**طراحی سیستم های دیجیتال**

**تکلیف اول**

**استاد مروستی**

**علیرضا نعمتی**

**کسری رشیدفر**

Table of Contents

[Question1 2](#_Toc68385857)

[Code 2](#_Toc68385858)

[Compilation report 3](#_Toc68385859)

[RTL 3](#_Toc68385860)

[Post-Mapping 4](#_Toc68385861)

[Question2 4](#_Toc68385862)

[Code 4](#_Toc68385863)

[Compilation report 5](#_Toc68385864)

[RTL 6](#_Toc68385865)

[Post-Mapping 7](#_Toc68385866)

[Question3 7](#_Toc68385867)

[Code 7](#_Toc68385868)

[Compilation report 8](#_Toc68385869)

[RTL 9](#_Toc68385870)

[Post-Mapping 9](#_Toc68385871)

[Question4 10](#_Toc68385872)

[A 10](#_Toc68385873)

[B 11](#_Toc68385874)

[Code 11](#_Toc68385875)

[Compilation report 12](#_Toc68385876)

[RTL 13](#_Toc68385877)

[Post-mapping 14](#_Toc68385878)

# Question1

## Code

library ieee;

use ieee.std\_logic\_1164.all;

entity shifter4 is

port(

s: in std\_logic\_vector(1 downto 0);

w: in std\_logic\_vector(3 downto 0);

y: out std\_logic\_vector(3 downto 0)

);

end shifter4;

architecture sel\_shift of shifter4 is

begin

with s select

y <= (w(3) & w(2) & w(1) & w(0)) when "00",

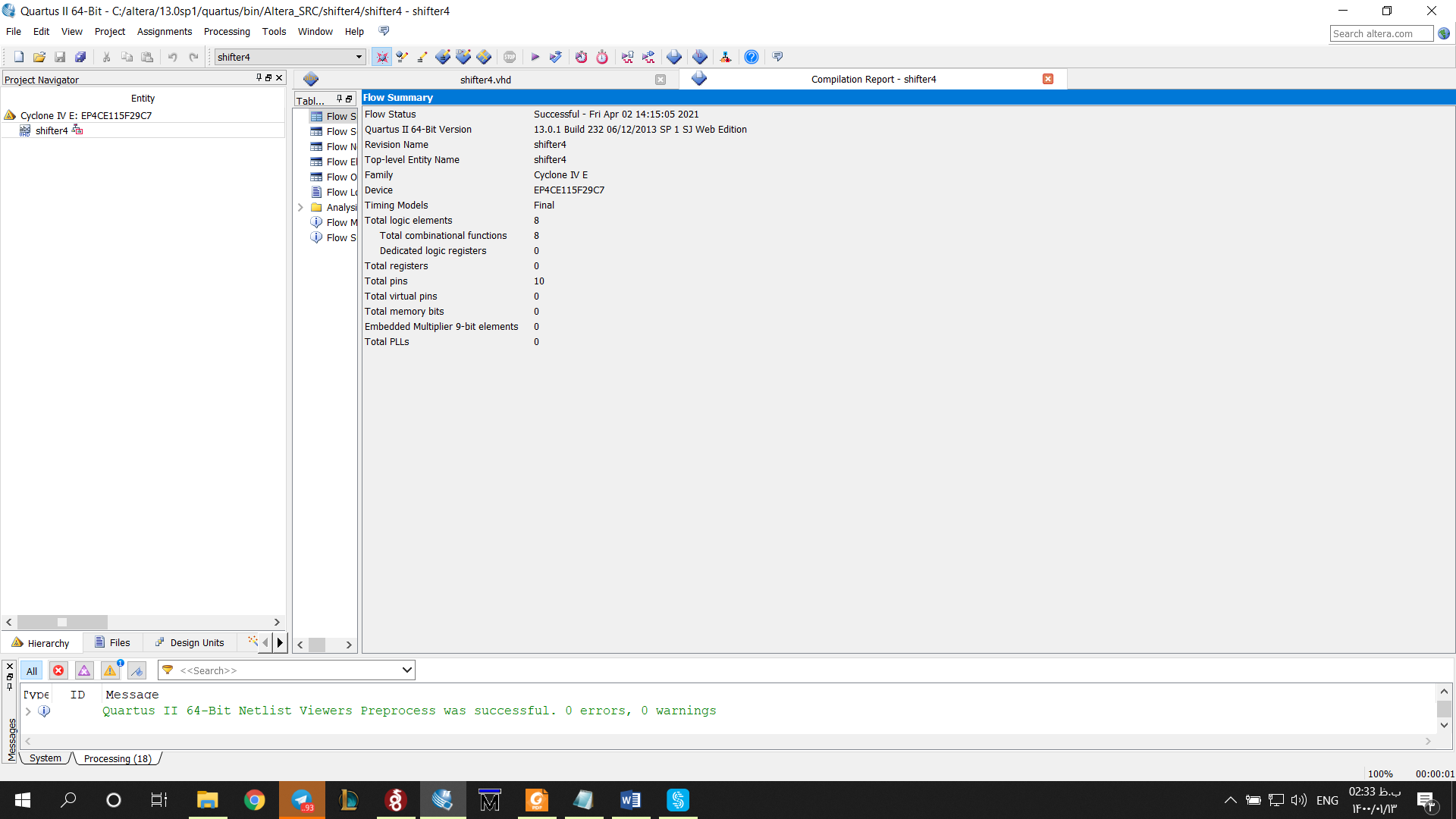
(w(0) & w(3) & w(2) & w(1)) when "01",

(w(1) & w(0) & w(3) & w(2)) when "10",

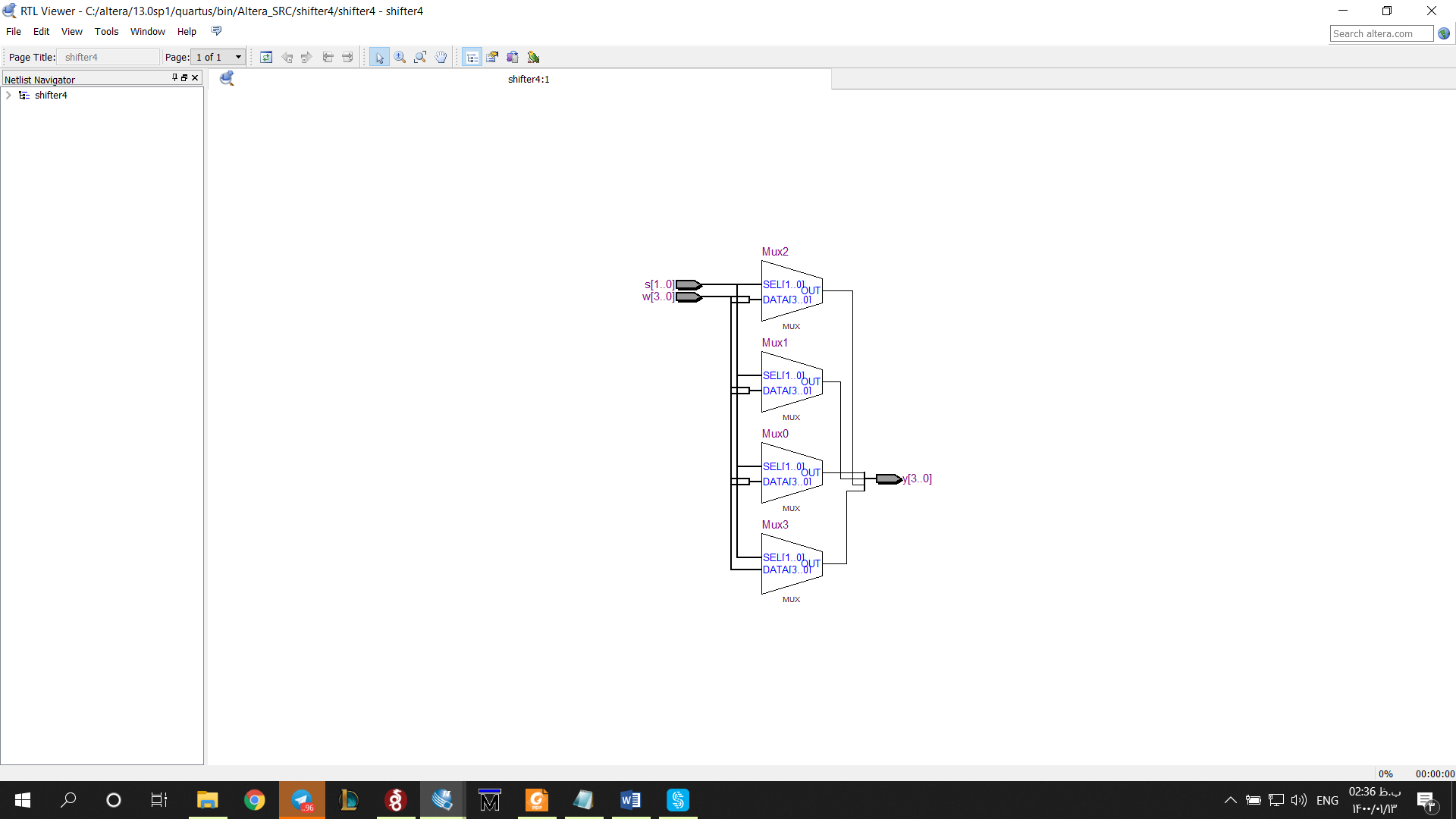
(w(2) & w(1) & w(0) & w(3)) when others;

end sel\_shift;

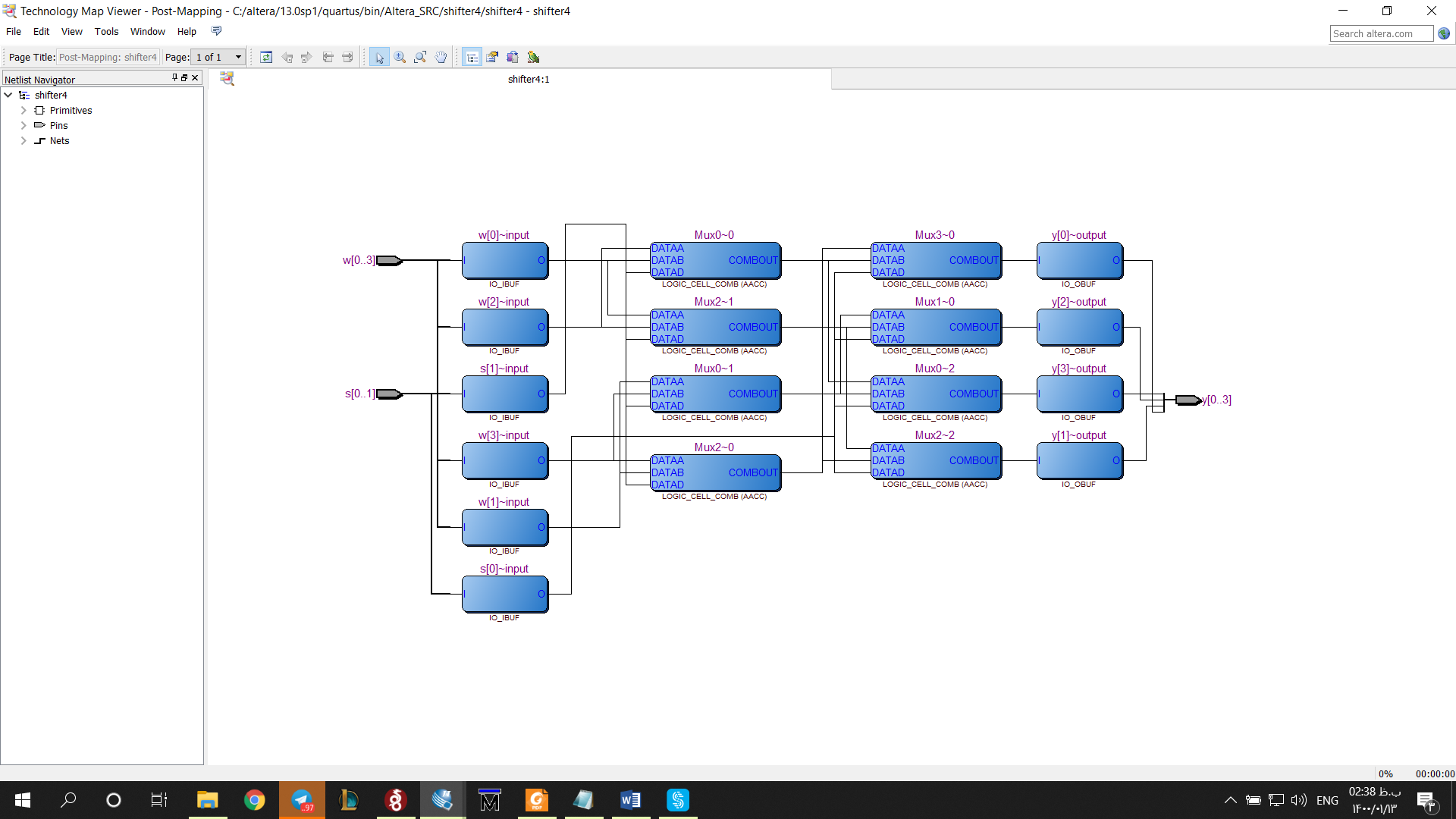
## Compilation report



## RTL



## Post-Mapping



# Question2

## Code

library ieee;

use ieee.std\_logic\_1164.all;

entity BCD2seg is

port(

w: in std\_logic\_vector(3 downto 0);

seg: out std\_logic\_vector(6 downto 0));

end BCD2seg;

architecture decoder\_arch of BCD2seg is

begin

process(w)

begin

case w is

when "0000" =>

seg <= "1111110";

when "0001" =>

seg <= "0110000";

when "0010" =>

seg <= "1101101";

when "0011" =>

seg <= "1111001";

when "0100" =>

seg <= "0110011";

when "0101" =>

seg <= "1011011";

when "0110" =>

seg <= "1011111";

when "0111" =>

seg <= "1110000";

when "1000" =>

seg <= "1111111";

when "1001" =>

seg <= "1111011";

when "1010" =>

seg <= "1111101";

when "1011" =>

seg <= "0011111";

when "1100" =>

seg <= "1001110";

when "1101" =>

seg <= "0111101";

when "1110" =>

seg <= "1001111";

when others =>

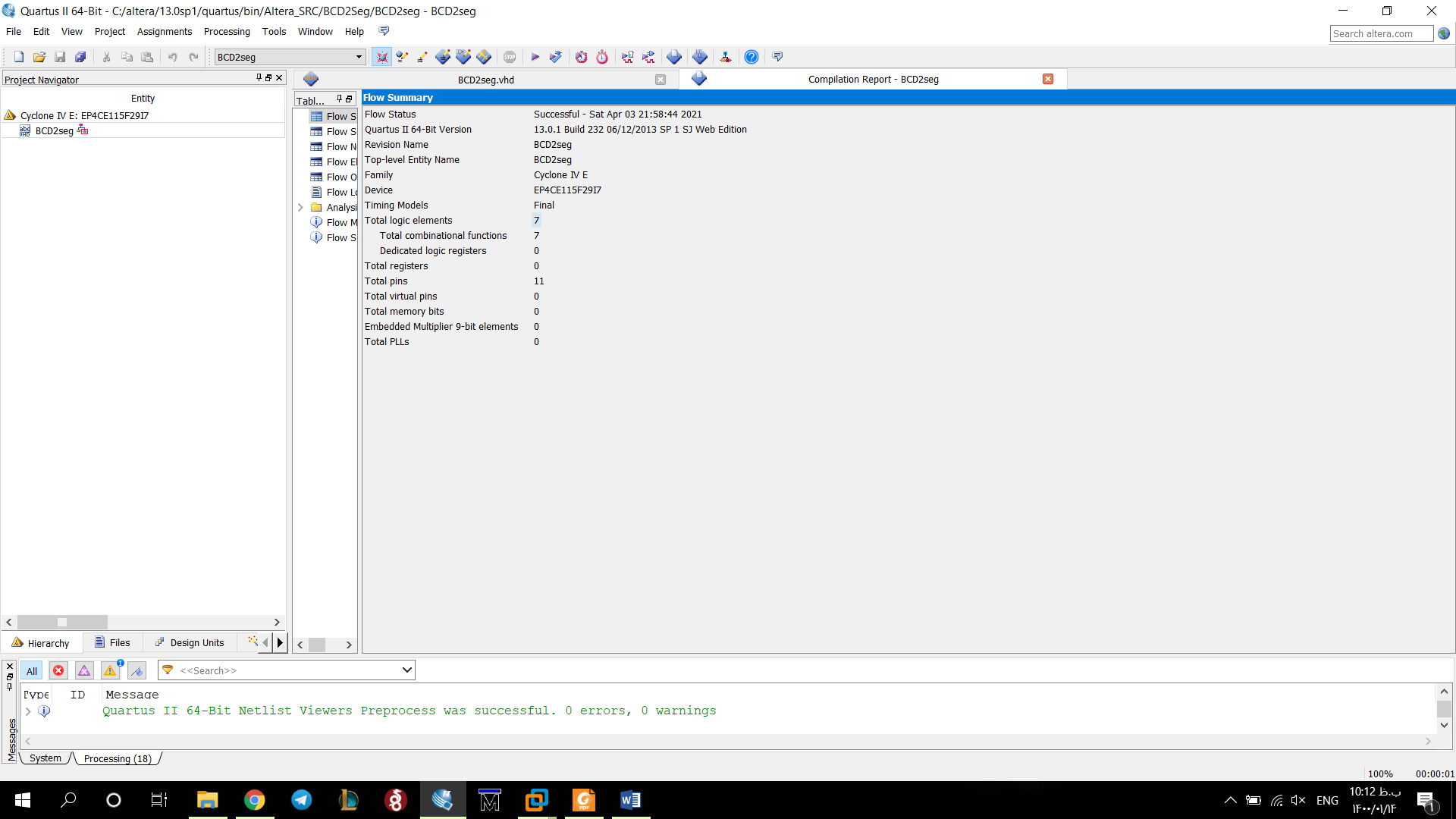
seg <= "1000111";

end case;

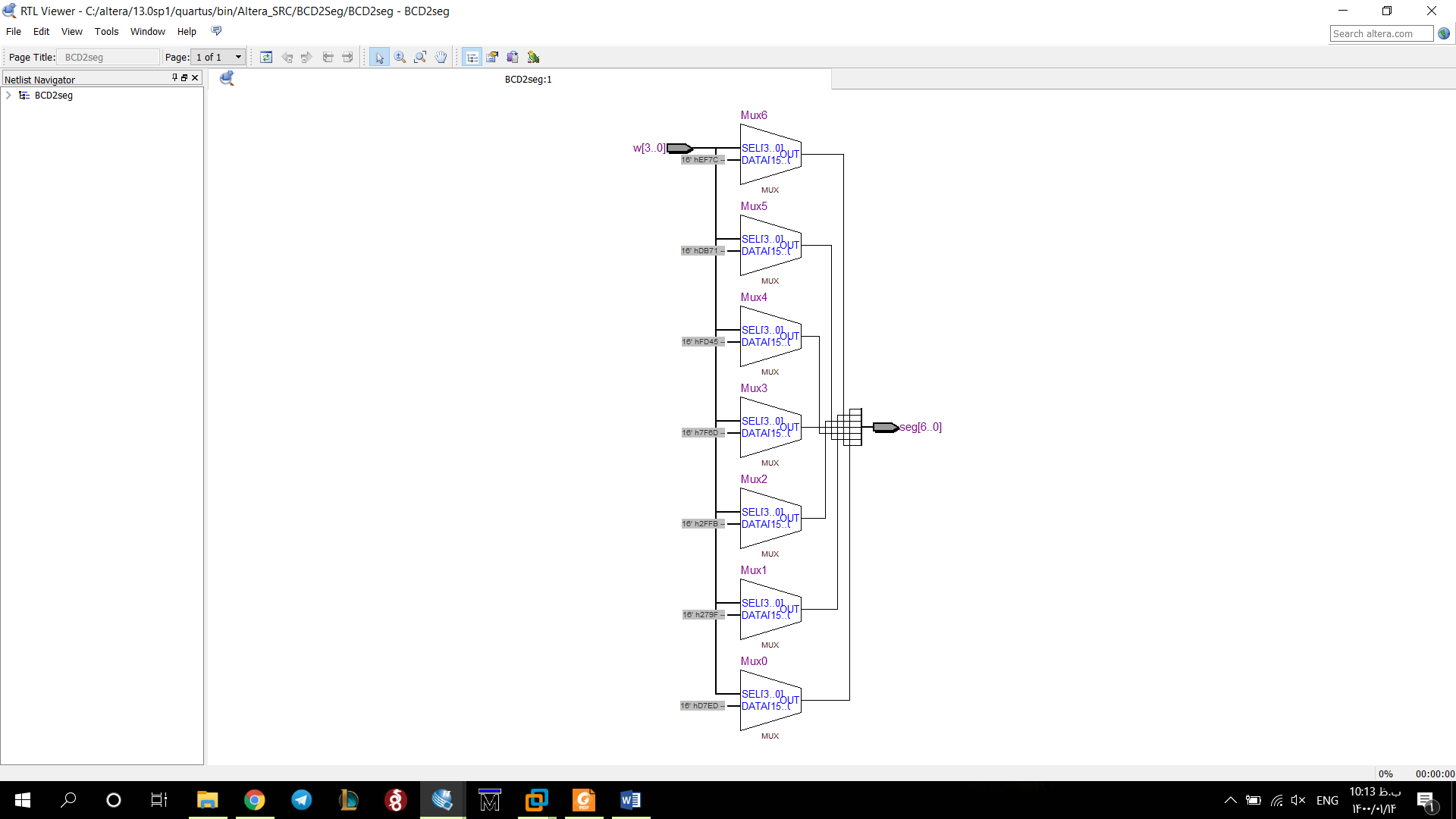
end process;

end decoder\_arch;

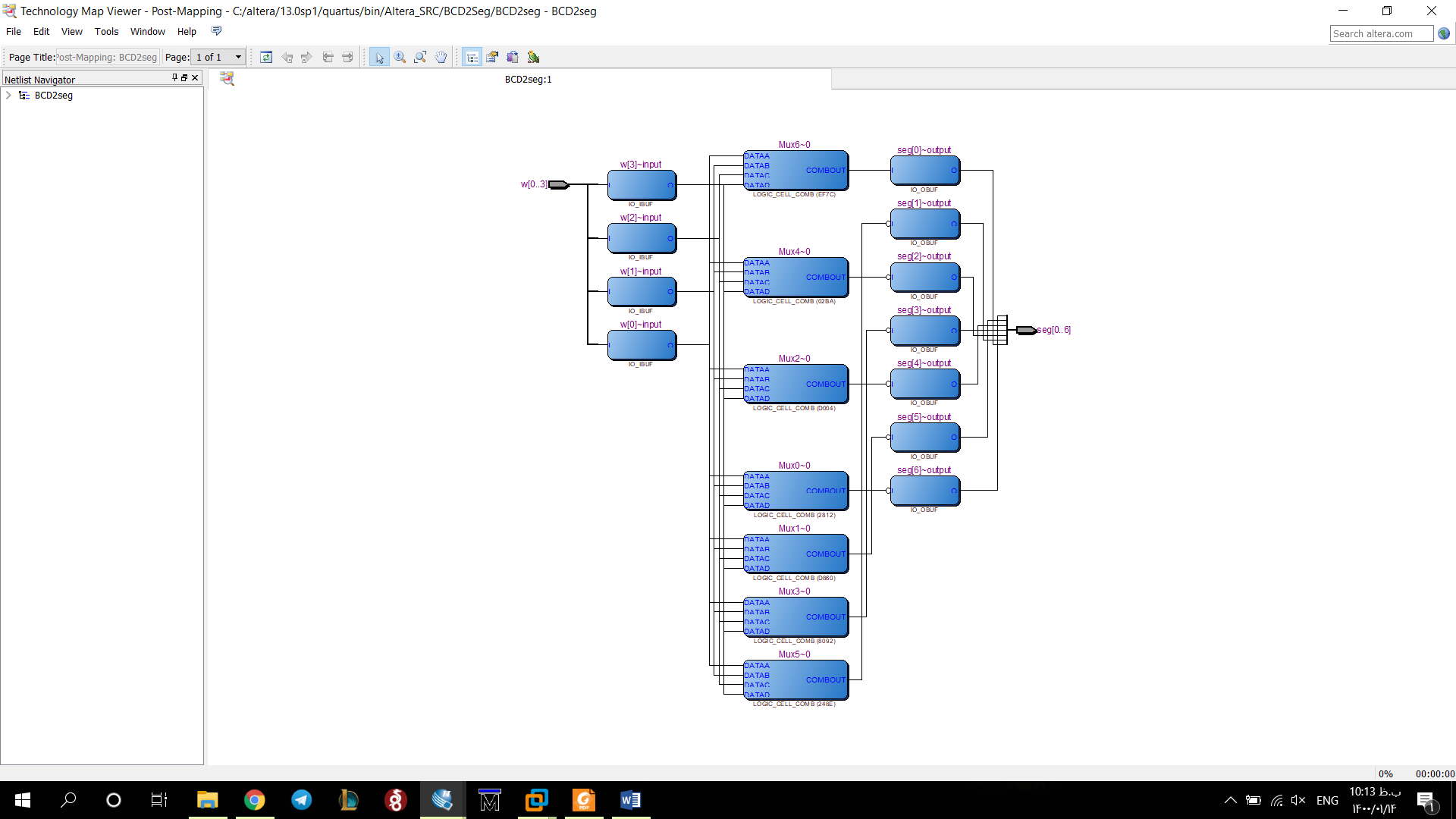
## Compilation report



## RTL



## Post-Mapping



# Question3

## Code

library ieee;

use ieee.std\_logic\_1164.all;

entity carry\_look\_ahead\_adder is

port(

input1,input2,carryIn : in bit;

sum,gen,propagate : out bit);

end entity;

architecture struct of carry\_look\_ahead\_adder is

begin

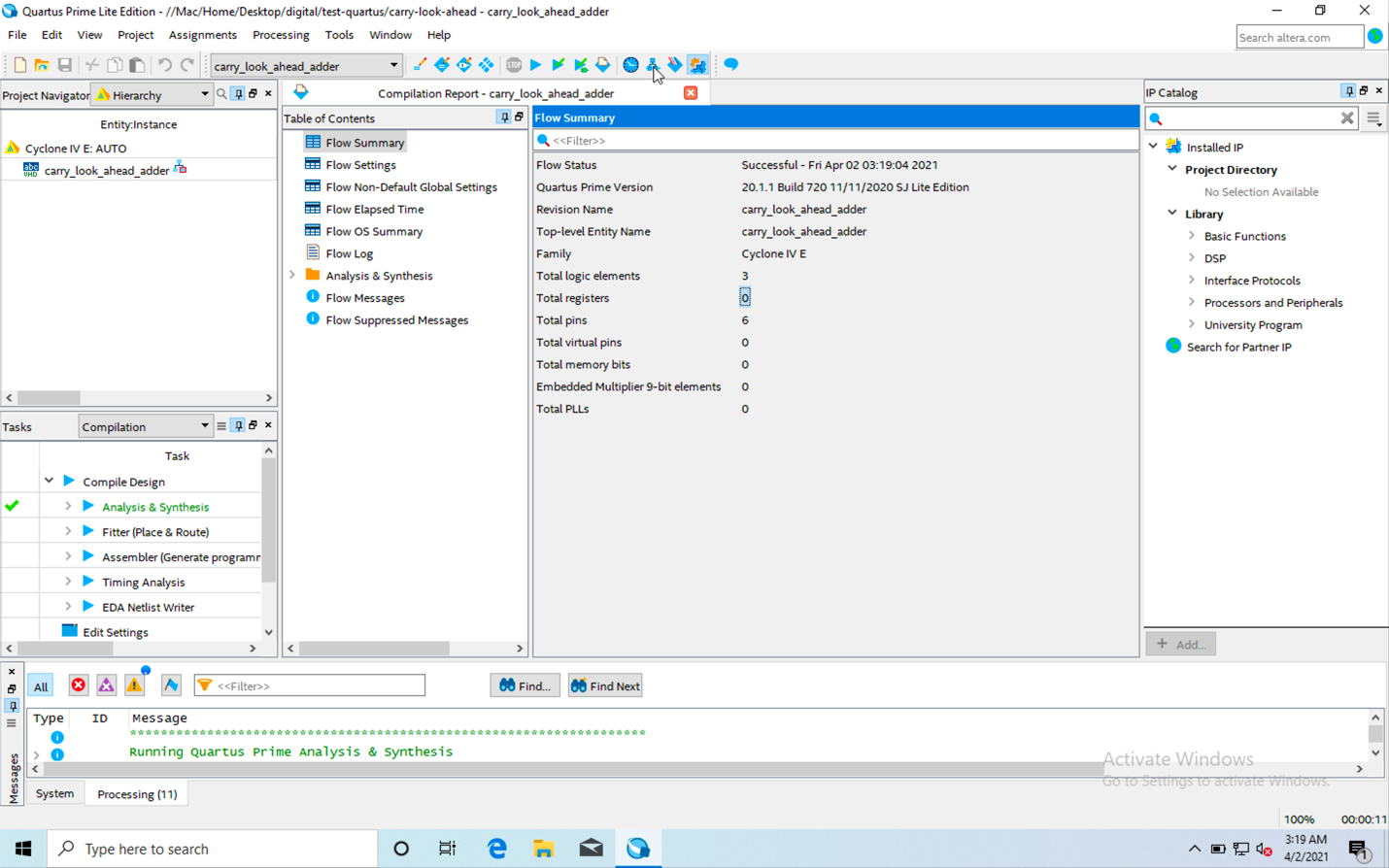
sum <= (input1 xor input2) xor carryIn;

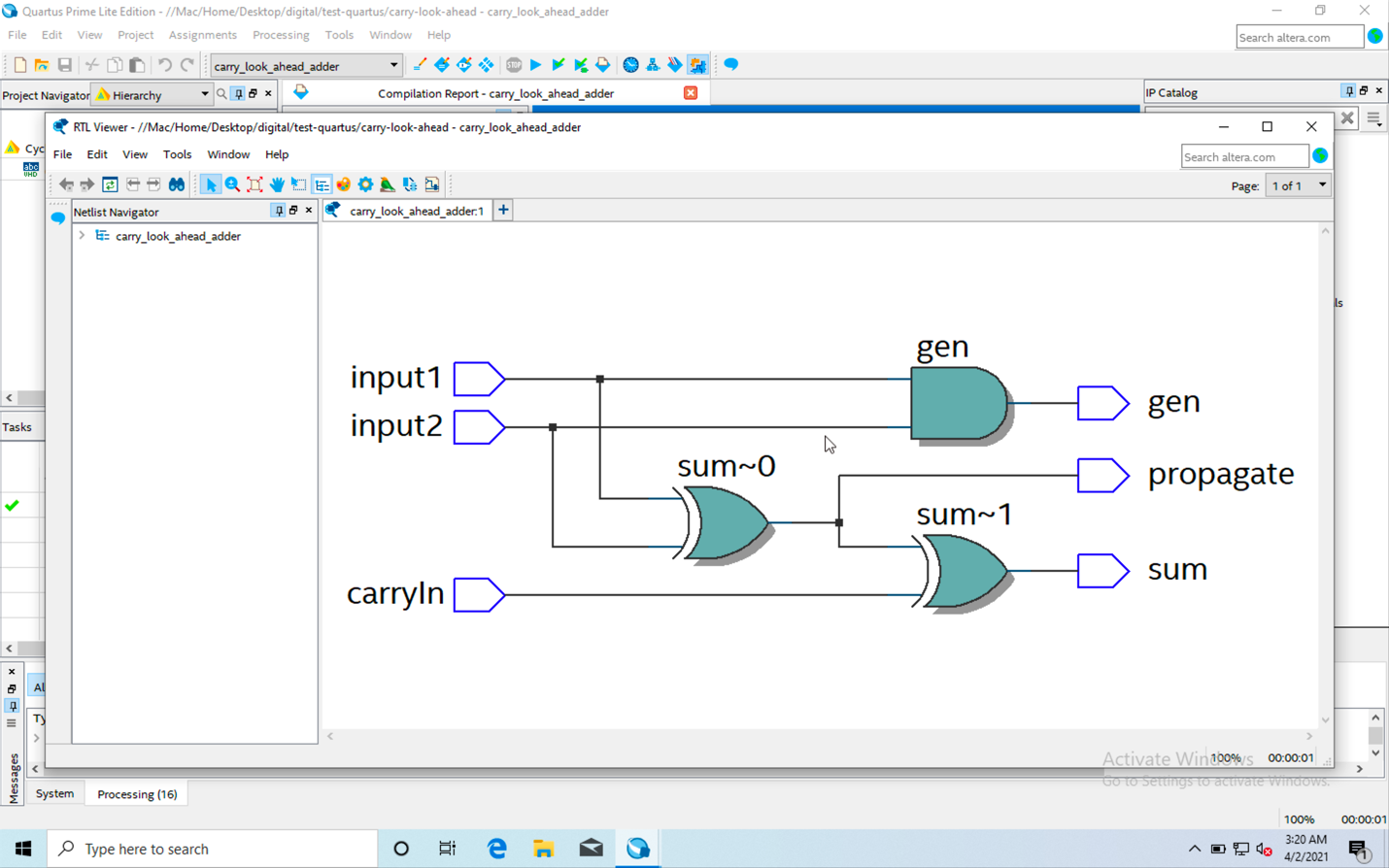
propagate <= input1 xor input2;

gen <= input1 and input2;

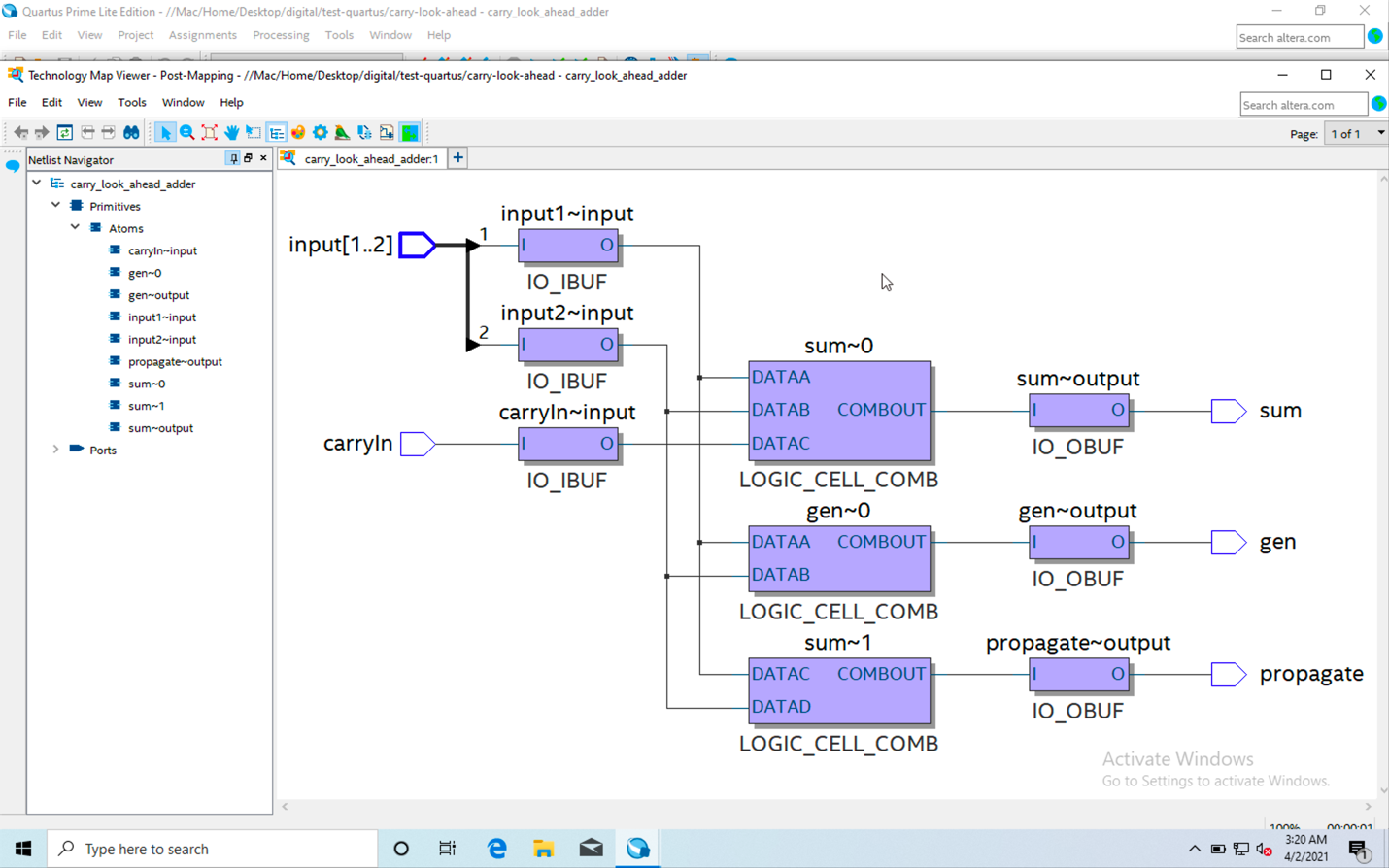
end architecture;

## Compilation report

****

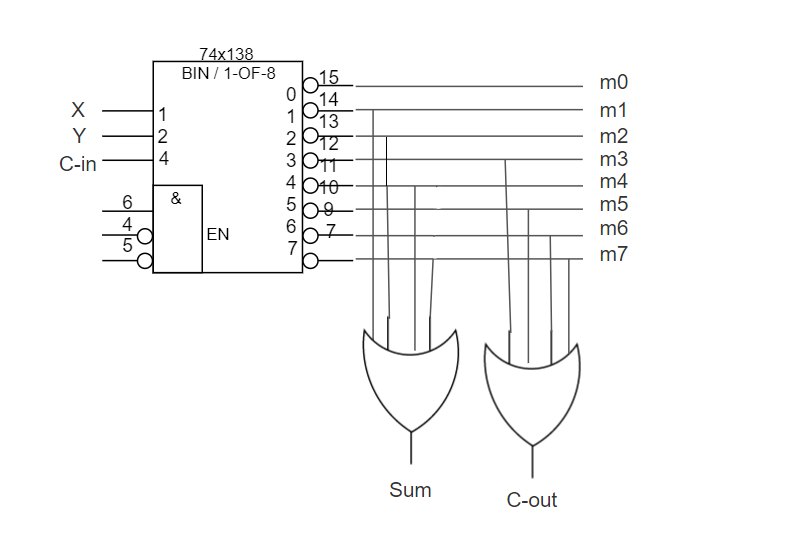
RTL****

## Post-Mapping

****

# Question4

## A

****

## B

### Code

library ieee;

use ieee.std\_logic\_1164.all;

entity full\_adder\_by\_decoder\_3x8 is

port(

inputDecoder : in STD\_LOGIC\_VECTOR(2 downto 0);

sum : out std\_logic;

carryOut : out std\_logic);

end entity;

architecture struct of full\_adder\_by\_decoder\_3x8 is

signal outPutDecoder : STD\_LOGIC\_VECTOR(7 downto 0);

begin

-- to see where m1,m2,m3,... come from, pleas pay attention to answer of section a

outPutDecoder <=("10000000") when (inputDecoder="000") else

("01000000") when (inputDecoder="001") else

("00100000") when (inputDecoder="010") else

("00010000") when (inputDecoder="011") else

("00001000") when (inputDecoder="100") else

("00000100") when (inputDecoder="101") else

("00000010") when (inputDecoder="110") else

("00000001");

process(outPutDecoder) is

variable m0,m1,m2,m3,m4,m5,m6,m7 : STD\_LOGIC;

begin

m0 :='0';

m1 :='0';

m2 :='0';

m3 :='0';

m4 :='0';

m5 :='0';

m6 :='0';

m7 :='0';

case outPutDecoder is

when "10000000" =>

m0 := '1';

when "01000000" =>

m1 := '1';

when "00100000" =>

m2 := '1';

when "00010000" =>

m3 := '1';

when "00001000" =>

m4 := '1';

when "00000100" =>

m5 := '1';

when "00000010" =>

m6 := '1';

when others =>

m7 := '1';

end case;

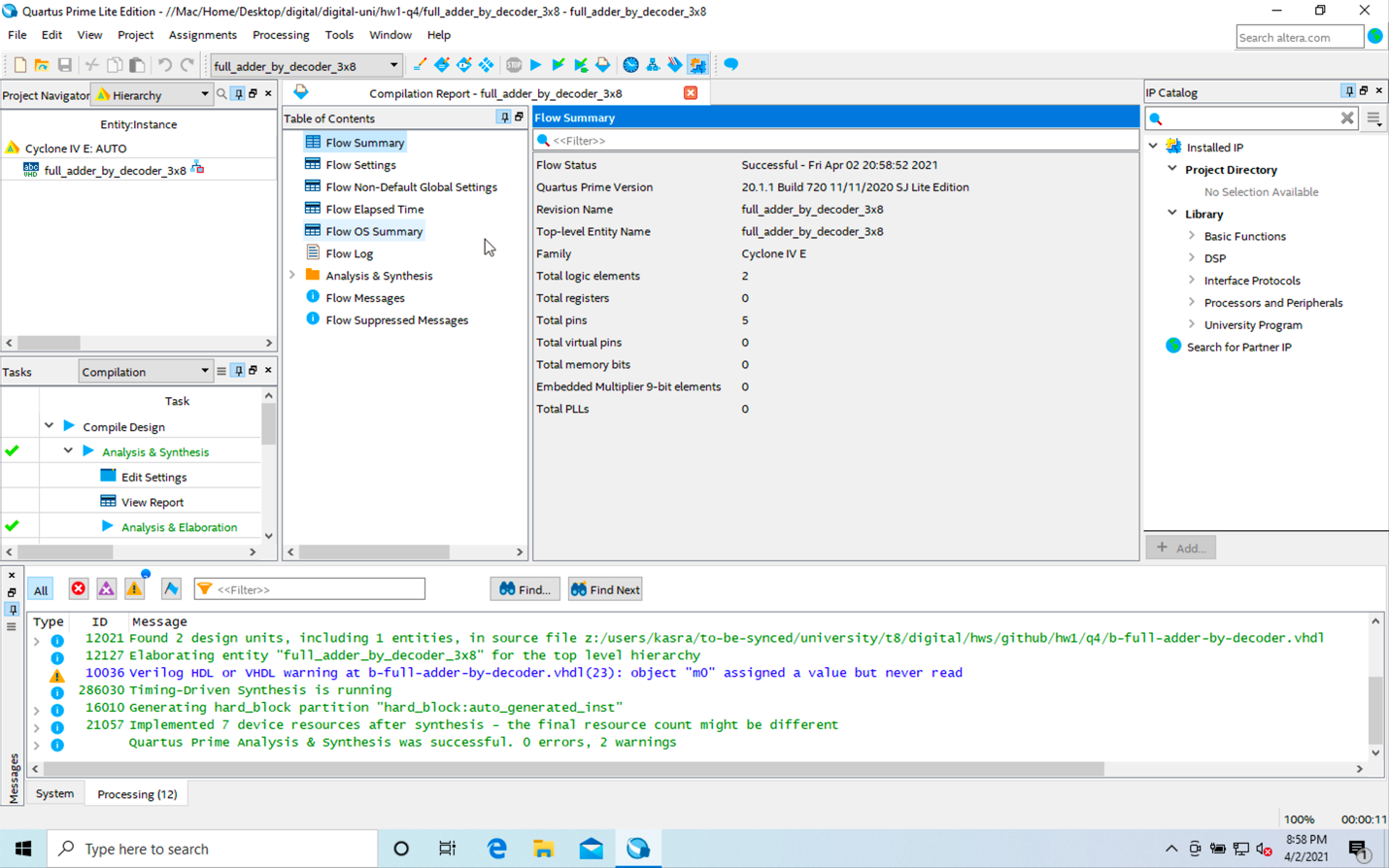
sum <= m1 OR m2 OR m4 OR m7;

carryOut <= m3 OR m5 OR m6 Or m7;

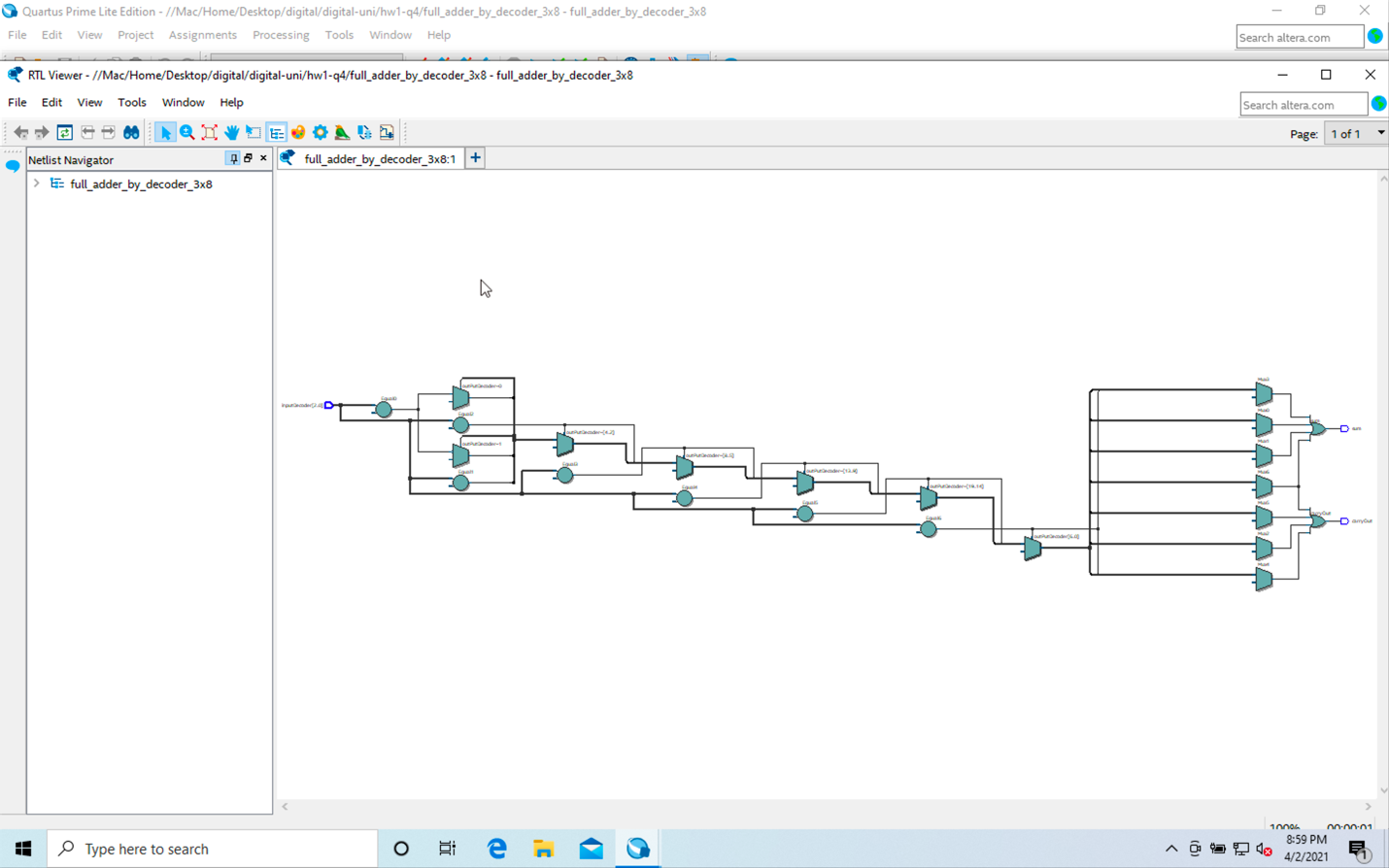
end process;

end architecture;

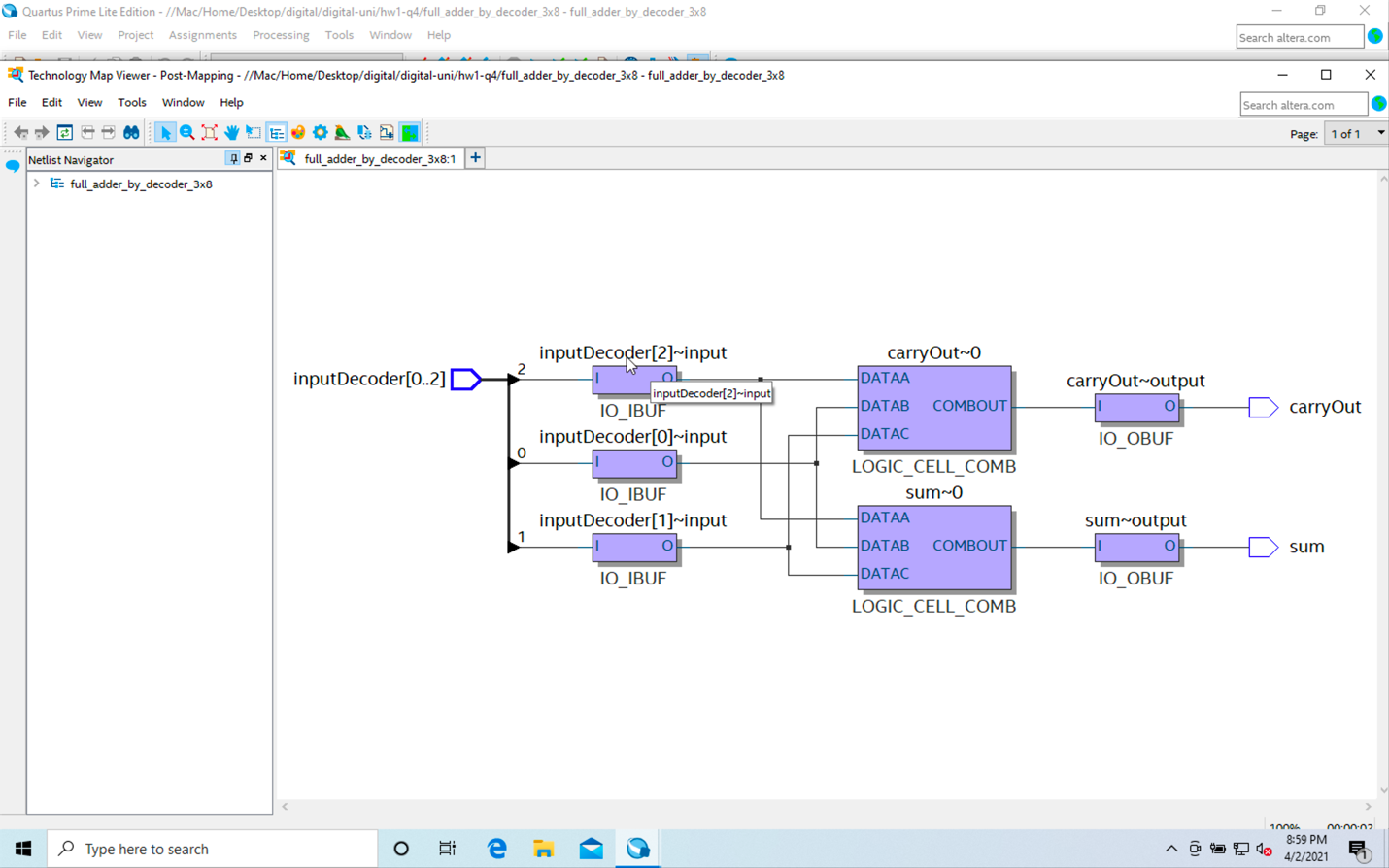
### Compilation report

****

### RTL

****

### Post-mapping

****