VHDL code of the half adder-based 2- bit adder:

```
----------------------------------------------------------------------------------
-- Company: Department of Electrical and Computer Engineering, University of Alberta
-- Engineer: Xinyue Chen
-- Create Date: 2020/10/11 11:19:27 AM
-- Design Name: half adder-based 2-bit adder
-- Module Name: ha_2bit - Behavioral
-- Project Name: half adder-based 2-bit adder
-- Target Devices: Zybo Z-7
-- Tool Versions:
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity ha_2bit is
    Port ( A : in STD_LOGIC_VECTOR (1 downto 0);
           B : in STD_LOGIC_VECTOR (1 downto 0);
           Cout : out STD_LOGIC;
           S : out STD_LOGIC_VECTOR (1 downto 0));
end ha_2bit;
architecture Behavioral of ha_2bit is
begin
    --write the karnaugh map equation for S(1)
     S(1) <= ((A(1) xor B(1)) and (not A(0))) or ((A(1) xor B(1)) and (not B(0))) or ((A(1) xnor B(1)) and A(0)
     and B(0)) ;
    --write the karnaugh map equation for S(0)
    S(0) <= A(0) xor B(0);
    --write the karnaugh map equation for Cout
    Cout <= (A(1) and B(1)) or ((A(1) xor B(1)) and A(0) and B(0));
    end Behavioral;
```

VHDL testbench of the half adder-based 2- bit adder:

```
----------------------------------------------------------------------------------
-- Company: Department of Electrical and Computer Engineering, University of Alberta
-- Engineer: Xinyue Chen
-- Create Date: 08/09/2020 07:06:23 PM
-- Design Name: testbench of half adder-based 2-bit adder
-- Module Name: ha_2bit_tb - Behavioral
-- Project Name: testbench of half adder-based 2-bit adder
-- Target Devices: Zybo Z-7
-- Tool Versions:
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity ha_2bit_tb is
end ha_2bit_tb;
architecture Behavioral of ha_2bit_tb is
component ha_2bit is
  Port ( A : in std_logic_vector(1 downto 0);
         B : in std_logic_vector(1 downto 0);
         Cout : out std_logic;
         S : out std_logic_vector(1 downto 0));
end component;
signal A_in, B_in : std_logic_vector(1 downto 0);
signal Co : std_logic;
signal Sum : std_logic_vector(1 downto 0);
begin
    ADDER_HA : component ha_2bit port map(A => A_in,
                                          B => B_in,
                                          Cout => Co,
                                          S => Sum);
    process
        constant time_period : time := 20ns;
        begin
        A_in <= "00"; B_in <= "00";
        assert ((Sum = "00") and (Co='0'))
        report "Error for input A_in=00 and B_in=00" severity error;
        wait for time_period;

        A_in <= "01"; B_in <= "00";
        assert ((Sum = "01") and (Co='0'))
        report "Error for input A_in=01 and B_in=00" severity error;
        wait for time_period;

        A_in <= "10"; B_in <= "00";
        assert ((Sum = "10") and (Co='0'))
        report "Error for input A_in=10 and B_in=00" severity error;
        wait for time_period;
```

```
A_in <= "11"; B_in <= "00";
assert ((Sum = "11") and (Co='0'))
report "Error for input A_in=11 and B_in=00" severity error;
wait for time_period;

A_in <= "00"; B_in <= "01";
assert ((Sum = "01") and (Co='0'))
report "Error for input A_in=00 and B_in=01" severity error;
wait for time_period;

A_in <= "01"; B_in <= "01";
assert ((Sum = "00") and (Co='0'))
report "Error for input A_in=01 and B_in=01" severity error;
wait for time_period;

A_in <= "10";    B_in <= "01";
assert ((Sum = "11") and (Co='0'))
report "Error for input A_in=10 and B_in=01" severity error;
wait for time_period;

A_in <= "11"; B_in <= "01";
assert ((Sum = "10") and (Co='0'))
report "Error for input A_in=11 and B_in=01" severity error;
wait for time_period;
A_in <= "00"; B_in <= "10";
assert ((Sum = "10") and (Co='0'))
report "Error for input A_in=00 and B_in=10" severity error;
wait for time_period;

A_in <= "01"; B_in <= "10";
assert ((Sum = "11") and (Co='0'))
report "Error for input A_in=01 and B_in=10" severity error;
wait for time_period;

A_in <= "10"; B_in <= "10";
assert ((Sum = "00") and (Co='1'))
report "Error for input A_in=10 and B_in=10" severity error;
wait for time_period;

A_in <= "11"; B_in <= "10";
assert ((Sum = "01") and (Co='1'))
report "Error for input A_in=11 and B_in=10" severity error;
wait for time_period;
```

```
        A_in <= "00"; B_in <= "11";
        assert ((Sum = "11") and (Co='0'))
        report "Error for input A_in=00 and B_in=11" severity error;
        wait for time_period;

        A_in <= "01"; B_in <= "11";
        assert ((Sum = "10") and (Co='0'))
        report "Error for input A_in=01 and B_in=11" severity error;
        wait for time_period;

        A_in <= "10"; B_in <= "11";
        assert ((Sum = "01") and (Co='1'))
        report "Error for input A_in=10 and B_in=11" severity error;
        wait for time_period;

        A_in <= "11"; B_in <= "11";
        assert ((Sum = "00") and (Co='1'))
        report "Error for input A_in=11 and B_in=11" severity error;
        wait for time_period;
        wait;
    end process;
    end Behavioral;
```

Constraint file of the half adder-based 2-bit adder:

In constraint file, each entity port is mapped to the Zybo Z7 board as shown below:

```
##Switches
set_property -dict { PACKAGE_PIN G15     IOSTANDARD LVCMOS33 } [get_ports { A[0] }];
#IO_L19N_T3_VREF_35 Sch=sw[0]
set_property -dict { PACKAGE_PIN P15     IOSTANDARD LVCMOS33 } [get_ports { A[1] }];
#IO_L24P_T3_34 Sch=sw[1]
set_property -dict { PACKAGE_PIN W13     IOSTANDARD LVCMOS33 } [get_ports { B[0] }];
#IO_L4N_T0_34 Sch=sw[2]
set_property -dict { PACKAGE_PIN T16     IOSTANDARD LVCMOS33 } [get_ports { B[1] }];
#IO_L9P_T1_DQS_34 Sch=sw[3]
##LEDs
set_property -dict { PACKAGE_PIN M14     IOSTANDARD LVCMOS33 } [get_ports { S[0] }];
#IO_L23P_T3_35 Sch=led[0]
set_property -dict { PACKAGE_PIN M15     IOSTANDARD LVCMOS33 } [get_ports { S[1] }];
#IO_L23N_T3_35 Sch=led[1]
set_property -dict { PACKAGE_PIN G14     IOSTANDARD LVCMOS33 } [get_ports { Cout }];
#IO_0_35 Sch=led[2]
```

PART 2 & PART 3:

The VHDL code, testbench and constraint file of part 3 is demonstrated together with those of part 2, marked with blue color.

VHDL code of the first multiplexer for output sum (0):

```
----------------------------------------------------------------------------------
-- Company: Department of Electrical and Computer Engineering, University of Alberta
-- Engineer: Xinyue Chen
-- Create Date: 08/10/2020 11:28:43 AM
-- Design Name: 2-bit adder mux_1
-- Module Name: s0_16_1_mux - Behavioral
-- Project Name: 2-bit adder
-- Target Devices: Zybo Z-7
-- Tool Versions:
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity s0_16_1_mux is
    Port ( cin : in STD_LOGIC;
           select_in : in STD_LOGIC_VECTOR (3 downto 0);
           y_s0: out STD_LOGIC);
end s0_16_1_mux;
architecture Behavioral of s0_16_1_mux is
signal input_S_0: std_logic_vector(15 downto 0) := (others => '0');
begin
    process(select_in,cin) is
    begin
    input_S_0 <= cin & (not (cin)) & cin & (not (cin)) & (not (cin)) & cin & (not (cin)) & cin & cin & (not
    (cin)) & cin & (not (cin)) & (not (cin)) & cin & (not (cin)) & cin;
    -- write the input line functions generated for 16:1 to derive the S(0) output.
    -- The input lines can be '0' / '1' / Cin / (not) Cin.
    -- It is possible to use if...else or case statements here.
        case select_in is
    -- write the VHDL code for all the 16 cases.
        when "0000" =>  y_s0 <= input_S_0(0);
        when "0001" =>  y_s0 <= input_S_0(1);
        when "0010" =>  y_s0 <= input_S_0(2);
        when "0011" =>  y_s0 <= input_S_0(3);
        when "0100" =>  y_s0 <= input_S_0(4);
        when "0101" =>  y_s0 <= input_S_0(5);
        when "0110" =>  y_s0 <= input_S_0(6);
        when "0111" =>  y_s0 <= input_S_0(7);
```

```
            when "1000" =>    y_s0 <= input_S_0(8);
            when "1001" =>    y_s0 <= input_S_0(9);
            when "1010" =>    y_s0 <= input_S_0(10);
            when "1011" =>    y_s0 <= input_S_0(11);
            when "1100" =>    y_s0 <= input_S_0(12);
            when "1101" =>    y_s0 <= input_S_0(13);
            when "1110" =>    y_s0 <= input_S_0(14);
            when "1111" =>    y_s0 <= input_S_0(15);
            when others =>    y_s0 <= '-';
            end case;
        end process;
end Behavioral;
```

VHDL code of the second multiplexer for output sum (1):

```
----------------------------------------------------------------------------------
-- Company: Department of Electrical and Computer Engineering, University of Alberta
-- Engineer: Xinyue Chen
-- Create Date: 2020/10/11 11:55:10
-- Design Name: 2-bit adder mux_2
-- Module Name: s1_16_1_mux - Behavioral
-- Project Name: 2-bit adder
-- Target Devices: Zybo Z-7
-- Tool Versions:
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity s1_16_1_mux is
        Port ( cin : in STD_LOGIC;
                select_in : in STD_LOGIC_VECTOR (3 downto 0);
                y_s1 : out STD_LOGIC);
end s1_16_1_mux;
architecture Behavioral of s1_16_1_mux is
signal input_S_1: std_logic_vector(15 downto 0) := (others => '0');
begin
        process(select_in,cin) is
        begin
        input_S_1 <= '1' & cin & '0' & (not (cin)) & cin & '0' & (not (cin)) & '1' & '0' & (not (cin)) & '1' & cin &
                        (not (cin)) & '1' & cin & '0';
        -- write the input line functions generated for 16:1 to derive the S(0) output.
        -- The input lines can be '0' / '1' / Cin / (not) Cin.
```

```
    -- It is possible to use if...else or case statements here.
        case select_in is
    -- Write the VHDL codes for all the input cases.
        when "0000" =>   y_s1 <= input_S_1(0);
        when "0001" =>   y_s1 <= input_S_1(1);
        when "0010" =>   y_s1 <= input_S_1(2);
        when "0011" =>   y_s1 <= input_S_1(3);
        when "0100" =>   y_s1 <= input_S_1(4);
        when "0101" =>   y_s1 <= input_S_1(5);
        when "0110" =>   y_s1 <= input_S_1(6);
        when "0111" =>   y_s1 <= input_S_1(7);
        when "1000" =>   y_s1 <= input_S_1(8);
        when "1001" =>   y_s1 <= input_S_1(9);
        when "1010" =>   y_s1 <= input_S_1(10);
        when "1011" =>   y_s1 <= input_S_1(11);
        when "1100" =>   y_s1 <= input_S_1(12);
        when "1101" =>   y_s1 <= input_S_1(13);
        when "1110" =>   y_s1 <= input_S_1(14);
        when "1111" =>   y_s1 <= input_S_1(15);
        when others =>   y_s1 <= '-';
        end case;
    end process;
end Behavioral;
```

VHDL code of the third multiplexer for output Cout:

```
----------------------------------------------------------------------------------
-- Company: Department of Electrical and Computer Engineering, University of Alberta
-- Engineer: Xinyue Chen
-- Create Date: 2020/10/11 13:35:06
-- Design Name: 2-bit adder mux_3
-- Module Name: cout_16_1_mux - Behavioral
-- Project Name: 2-bit adder
-- Target Devices: Zybo Z-7
-- Tool Versions:
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity cout_16_1_mux is
    Port ( cin : in STD_LOGIC;
         select_in : in STD_LOGIC_VECTOR (3 downto 0);
```

```
            y_cout : out STD_LOGIC);
end cout_16_1_mux;
architecture Behavioral of cout_16_1_mux is
signal input_count: std_logic_vector(15 downto 0) := (others => '0');
begin
    process(select_in,cin) is
    begin
     input_count <= '1' & '1' & '1' & cin & '1' & '1' & cin & '0' & '1' & cin & '0' & '0' & cin & '0' & '0' & '0';
    -- write the input line functions generated for 16:1 to derive the carry output.
    -- The input lines can be '0' / '1' / Cin / (not) Cin.
    -- It is possible to use if...else or case statements here.
        case select_in is
    -- write the vhdl code for all the input cases.
        when "0000" =>    y_cout <= input_count(0);
        when "0001" =>    y_cout <= input_count(1);
        when "0010" =>    y_cout <= input_count(2);
        when "0011" =>    y_cout <= input_count(3);
        when "0100" =>    y_cout <= input_count(4);
        when "0101" =>    y_cout <= input_count(5);
        when "0110" =>    y_cout <= input_count(6);
        when "0111" =>    y_cout <= input_count(7);
        when "1000" =>    y_cout <= input_count(8);
        when "1001" =>    y_cout <= input_count(9);
        when "1010" =>    y_cout <= input_count(10);
        when "1011" =>    y_cout <= input_count(11);
        when "1100" =>    y_cout <= input_count(12);
        when "1101" =>    y_cout <= input_count(13);
        when "1110" =>    y_cout <= input_count(14);
        when "1111" =>    y_cout <= input_count(15);
        when others =>    y_cout <= '-';
        end case;
    end process;
end Behavioral;
```

VHDL code of the 2-bit full adder & the comparator:

```
----------------------------------------------------------------------------------
-- Company: Department of Electrical and Computer Engineering, University of Alberta
-- Engineer: Xinyue Chen
-- Create Date: 2020/10/11 13:51:49
-- Design Name: 2-bit adder and comparator
-- Module Name: fa_2bit - Behavioral
-- Project Name: 2-bit adder and comparator
-- Target Devices: Zybo Z-7
```

```vhdl
-- Tool Versions:
-- Description:
-- Dependencies:
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity fa_2bit is
    Port ( A : in STD_LOGIC_VECTOR (1 downto 0);
           B : in STD_LOGIC_VECTOR (1 downto 0);
           C_in : in STD_LOGIC;
           Sum : out STD_LOGIC_VECTOR (1 downto 0);
           C_out : out STD_LOGIC;
           compare_result : out std_logic_vector(2 downto 0));
-- compare the input vector A and B : mapped to RGB led in xdc file
end fa_2bit;
architecture Behavioral of fa_2bit is
signal ld_red: std_logic;
signal ld_green: std_logic;
signal ld_blue: std_logic;
component cout_16_1_mux is
  Port ( cin : in std_logic;
         select_in : in std_logic_vector(3 downto 0);
         y_cout : out std_logic );
end component cout_16_1_mux;
component s1_16_1_mux is
  Port ( cin : in std_logic;
         select_in : in std_logic_vector(3 downto 0);
         y_s1 : out std_logic );
end component s1_16_1_mux;
component s0_16_1_mux is
  Port ( cin : in std_logic;
         select_in : in std_logic_vector(3 downto 0);
         y_s0 : out std_logic );
end component s0_16_1_mux;

begin
-- PART 3....
-- The "LD6" - RGB led on board is used as an indication if A>B or A<B or A=B.
 ld_red <= ((not B(1)) and A(1)) or (B(1) and (not B(0)) and A(1) and A(0)) or ((not B(1)) and (not B(0)) and
          (not A(1)) and A(0));
 compare_result(0) <= ld_red;
 ld_green <= (B(1) and (not A(1))) or (B(1) and B(0) and A(1) and (not A(0))) or ((not B(1)) and B(0) and (not
            A(1)) and (not A(0)));
 compare_result(1) <= ld_green;

 ld_blue <= (A(0) xnor B(0)) and (A(1) xnor B(1));
 compare_result(2) <= ld_blue;
-- to turn the LED red when A>B, green when A<B and blue when A=B.
carry_map : component cout_16_1_mux port map( cin => C_in,
                                              select_in(3 downto 2) => A,
                                              select_in(1 downto 0) => B,
                                           y_cout => C_out);
-- port map the component for generating the C_out
S_1_map : component s1_16_1_mux port map( cin => C_in,
                                           select_in(3 downto 2) => A,
                                           select_in(1 downto 0) => B,
                                           y_s1 => Sum(1));
-- port map the component for generating the S(1)
    S_0_map : component s0_16_1_mux port map( cin => C_in,
                                           select_in(3 downto 2) => A,
                                           select_in(1 downto 0) => B,
                                           y_s0 => Sum(0));
-- port map the component for generating the S(0)
end Behavioral;
```

VHDL testbench of the 2-bit full adder & the comparator:

```
----------------------------------------------------------------------------------
-- Company: Department of Electrical and Computer Engineering, University of Alberta
-- Engineer: Xinyue Chen
-- Create Date: 2020/10/18 10:56:37
-- Design Name: testbench for 2-bit adder and comparator
-- Module Name: fa_2bit_tb - Behavioral
-- Project Name: testbench for 2-bit adder and comparator
-- Target Devices: Zybo Z-7
-- Tool Versions:
-- Description:
-- Dependencies:
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
```

```
entity fa_2bit_tb is
end fa_2bit_tb;

architecture Behavioral of fa_2bit_tb is
component fa_2bit is
  Port ( A : in STD_LOGIC_VECTOR (1 downto 0);
            B : in STD_LOGIC_VECTOR (1 downto 0);
            C_in : in STD_LOGIC;
            Sum : out STD_LOGIC_VECTOR (1 downto 0);
            C_out : out STD_LOGIC;
     compare_result : out std_logic_vector(2 downto 0));
-- compare the input vector A and B : mapped to RGB led in xdc file
end component fa_2bit;
signal A_tb : std_logic_vector(1 downto 0) := "00";
signal B_tb : std_logic_vector(1 downto 0) := "00";
signal C_in_tb : std_logic := '0';
signal Sum_tb : std_logic_vector(1 downto 0);
signal C_out_tb : std_logic;
signal compare_result_tb : std_logic_vector(2 downto 0);
begin
    TEST_FA : component fa_2bit port map(A => A_tb,
                                         B => B_tb,
                                         C_in => C_in_tb,
                                         C_out => C_out_tb,
                                         Sum => Sum_tb,
                                         compare_result => compare_result_tb);
    stimu: process
        constant time_period : time := 20ns;
        begin
        wait for time_period;
          A_tb <= "00"; B_tb <= "00"; C_in_tb <= '0';
        wait for time_period;
          A_tb <= "00"; B_tb <= "00"; C_in_tb <= '1';
        wait for time_period;
          A_tb <= "00"; B_tb <= "01"; C_in_tb <= '0';
        wait for time_period;
          A_tb <= "00"; B_tb <= "01"; C_in_tb <= '1';
        wait for time_period;
          A_tb <= "00"; B_tb <= "10"; C_in_tb <= '0';
        wait for time_period;
          A_tb <= "00"; B_tb <= "10"; C_in_tb <= '1';
        wait for time_period;
          A_tb <= "00"; B_tb <= "11"; C_in_tb <= '0';
        wait for time_period;
```

```
      A_tb <= "00"; B_tb <= "11"; C_in_tb <= '1';
  wait for time_period;
      A_tb <= "01"; B_tb <= "00"; C_in_tb <= '0';
  wait for time_period;
      A_tb <= "01"; B_tb <= "00"; C_in_tb <= '1';
  wait for time_period;
      A_tb <= "01"; B_tb <= "01"; C_in_tb <= '0';
  wait for time_period;
      A_tb <= "01"; B_tb <= "01"; C_in_tb <= '1';
  wait for time_period;
      A_tb <= "01"; B_tb <= "10"; C_in_tb <= '0';
  wait for time_period;
      A_tb <= "01"; B_tb <= "10"; C_in_tb <= '1';
  wait for time_period;
      A_tb <= "01"; B_tb <= "11"; C_in_tb <= '0';
  wait for time_period;
      A_tb <= "01"; B_tb <= "11"; C_in_tb <= '1';
  wait for time_period;
      A_tb <= "10"; B_tb <= "00"; C_in_tb <= '0';
  wait for time_period;
      A_tb <= "10"; B_tb <= "00"; C_in_tb <= '1';
  wait for time_period;
      A_tb <= "10"; B_tb <= "01"; C_in_tb <= '0';
  wait for time_period;
      A_tb <= "10";    B_tb <= "01"; C_in_tb <= '1';
  wait for time_period;
      A_tb <= "10"; B_tb <= "10"; C_in_tb <= '0';
wait for time_period;
      A_tb <= "10"; B_tb <= "10"; C_in_tb <= '1';
wait for time_period;
      A_tb <= "10"; B_tb <= "11"; C_in_tb <= '0';
wait for time_period;
      A_tb <= "10"; B_tb <= "11"; C_in_tb <= '1';
wait for time_period;
      A_tb <= "11"; B_tb <= "00"; C_in_tb <= '0';
wait for time_period;
      A_tb <= "11";    B_tb <= "00"; C_in_tb <= '1';
wait for time_period;
      A_tb <= "11"; B_tb <= "01"; C_in_tb <= '0';
wait for time_period;
        A_tb <= "11"; B_tb <= "01"; C_in_tb <= '1';
wait for time_period;
        A_tb <= "11"; B_tb <= "10"; C_in_tb <= '0';
wait for time_period;
```

```
            A_tb <= "11"; B_tb <= "10"; C_in_tb <= '1';
        wait for time_period;
            A_tb <= "11"; B_tb <= "11"; C_in_tb <= '0';
        wait for time_period;
            A_tb <= "11"; B_tb <= "11"; C_in_tb <= '1';
         wait;
    end process stimu;
end Behavioral;
```

Constraint file of the 2-bit full adder & the comparator:

In constraint file, each entity port is mapped to the Zybo Z7 board as shown below:

```
##Switches
set_property -dict { PACKAGE_PIN G15    IOSTANDARD LVCMOS33 } [get_ports { A[0] }];
#IO_L19N_T3_VREF_35 Sch=sw[0]
set_property -dict { PACKAGE_PIN P15    IOSTANDARD LVCMOS33 } [get_ports { A[1] }];
#IO_L24P_T3_34 Sch=sw[1]
set_property -dict { PACKAGE_PIN W13    IOSTANDARD LVCMOS33 } [get_ports { B[0] }];
#IO_L4N_T0_34 Sch=sw[2]
set_property -dict { PACKAGE_PIN T16    IOSTANDARD LVCMOS33 } [get_ports { B[1] }];
#IO_L9P_T1_DQS_34 Sch=sw[3]
##Buttons
set_property -dict { PACKAGE_PIN K18    IOSTANDARD LVCMOS33 } [get_ports { C_in }];
#IO_L12N_T1_MRCC_35 Sch=btn[0]
##LEDs
set_property -dict { PACKAGE_PIN M14    IOSTANDARD LVCMOS33 } [get_ports { Sum[0] }];
#IO_L23P_T3_35 Sch=led[0]
set_property -dict { PACKAGE_PIN M15    IOSTANDARD LVCMOS33 } [get_ports { Sum[1] }];
#IO_L23N_T3_35 Sch=led[1]
set_property -dict { PACKAGE_PIN G14    IOSTANDARD LVCMOS33 } [get_ports { C_out }];
#IO_0_35 Sch=led[2]
##RGB LED 6
set_property -dict { PACKAGE_PIN V16    IOSTANDARD LVCMOS33 } [get_ports { compare_result[0] }];
#IO_L18P_T2_34 Sch=led6_r
set_property -dict { PACKAGE_PIN F17    IOSTANDARD LVCMOS33 } [get_ports { compare_result[1] }];
#IO_L6N_T0_VREF_35 Sch=led6_g
set_property -dict { PACKAGE_PIN M17    IOSTANDARD LVCMOS33 } [get_ports { compare_result[2] }];
#IO_L8P_T1_AD10P_35 Sch=led6_b
```