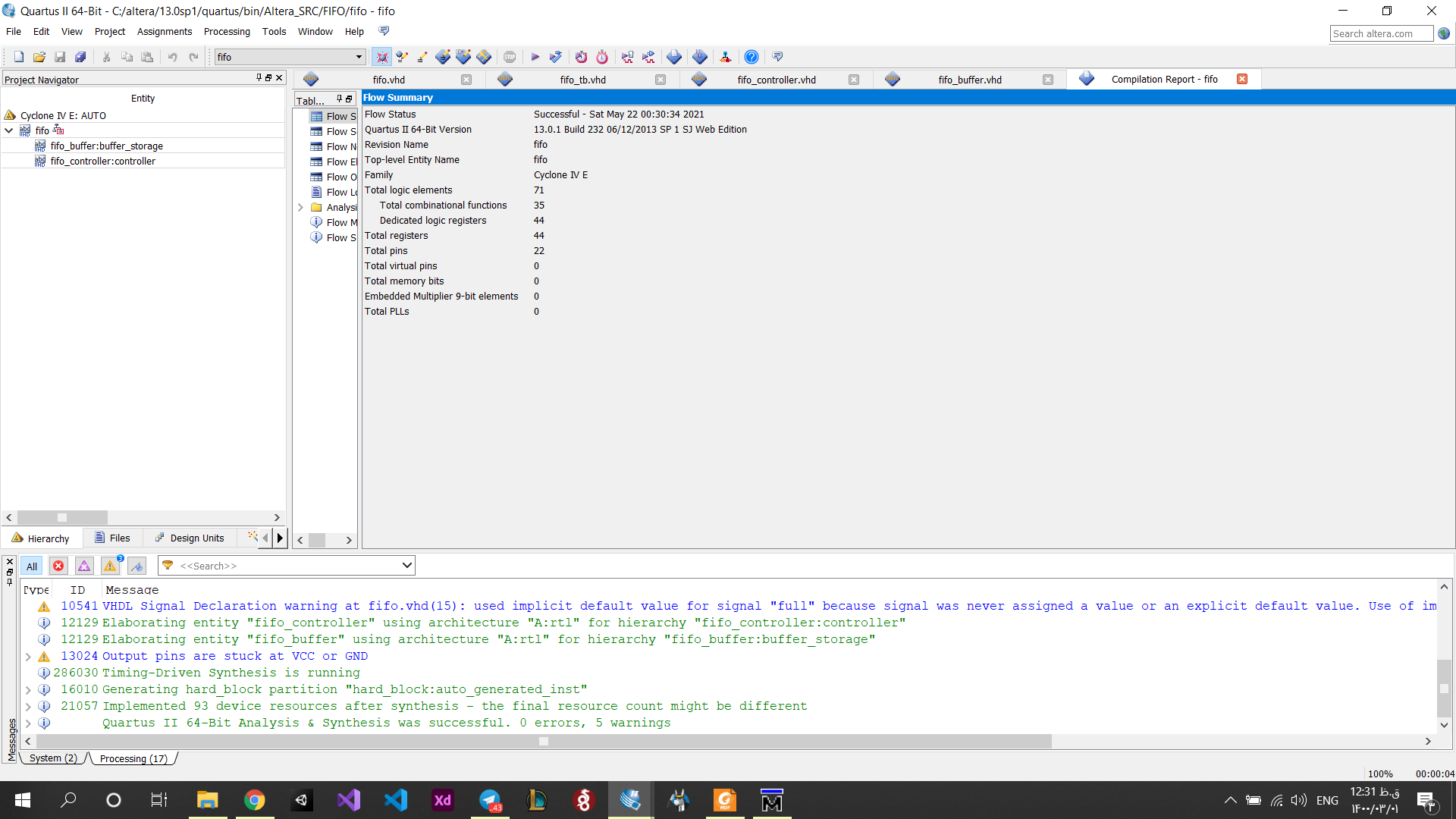
wr <= '0';

rd <= '1';

end process;

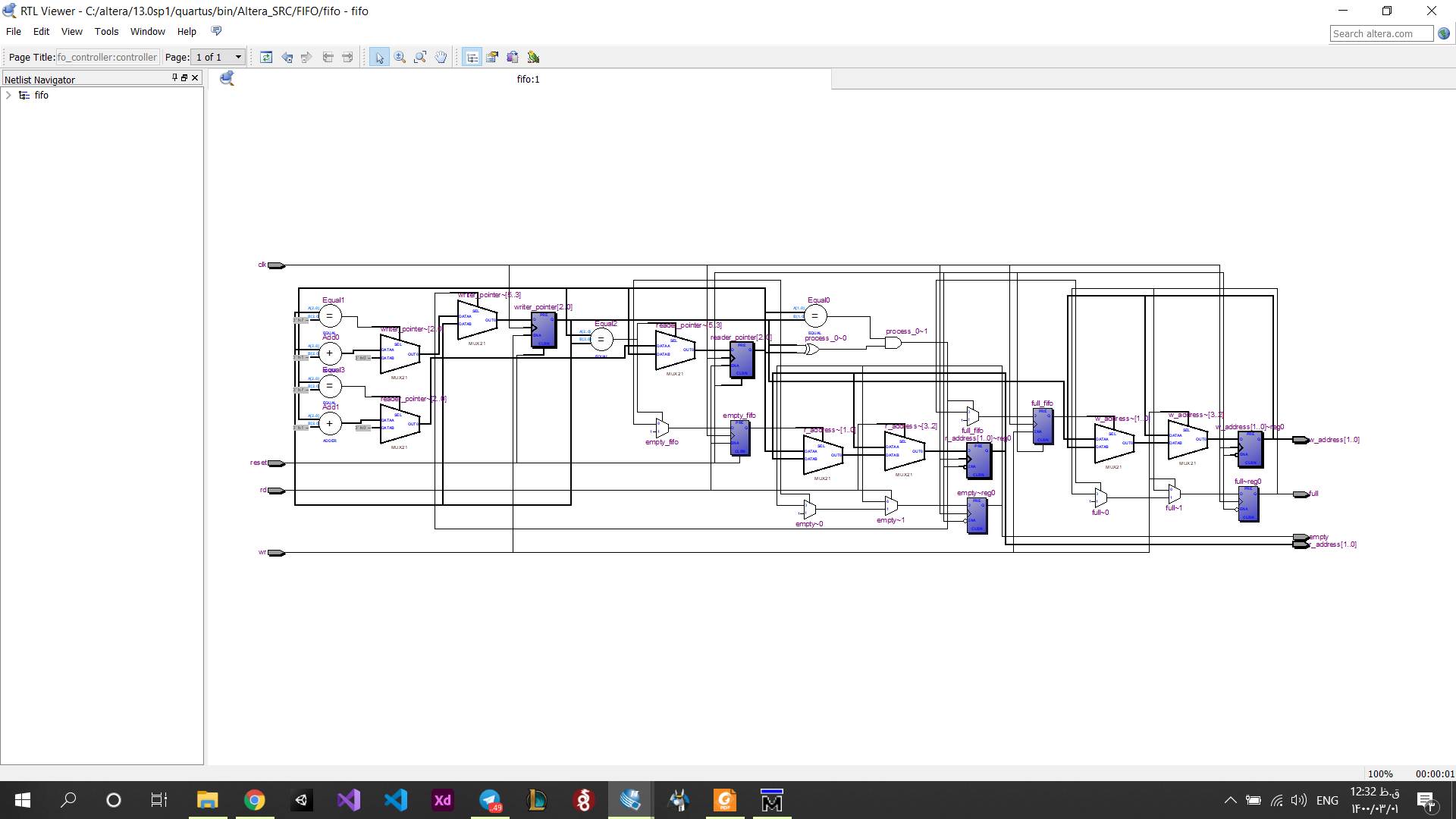
end architecture;

## Compilation report

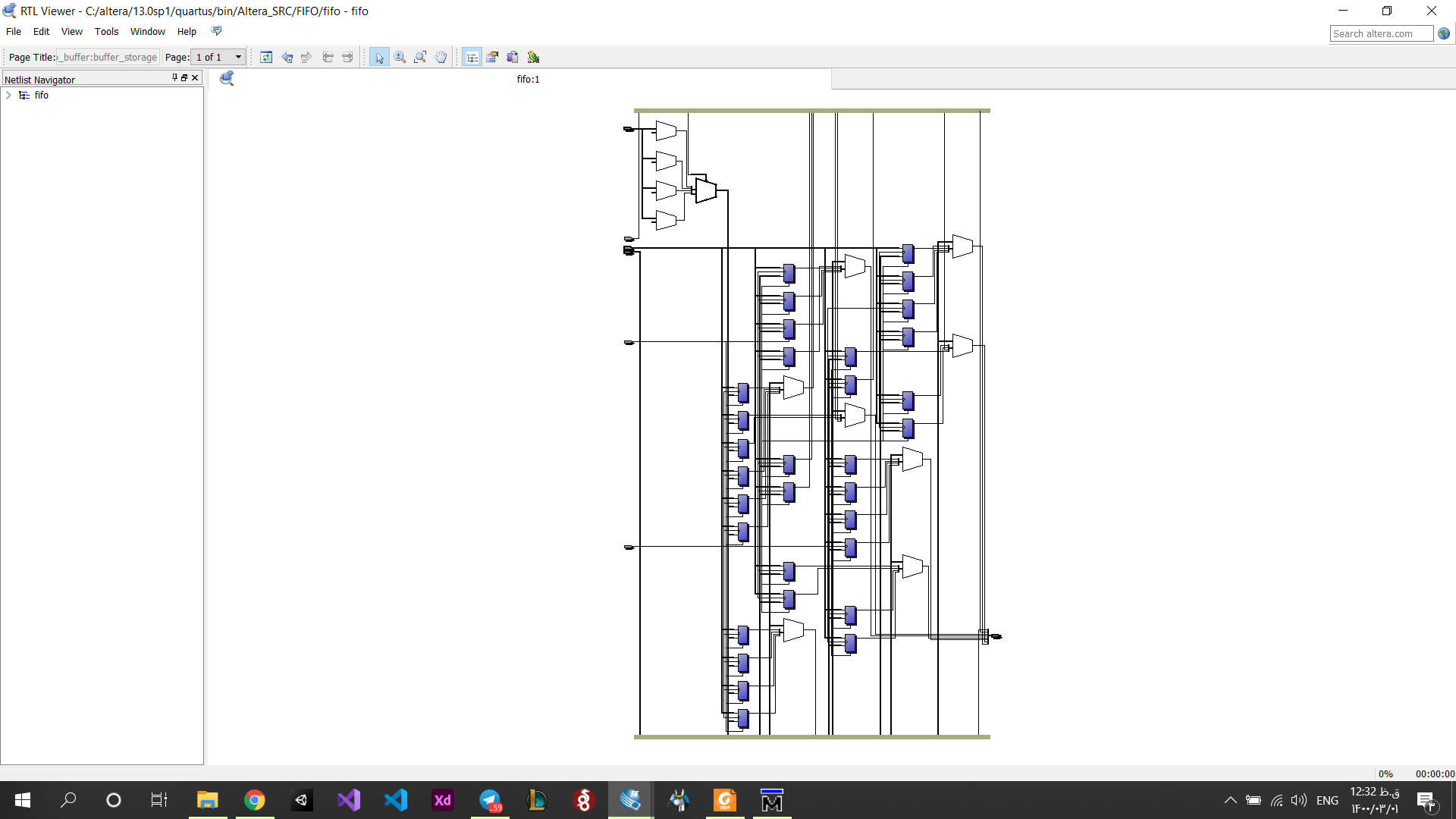
****

## RTL

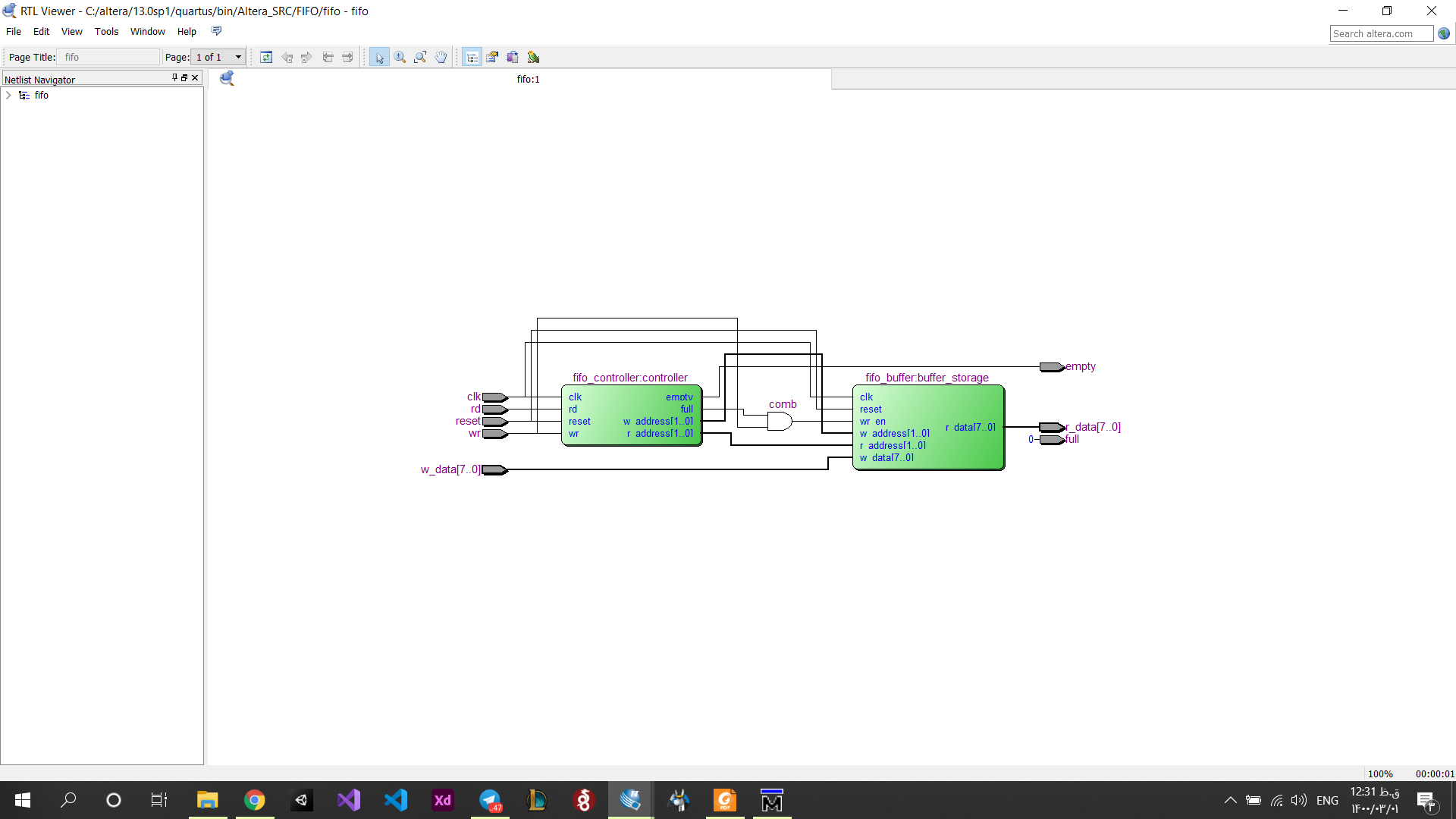
### Fifo controller



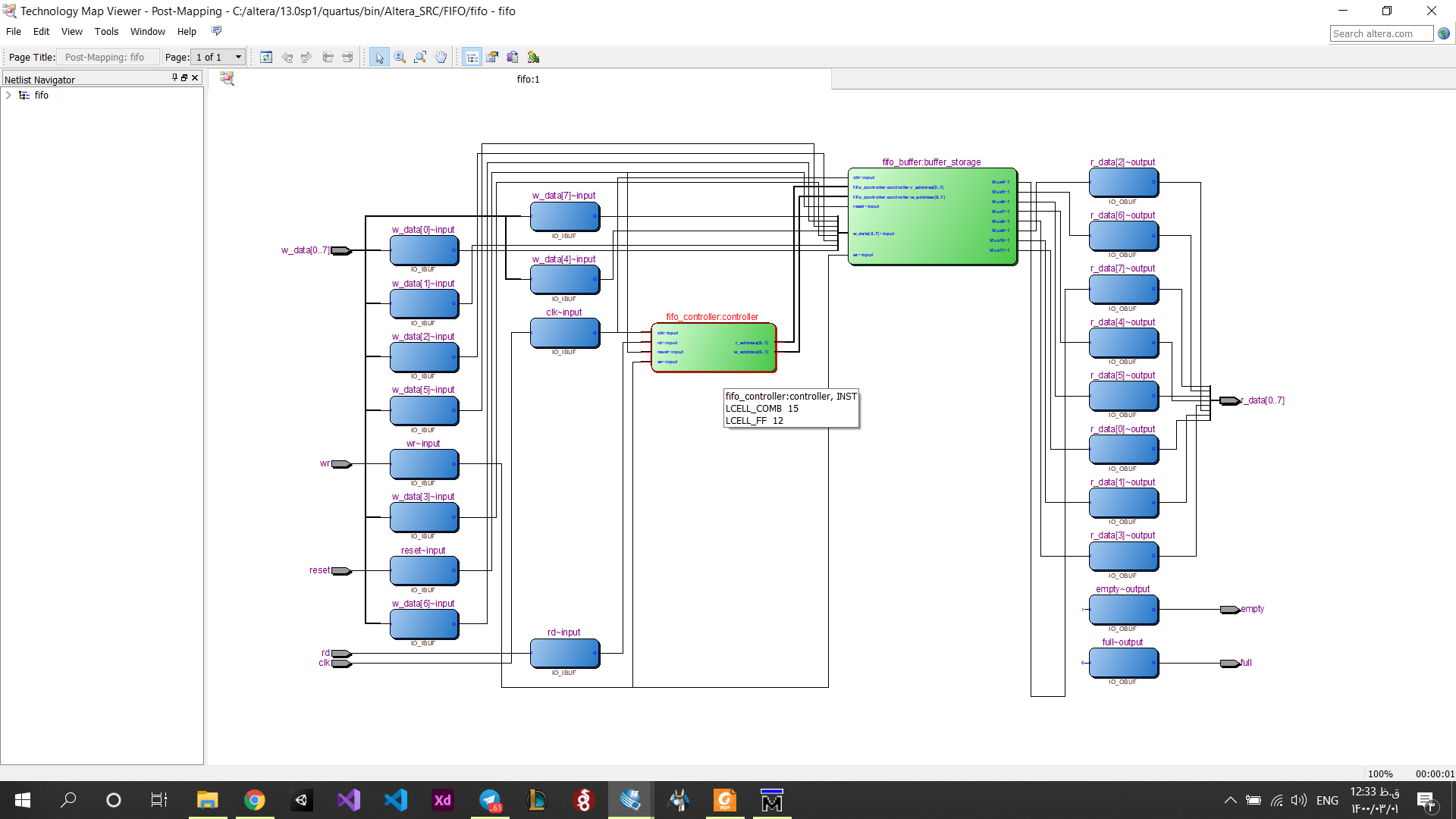
### Fifo buffer



### 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.**