به نام خدا

تکلیف شماره اول

درس:

طراحی سیستم های دیجیتال

استاد درس:

دکتر مروستی

اعضا:

محمدعلی مجتهدسلیمانی-۹۹۲۰۲۳۰۳۹

حامد خسروی-۹۹۲۰۲۳۰۱۰

Q1. Explain the basic blocks and interconnections of FPGA, CPLD, and PAL. Then, compare these chips in terms of area, performance, power, applications, and security.

- Field-Programmable Gate Arrays (FPGAs), Complex Programmable Logic Devices (CPLDs), and Programmable Array Logic (PAL) devices are all types of programmable logic devices, but they differ in their architecture, complexity, and capabilities.

1. FPGAs (Field-Programmable Gate Arrays):

- Basic Blocks:
- Logic Blocks (Configurable Logic Blocks or CLBs): These are the basic building blocks containing lookup tables (LUTs) and flip-flops for implementing logic functions.
- Routing Resources: FPGAs have a vast interconnection network of programmable switches and wires that allow the logic blocks to be connected in various configurations.
- Input/Output (I/O) Blocks: These blocks interface the FPGA with external components and provide I/O capabilities.
- Embedded Resources: Modern FPGAs may also include embedded memory blocks (Block RAMs), Digital Signal Processing (DSP) blocks, and even hard-wired processors or microcontrollers.
- Interconnections: FPGAs have a highly flexible and hierarchical interconnect structure, allowing logic blocks to be connected in various ways.

2. CPLDs (Complex Programmable Logic Devices):

- Basic Blocks:
- Macrocells: CPLDs are based on a sum-of-products (SOP) or product-of-sums (POS) architecture, where macrocells are used to implement logic functions.
- Interconnect Matrix: CPLDs have a centralized interconnect matrix that connects the macrocells.

- Interconnections: CPLDs have a simpler interconnect structure compared to FPGAs, with a centralized interconnect matrix.

3. PALs (Programmable Array Logic):

- Basic Blocks:
- AND Array: PALs consist of an AND array of programmable fuses that implement product terms (ANDs).
- OR Array: The outputs of the AND array feed into an OR array, which combines the product terms to implement the desired logic functions.
- Interconnections: PALs have a fixed interconnect structure, with the AND array feeding into the OR array.

❖ Comparison:

- **Area**: FPGAs are the largest devices, occupying the most silicon area, followed by CPLDs, and then PALs, which are the smallest.
- **Performance**: FPGAs offer the highest performance due to their flexible and optimized interconnect structure, followed by CPLDs and then PALs.
- **Power**: PALs typically have the lowest power consumption, followed by CPLDs and then FPGAs, which consume the most power due to their complexity and high capacitance interconnects.
- **Applications**:
- FPGAs are used in a wide range of applications requiring high performance, flexibility, and parallel processing, such as digital signal processing, video processing, networking, and prototyping.
- CPLDs are suitable for mid-range applications that require moderate complexity and performance, such as control systems, communications, and glue logic.
- PALs are typically used in simpler and lower-density applications, such as address decoders, state machines, and glue logic.
- Security: FPGAs and CPLDs offer various security features, such as bitstream encryption, secure boot, and tamper protection. PALs, being older devices, lack advanced security features found in modern FPGAs and CPLDs.

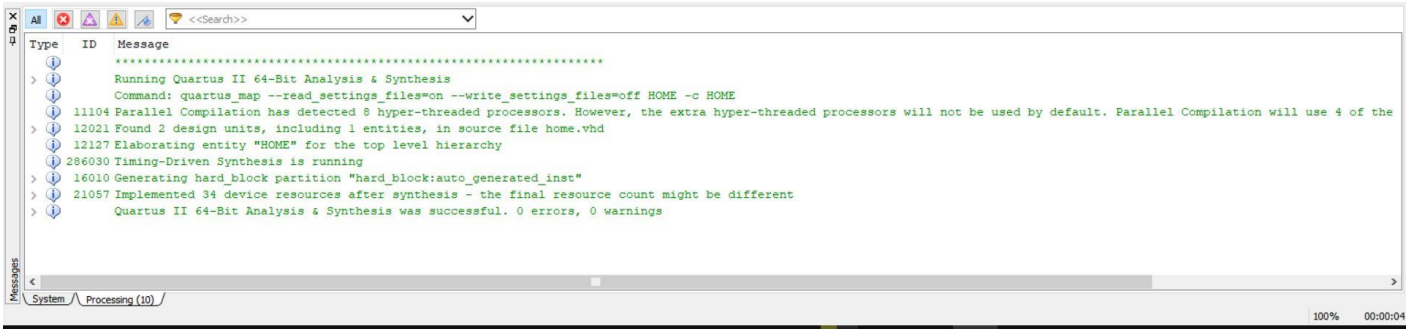
Q2.In digital systems it is often necessary to have circuits that can shift the bits of a vector by one or more bit positions to the left or right. One kind of shifter circuit shifts more bit positions at a time. If the bits that are shifted out are placed into the vacated positions on the left, then the circuit effectively rotates the bits of the input vector by a specified number of bit positions. Such a circuit is often called a barrel shifter. Write a VHDL code to implement a four-bit barrel shifter that rotates the bits by 0, 1, 2, or 3 bit positions as determined by the valuation of two control signals s1 and s0.

❖ Code:

```
library IEEE;
use IEEE.std_logic_1164.all;
entity HOME is
 port (
   data_in  : in  std_logic_vector(3 downto 0);
   control  : in  std_logic_vector(1 downto 0);
   data_out : out std_logic_vector(3 downto 0)
 );
end entity HOME;
architecture rtl of HOME is
signal temp  : std_logic_vector(3 downto 0);
begin

 temp(0) <= data_in(0) when control = "00" else
            data_in(1) when control = "01" else
            data_in(2) when control = "10" else
            data_in(3);
 temp(1) <= data_in(0) when control = "00" else
            data_in(2) when control = "01" else
            data_in(3) when control = "10" else
            data_in(1);
 temp(2) <= data_in(0) when control = "00" else
            data_in(3) when control = "01" else
```

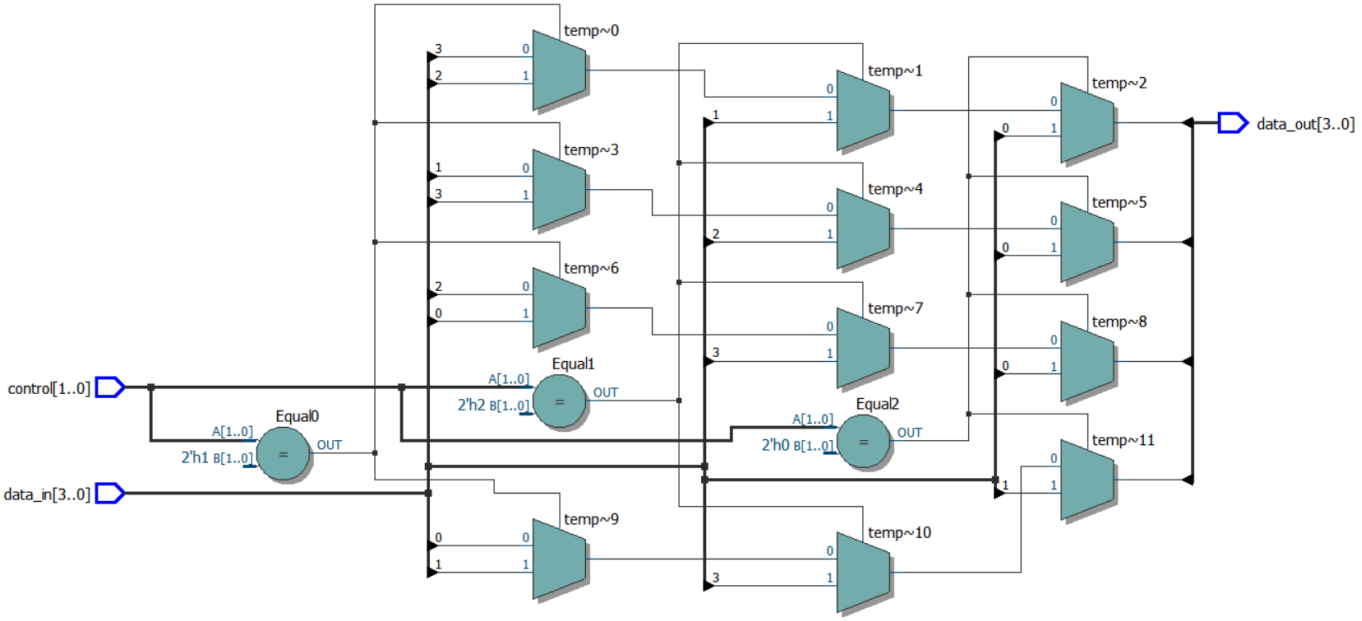
```
            data_in(0) when control = "10" else
            data_in(2);
 temp(3) <= data_in(1) when control = "00" else
            data_in(3) when control = "01" else
            data_in(1) when control = "10" else
            data_in(0);
 data_out <= temp;
end architecture rtl;
```

❖ output after compile:

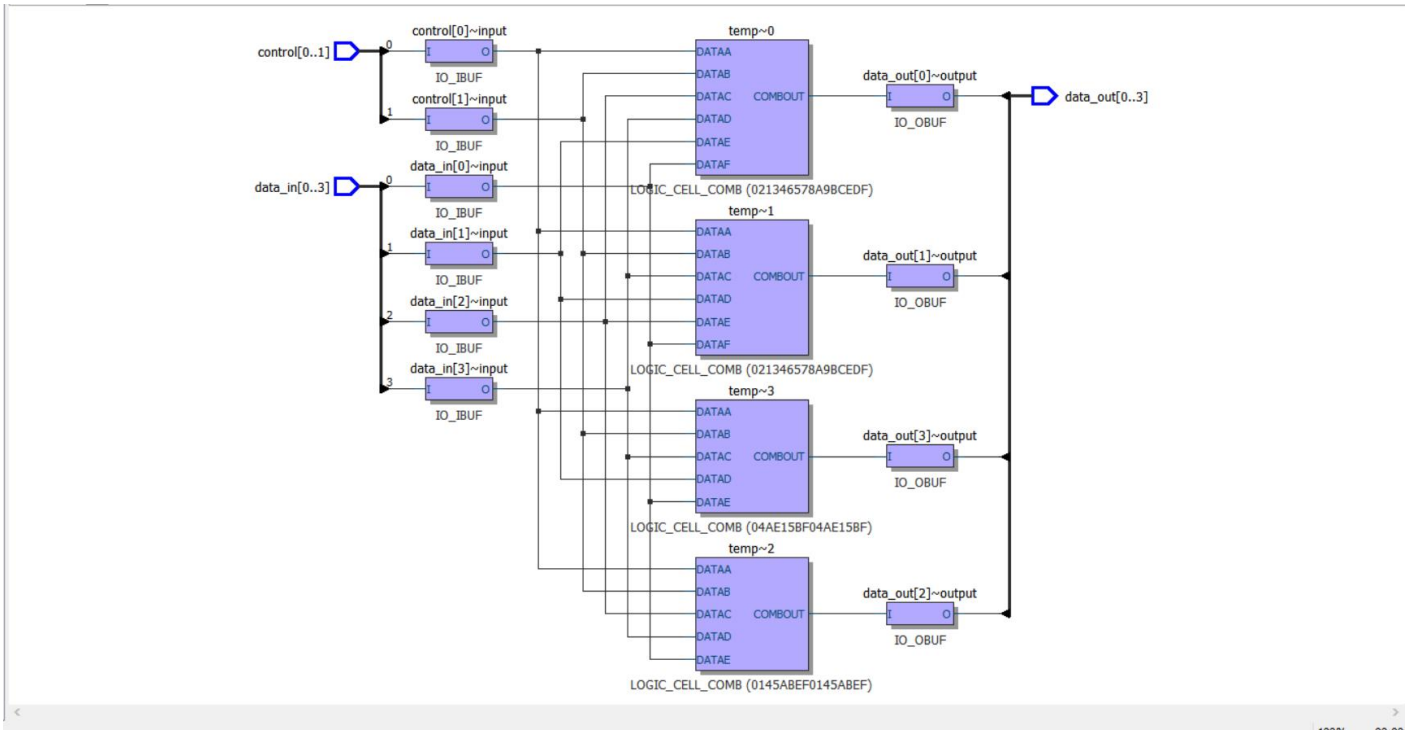


Flow Status	Successful - Fri Apr 19 23:19:15 2024
Quartus II 64-Bit Version	13.1.0 Build 162 10/23/2013 SJ Web Edition
Revision Name	HOME
Top-level Entity Name	HOME
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Preliminary
Logic utilization (in ALMs)	N/A
Total registers	0
Total pins	10
Total virtual pins	0
Total block memory bits	0
Total DSP Blocks	0
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI TX Channels	0
Total PLLs	0
Total DLLs	0

❖ RTL VIEW:



❖ Post-mapping view:



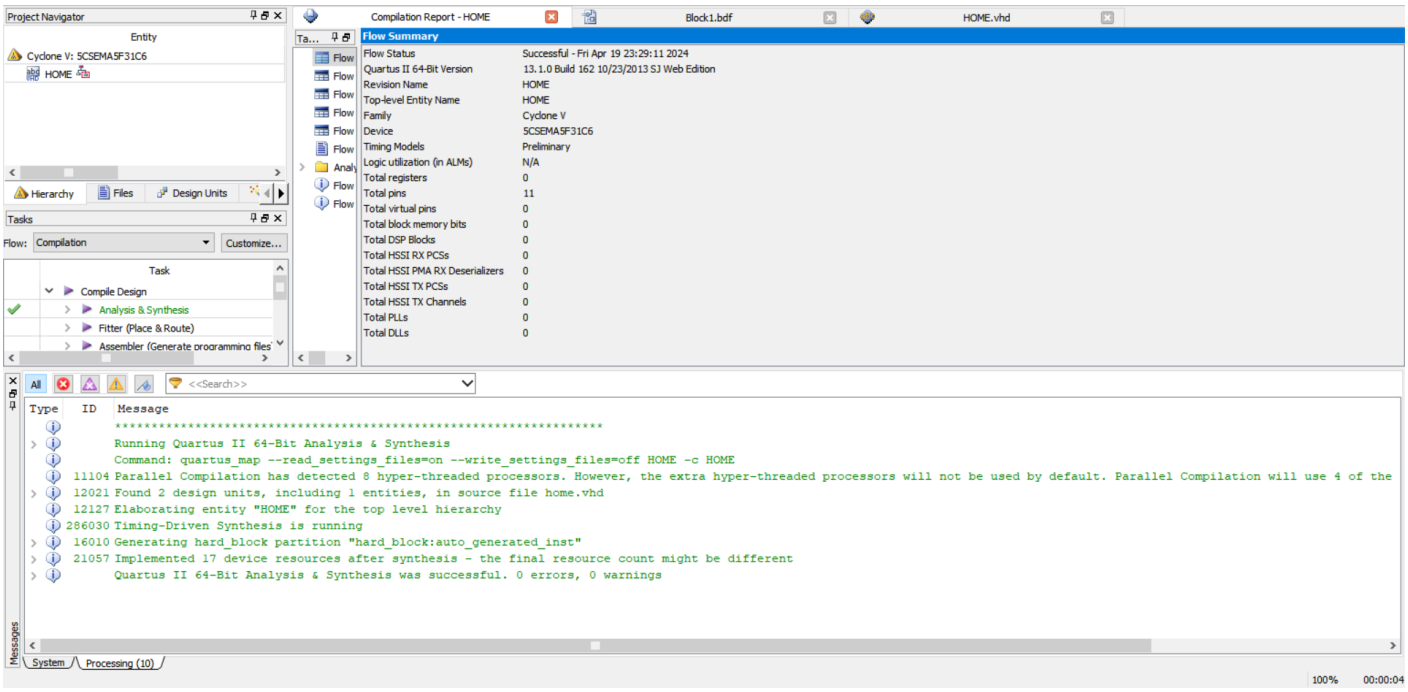
Q3.

- CODE

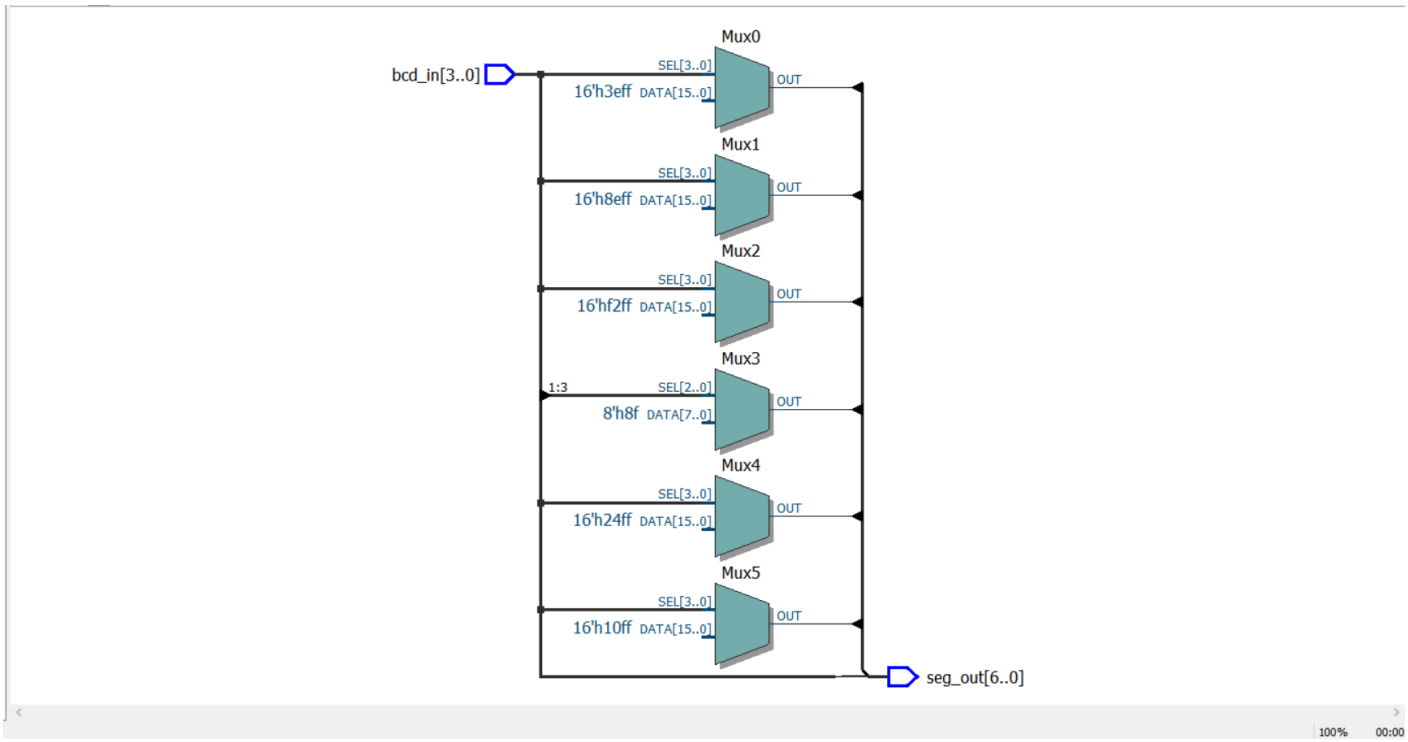
```
library IEEE;
use ieee.numeric_std.all;

entity HOME is
  port (
    bcd_in : in  unsigned(3 downto 0);  -- BCD input
    seg_out: out  unsigned(6 downto 0)  -- 7-segment output
  );
end entity HOME;

-- Architecture definition
architecture rtl of HOME is

begin

  -- Combinational logic for the decoder
  process(bcd_in)
  begin
    case bcd_in is
      when "0000" => seg_out <= "0111000";  -- Display 0
      when "0001" => seg_out <= "0011000";  -- Display 1
      when "0010" => seg_out <= "1010100";  -- Display 2
      when "0011" => seg_out <= "1010010";  -- Display 3
      when "0100" => seg_out <= "1100000";  -- Display 4
```

```
    when "0101" => seg_out <= "1100100";  -- Display 5
    when "0110" => seg_out <= "1110000";  -- Display 6
    when "0111" => seg_out <= "0000000";  -- Display 7
    when others => seg_out <= "1111111";  -- Display error
  end case;
end process;

end architecture rtl;
```

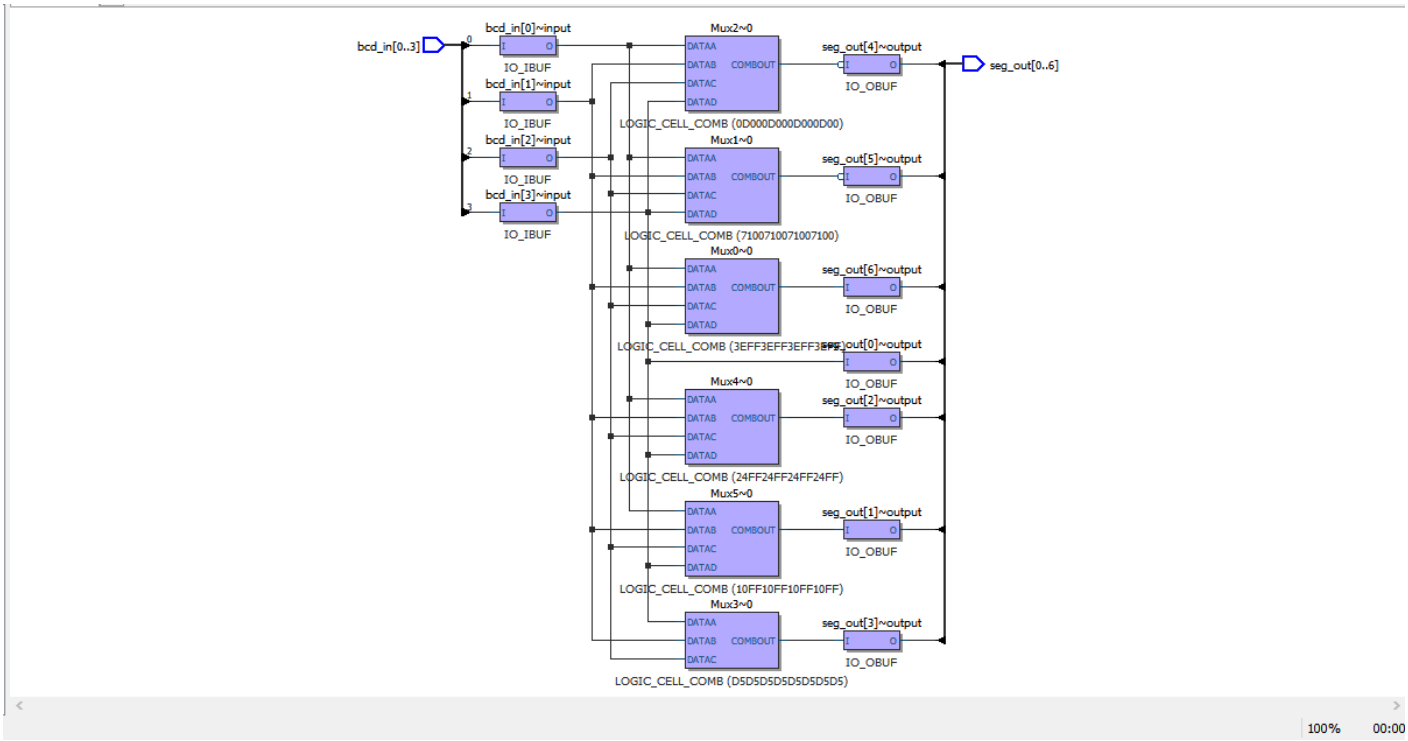
❖ output after compile:



❖ RTL VIEW:



❖ Post-mapping view:

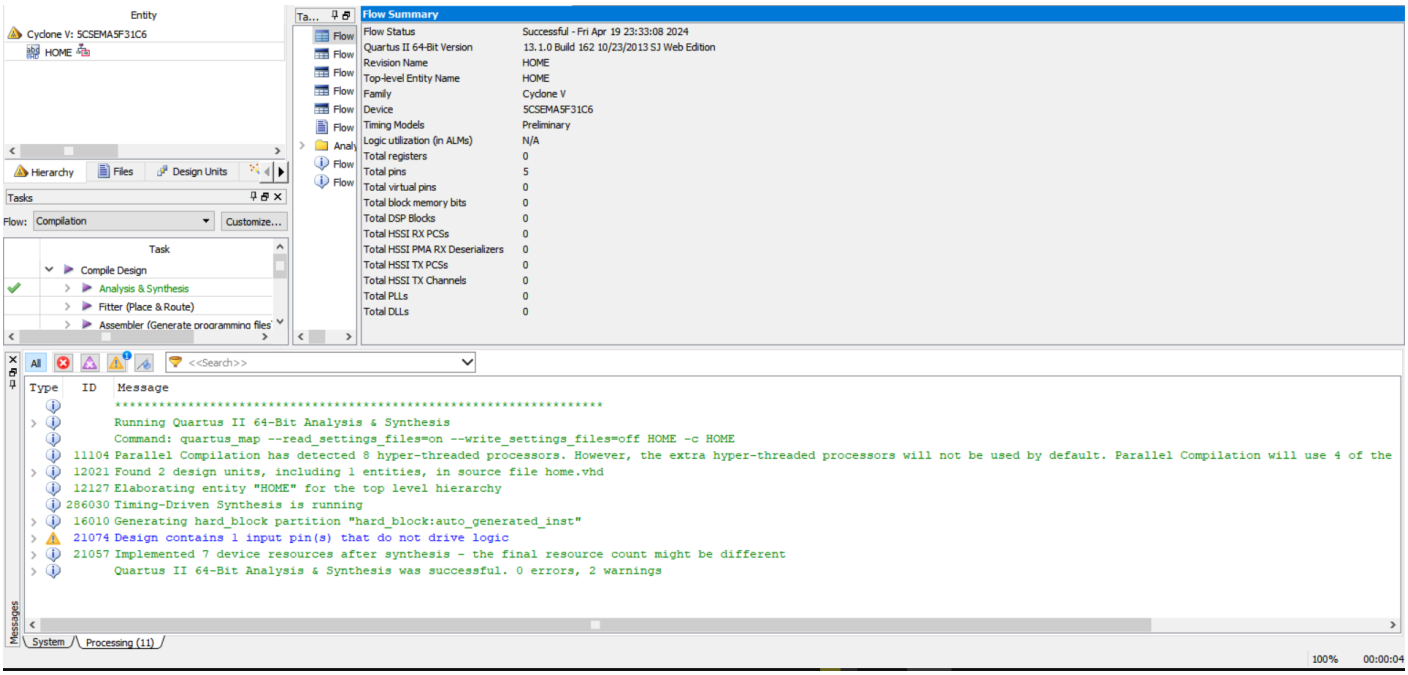


Q4.

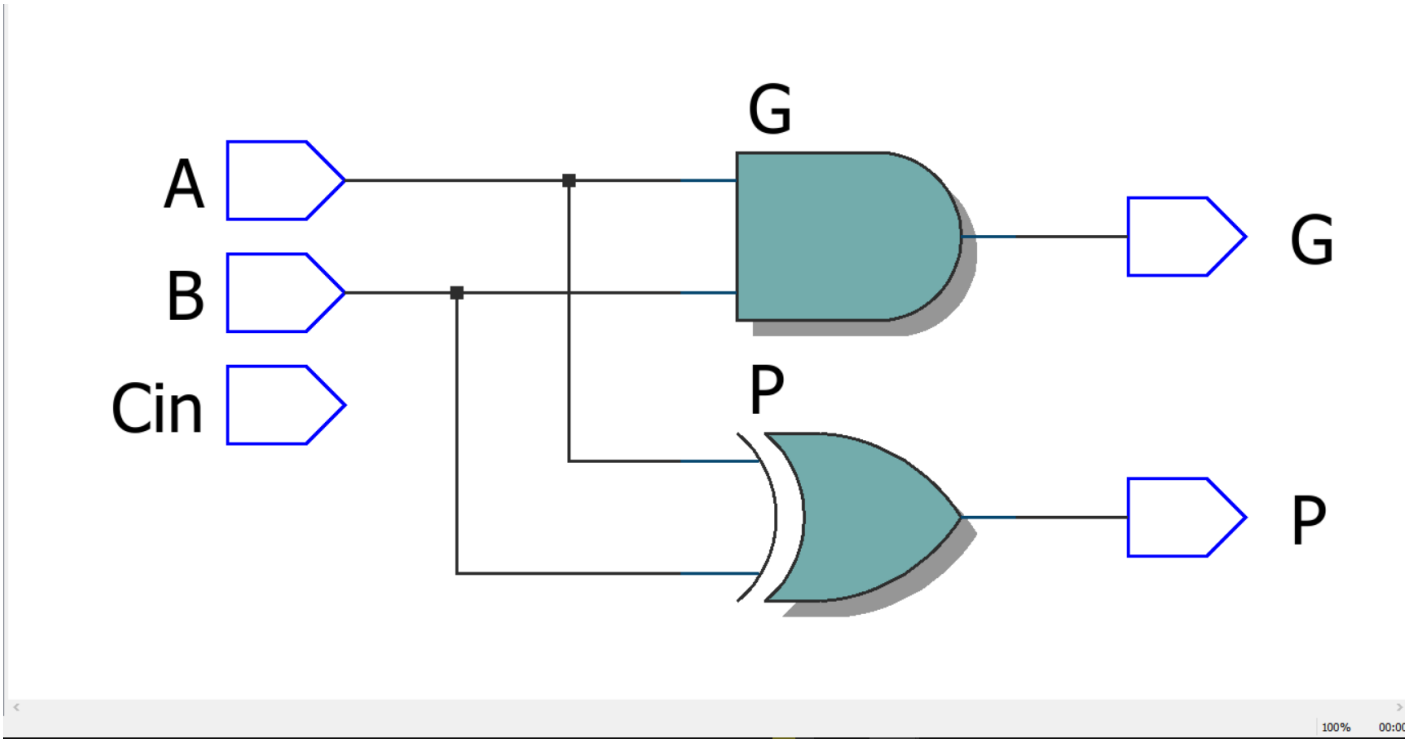
❖ Code:

```
-- Entity declaration for partial full adder
entity HOME is
  port (
    A : in  bit;
    B : in  bit;
    Cin: in  bit;
    P : out bit;
    G : out bit
  );
end entity HOME;

-- Architecture definition for partial full adder
architecture Behavioral of HOME is
begin

  -- Calculate propagate (P)
  P <= A xor B;

  -- Calculate generate (G)
  G <= A and B;

end architecture Behavioral;
```

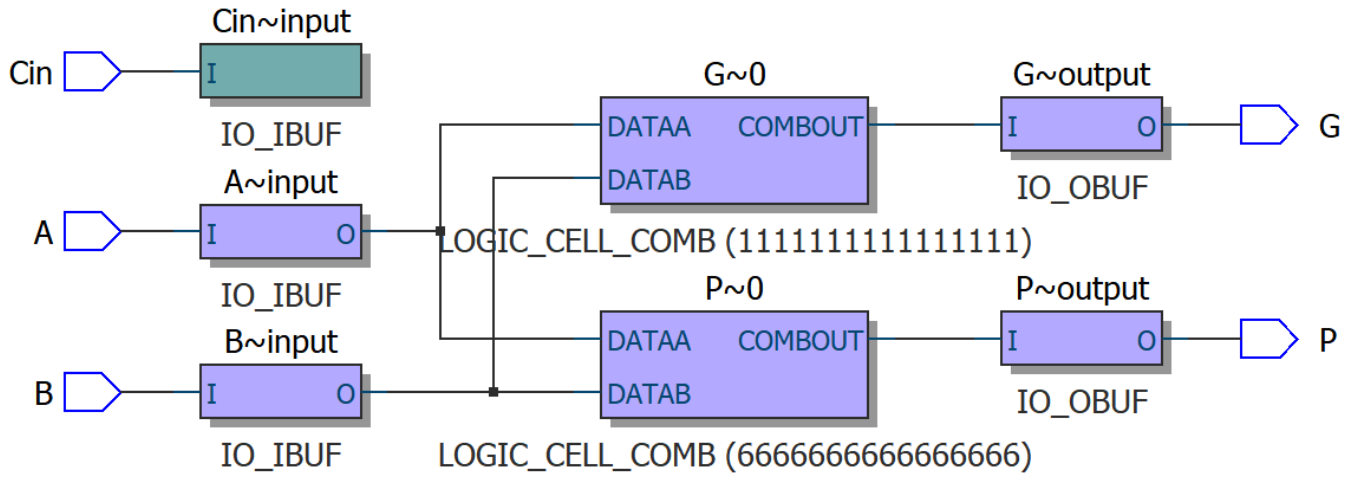
❖ output after compile:



❖ RTL VIEW:



❖ Post-mapping view:



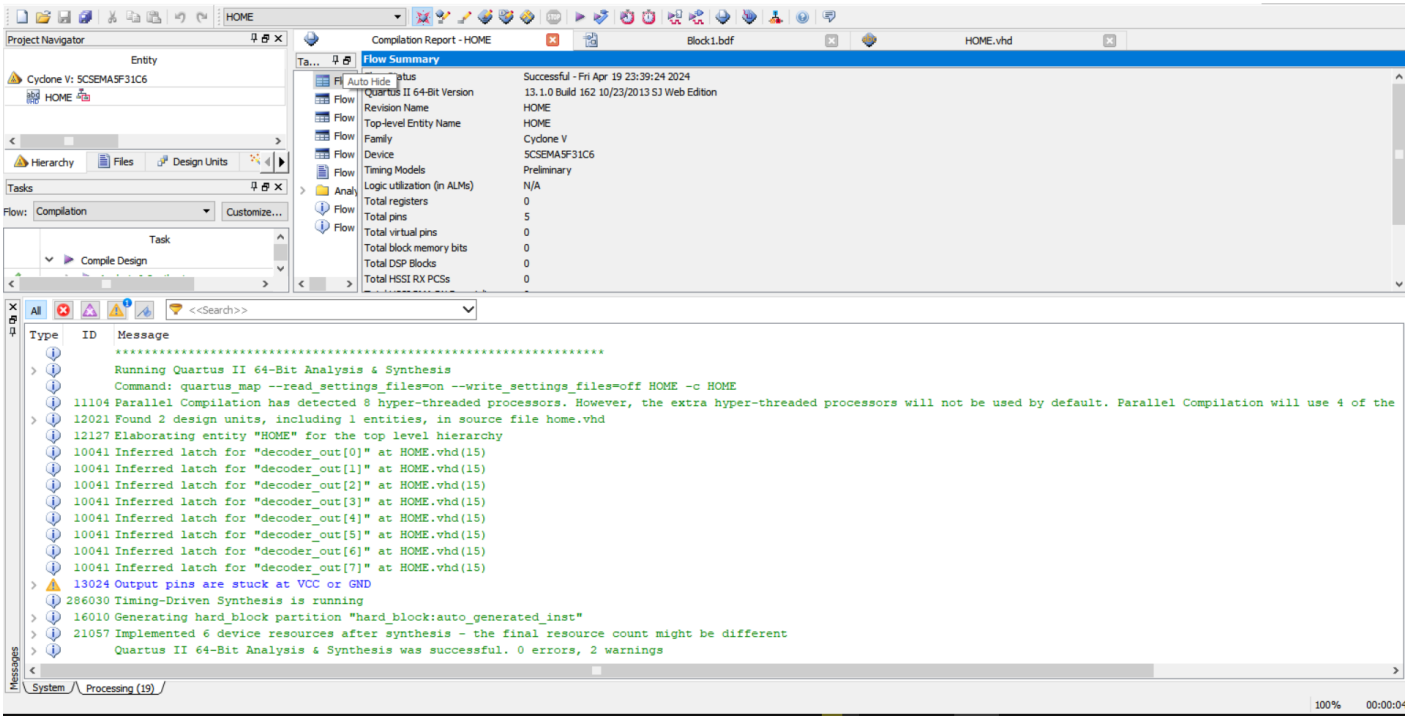
Q5.

❖ Code

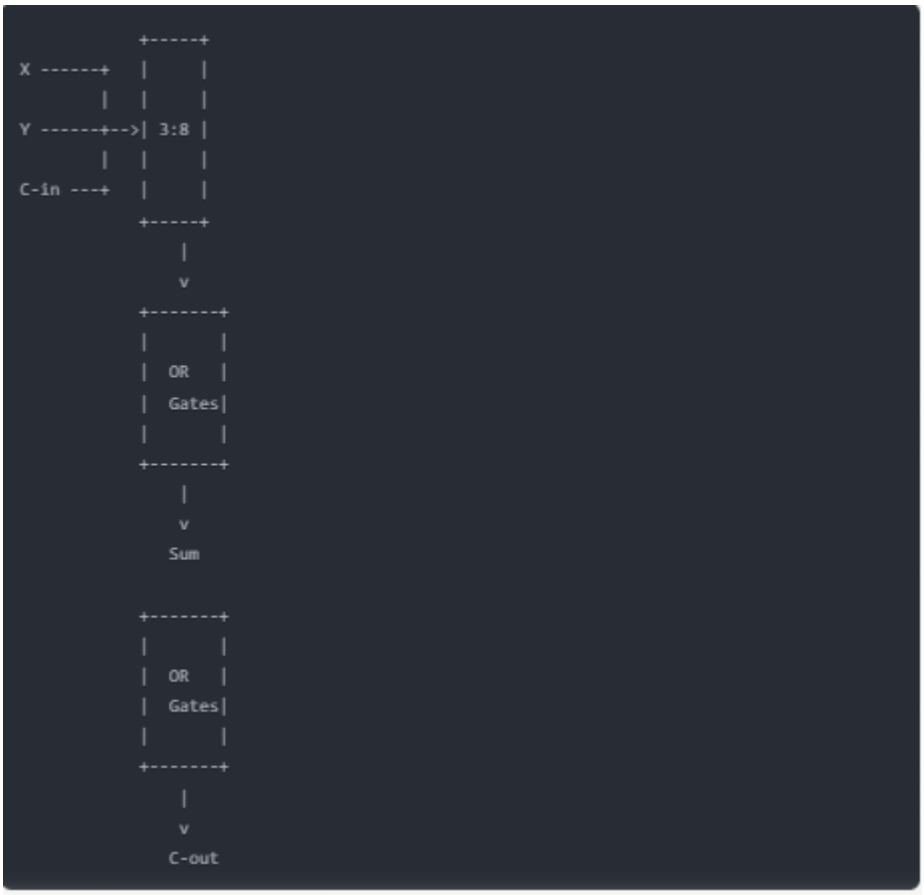
```
library ieee;
use ieee.std_logic_1164.all;

entity HOME is
    port (
        X, Y, C_in : in std_logic;
        Sum, C_out : out std_logic
    );
end entity HOME;

architecture rtl of HOME is
    signal decoder_out : std_logic_vector(7 downto 0);
begin
    -- 3:8 decoder
    decoder_out <= "00000000" when X = '0' and Y = '0' and C_in = '0' else
                   "00000001" when X = '0' and Y = '0' and C_in = '1' else
```

```
                "00000010" when X = '0' and Y = '1' and C_in = '0' else
                "00000011" when X = '0' and Y = '1' and C_in = '1' else
                "00000100" when X = '1' and Y = '0' and C_in = '0' else
                "00000101" when X = '1' and Y = '0' and C_in = '1' else
                "00000110" when X = '1' and Y = '1' and C_in = '0' else
                "00000111" when X = '1' and Y = '1' and C_in = '1';

    -- Sum output
    Sum <= decoder_out(0) or decoder_out(1) or decoder_out(2) or decoder_out(4) or decoder_out(7);

    -- C-out output
    C_out <= decoder_out(3) or decoder_out(5) or decoder_out(6) or decoder_out(7);
end architecture rtl;
```

- output after compile:



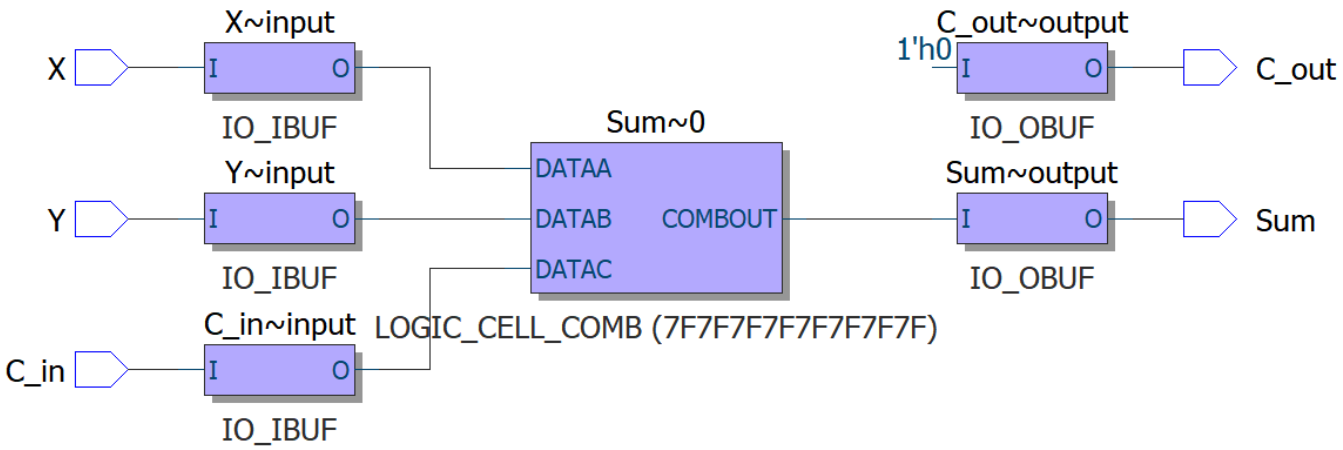
❖ A part: Architecture:



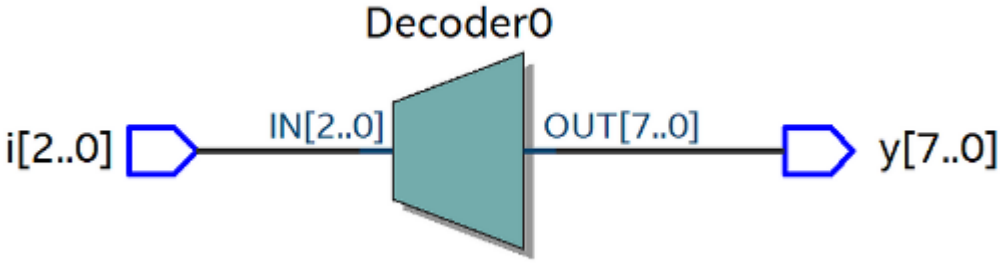
❖ Truth table:

XYC	01234567
--------	--
000	10000000
001	01000000
010	00100000
011	00010000
100	00001000
101	00000100
110	00000010
111	00000001

❖ Post-mapping view:



❖ RTL view:

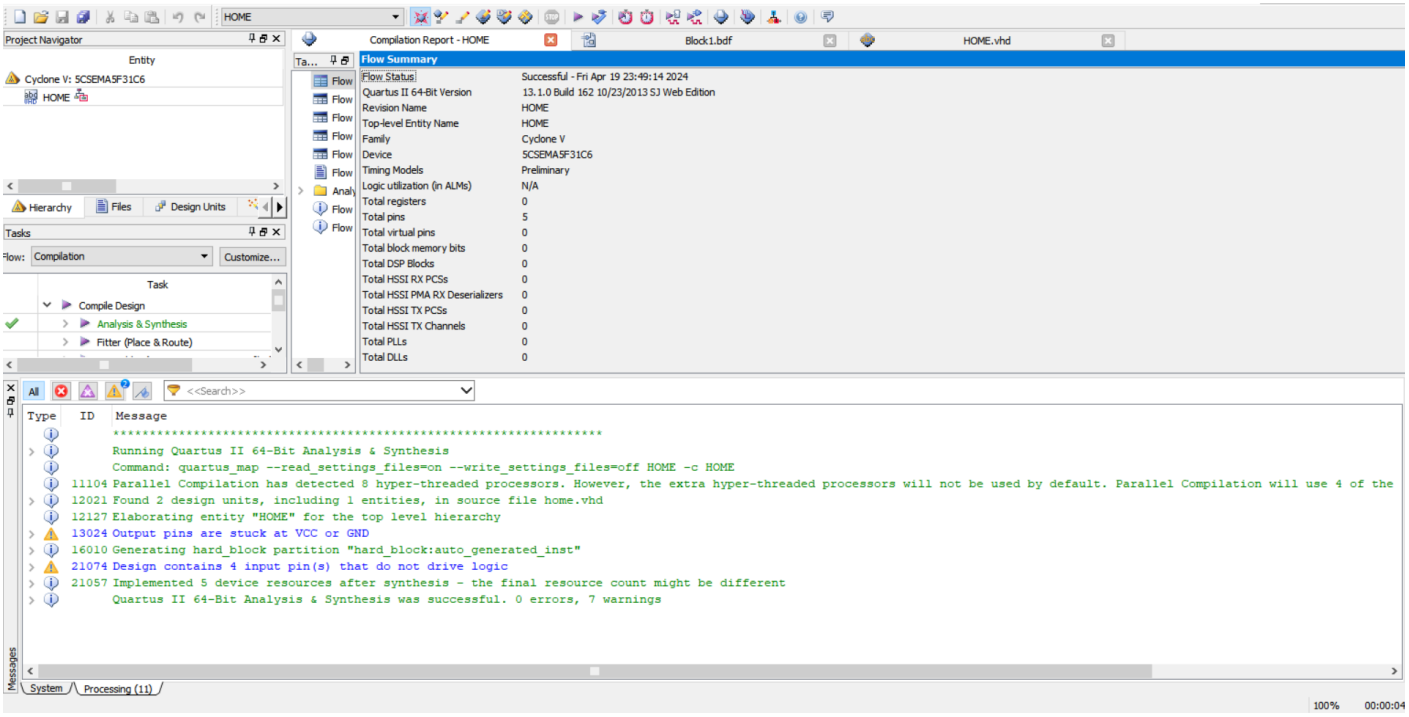


Q6.

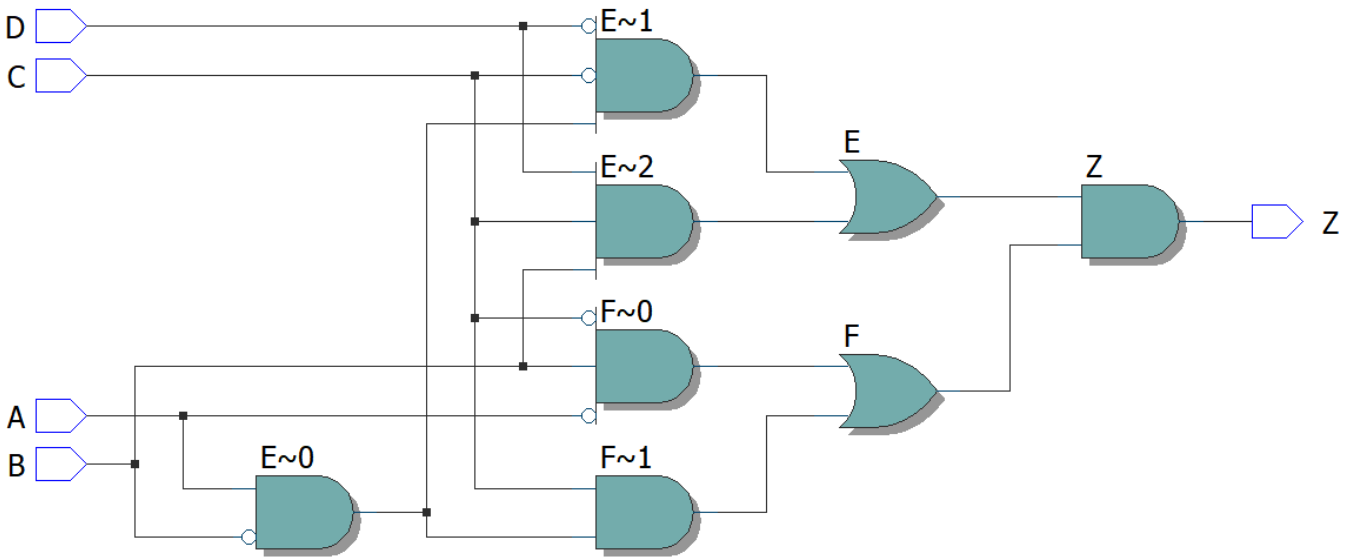
- CODE

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


entity HOME is
    port (
        A, B, C, D : in std_logic;
        Z : out std_logic
    );
end entity HOME;


architecture rtl of HOME is
    signal E, F : std_logic;
begin
    E <= (A and not B and not C and not D) or (B and C and D);
    F <= (not A and B and not C) or (A and not B and C);
    Z <= E and F;
end architecture rtl;
```

- output after compile:


Project Navigator
Entity
Cyclone V: 5CSEMA5F31C6
HOME
Hierarchy
Files
Design Units
Tasks
Flow: Compilation
Customize...
Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Compilation Report - HOME
Block1.bdf
HOME.vhd
Flow Summary
Flow Status Successful - Fri Apr 19 23:49:14 2024
Quartus II 64-Bit Version 13.1.0 Build 162 10/23/2013 SJ Web Edition
Revision Name HOME
Top-level Entity Name HOME
Family Cyclone V
Device 5CSEMA5F31C6
Timing Models Preliminary
Logic utilization (in ALMs) N/A
Total registers 0
Total pins 5
Total virtual pins 0
Total block memory bits 0
Total DSP Blocks 0
Total HSSI RX PCSs 0
Total HSSI PMA RX Deserializers 0
Total HSSI TX PCSs 0
Total HSSI TX Channels 0
Total PLLs 0
Total DLLs 0
Type ID Message
Running Quartus II 64-Bit Analysis & Synthesis
Command: quartus_map --read_settings_files=on --write_settings_files=off HOME -c HOME
11104 Parallel Compilation has detected 8 hyper-threaded processors. However, the extra hyper-threaded processors will not be used by default. Parallel Compilation will use 4 of the
12021 Found 2 design units, including 1 entities, in source file home.vhd
12127 Elaborating entity "HOME" for the top level hierarchy
13024 Output pins are stuck at VCC or GND
16010 Generating hard_block partition "hard_block:auto_generated_inst"
21074 Design contains 4 input pin(s) that do not drive logic
21057 Implemented 5 device resources after synthesis - the final resource count might be different
Quartus II 64-Bit Analysis & Synthesis was successful. 0 errors, 7 warnings
System
Processing (11)
100%
00:00:04

❖ RTL view:


D
C
A
B
E~1
E~2
E~0
F~0
F~1
E
F
Z
Z

❖ Post-mapping view: