VHDL CODE DEVELOPMENT AND CPLD BOARD OPERATING

Aim: The purpose is to create VHDL code, feed it to a CPLD board, and test the board's operation.

Summary: In order to test the functionality of the experiment, the following steps are taken:-

a) Create VHDL code for a 3-bit adder/subtractor in the structural modeling fashion utilizing a 1-bit full adder and m as the operation control signal.

b) Creating behavioral modeling-style VHDL code for a 3-bit ALU and loading it into a CPLD based on a 2-bit selector line executes the following actions/operations.

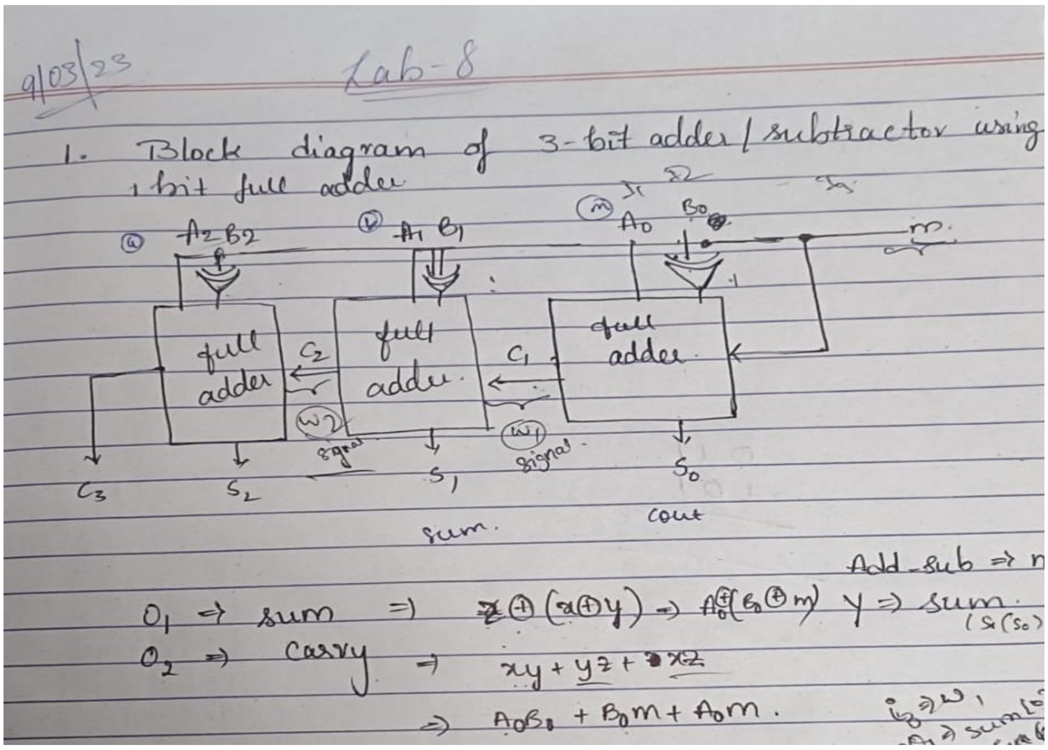
- For 0,0 =>A+B
- For 0,1 => A-B
- For 1,0 => A bitwise and B
- For 1,1 => A bitwise Xor B

c) Writing VHDL code in the behavioral modeling style for a 4:2 priority encoder with an active high enable pin and loading it into a CPLD board**.**

Components Used: CPLD Max3000A Board, JTAG port, type-B USB cable, Altera Bus Blaster cable, Bus Blaster HDL code (for CPLD)

Circuit Diagram:

1) 3-bit adder/subtractor



9/03/23

Lab-8

1. Block diagram of 3-bit adder / subtractor using 1 bit full adder

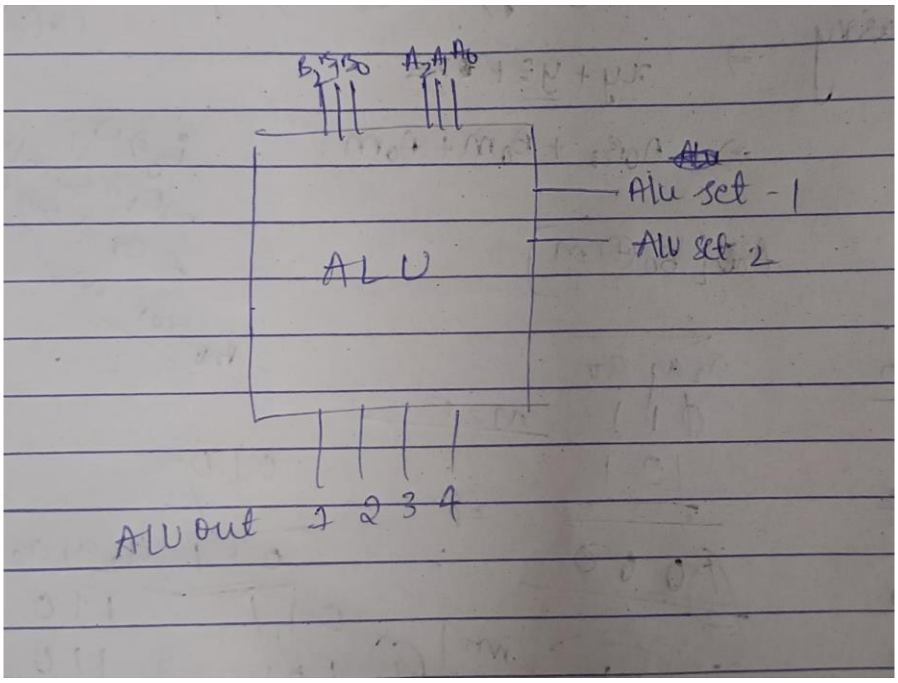
Add-sub ⇒ m

$O_1 \Rightarrow$ sum $\Rightarrow$ $z \oplus (x \oplus y) \Rightarrow A_0 \oplus (B_0 \oplus m)$ $y \Rightarrow$ sum (S(S₀)

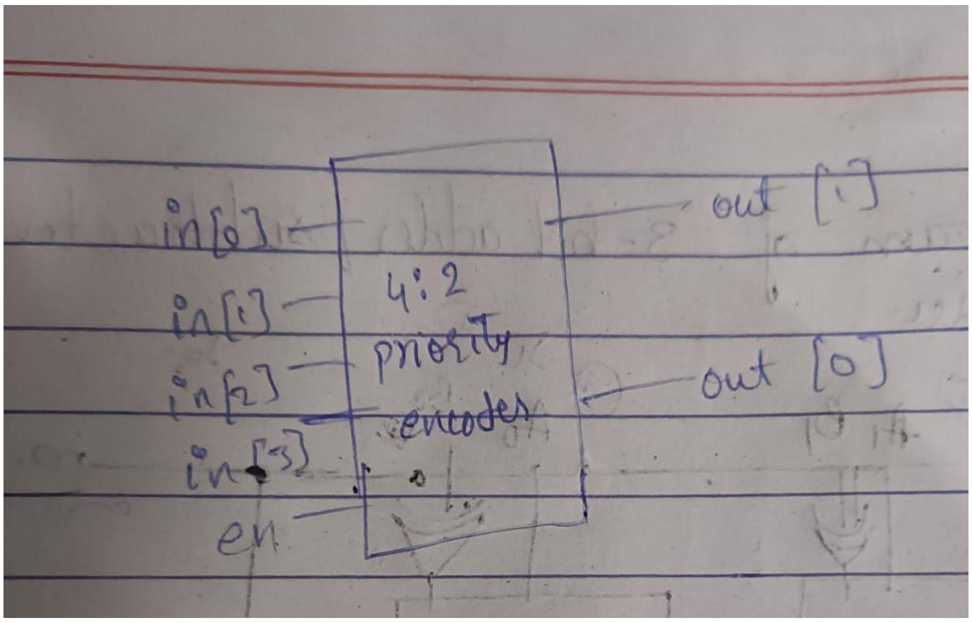
$O_2 \Rightarrow$ carry $\Rightarrow$ $xy + yz + xz$

$\Rightarrow A_0B_0 + B_0m + A_0m.$

2) 3-bit ALU

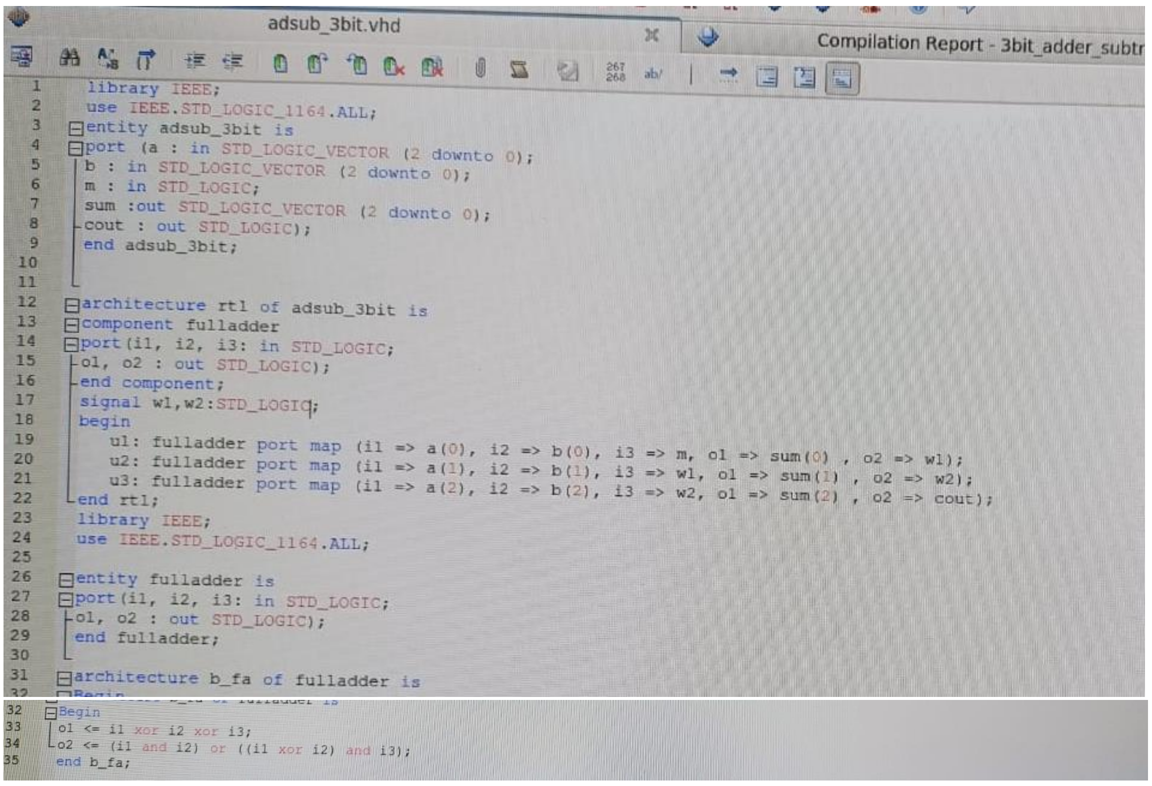


3) 4:2 priority encoder

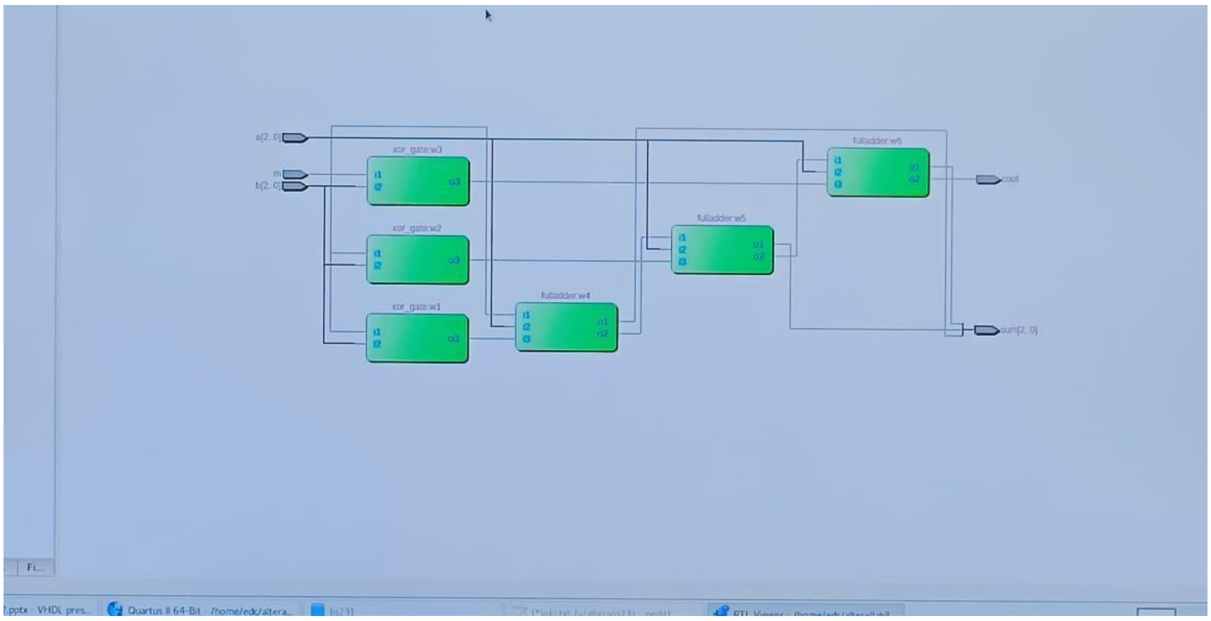


Snapshots of VHDL Code& their Gate-level Netlist:

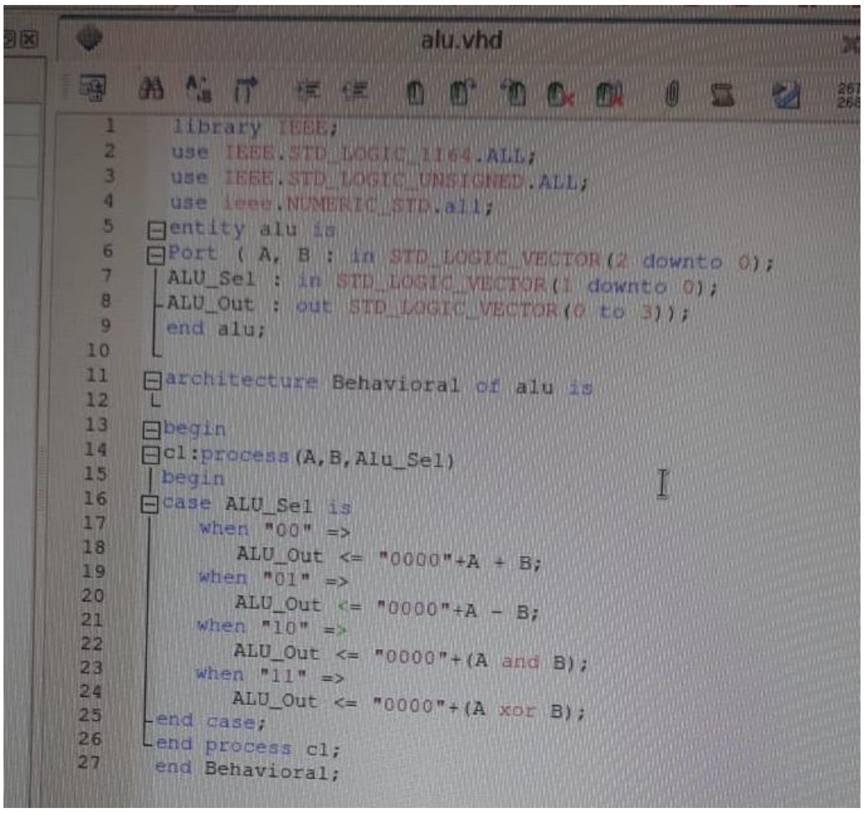
- 3-bit adder/subtractor in structural modeling style



```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity adsub_3bit is
port (a : in STD_LOGIC_VECTOR (2 downto 0);
b : in STD_LOGIC_VECTOR (2 downto 0);
m : in STD_LOGIC;
sum :out STD_LOGIC_VECTOR (2 downto 0);
cout : out STD_LOGIC);
end adsub_3bit;


architecture rtl of adsub_3bit is
component fulladder
port(i1, i2, i3: in STD_LOGIC;
o1, o2 : out STD_LOGIC);
end component;
signal w1,w2:STD_LOGIC;
begin
    u1: fulladder port map (i1 => a(0), i2 => b(0), i3 => m, o1 => sum(0) , o2 => w1);
    u2: fulladder port map (i1 => a(1), i2 => b(1), i3 => w1, o1 => sum(1) , o2 => w2);
    u3: fulladder port map (i1 => a(2), i2 => b(2), i3 => w2, o1 => sum(2) , o2 => cout);
end rtl;
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity fulladder is
port(i1, i2, i3: in STD_LOGIC;
o1, o2 : out STD_LOGIC);
end fulladder;

architecture b_fa of fulladder is
Begin
o1 <= i1 xor i2 xor i3;
o2 <= (i1 and i2) or ((i1 xor i2) and i3);
end b_fa;
```

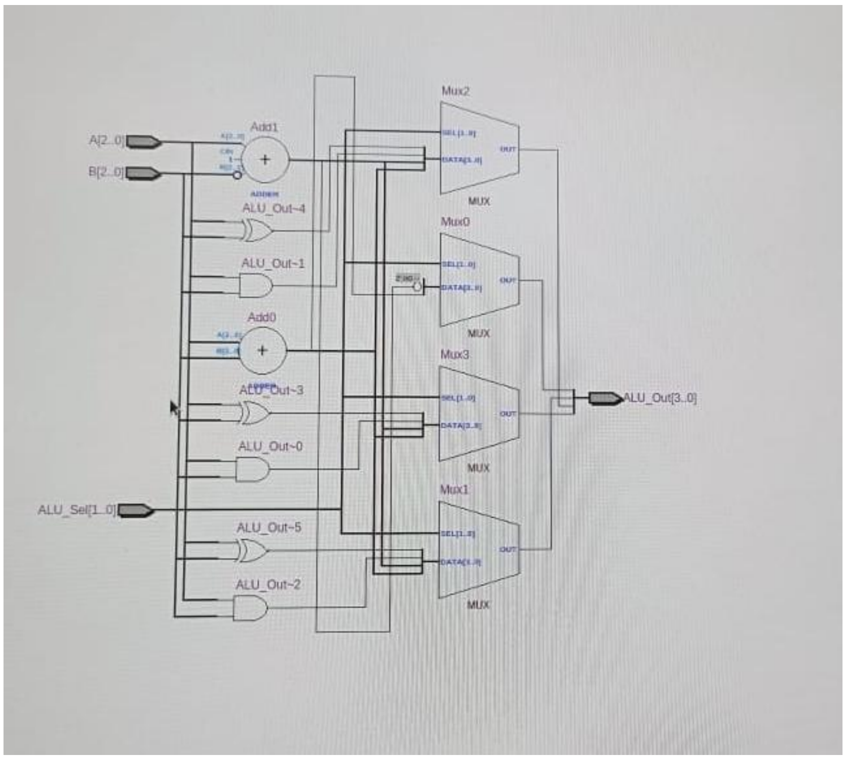


➢ 3-bit ALU in behavioral modeling style

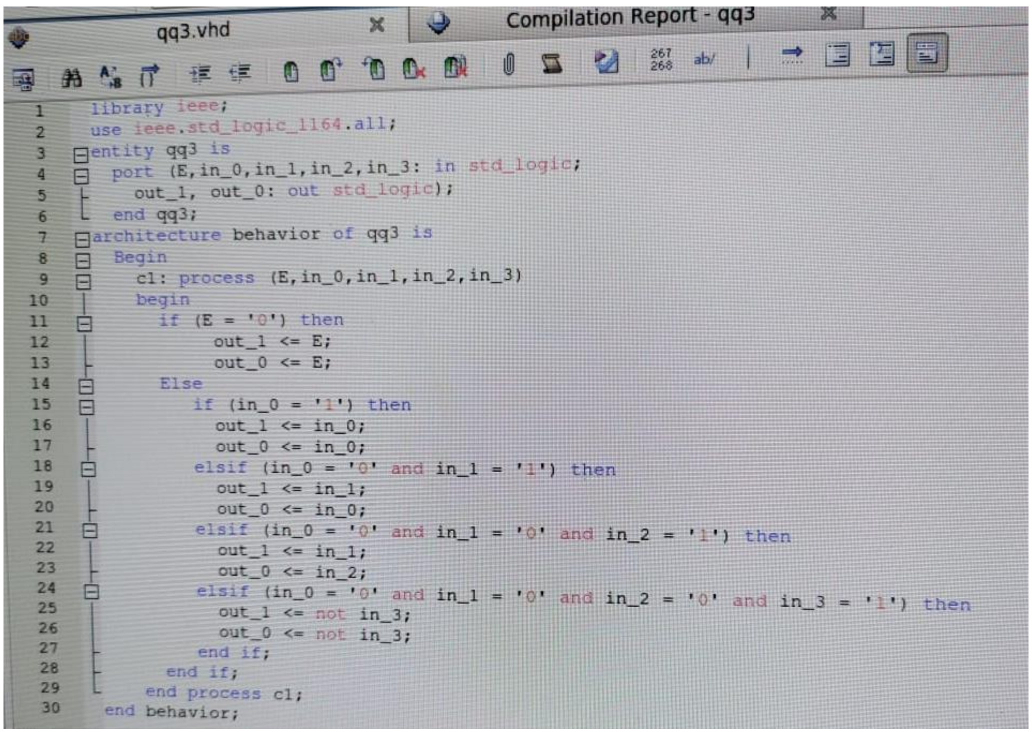


alu.vhd

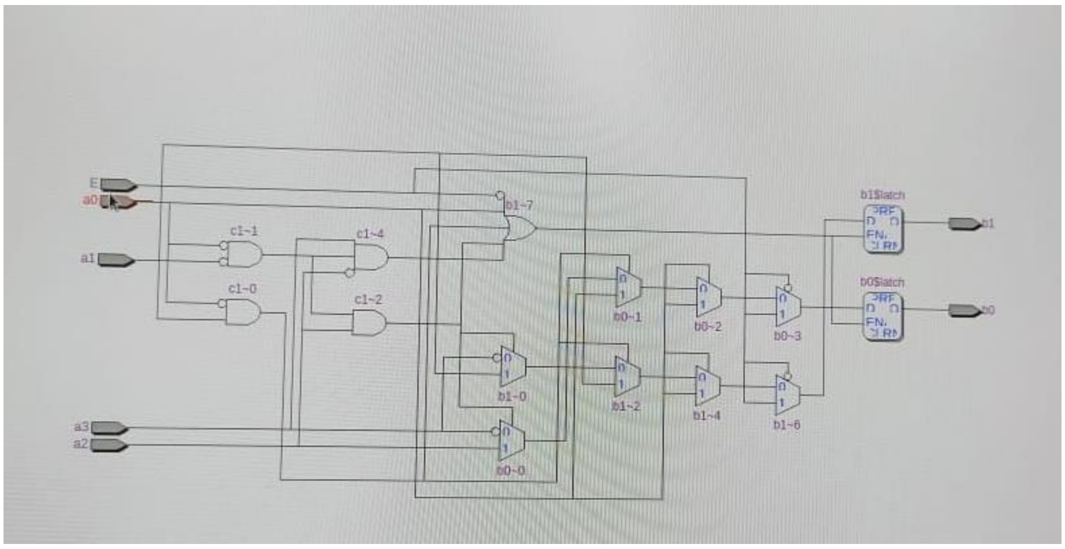
```vhdl
  library IEEE;
  use IEEE.STD_LOGIC_1164.ALL;
  use IEEE.STD_LOGIC_UNSIGNED.ALL;
  use ieee.NUMERIC_STD.all;
entity alu is
Port ( A, B : in STD_LOGIC_VECTOR(2 downto 0);
ALU_Sel : in STD_LOGIC_VECTOR(1 downto 0);
ALU_Out : out STD_LOGIC_VECTOR(0 to 3));
end alu;

architecture Behavioral of alu is

begin
cl:process(A,B,Alu_Sel)
begin
case ALU_Sel is
    when "00" =>
        ALU_Out <= "0000"+A + B;
    when "01" =>
        ALU_Out <= "0000"+A - B;
    when "10" =>
        ALU_Out <= "0000"+(A and B);
    when "11" =>
        ALU_Out <= "0000"+(A xor B);
end case;
end process cl;
end Behavioral;
```



➢ 4:2 priority encoder in behavioral modeling style

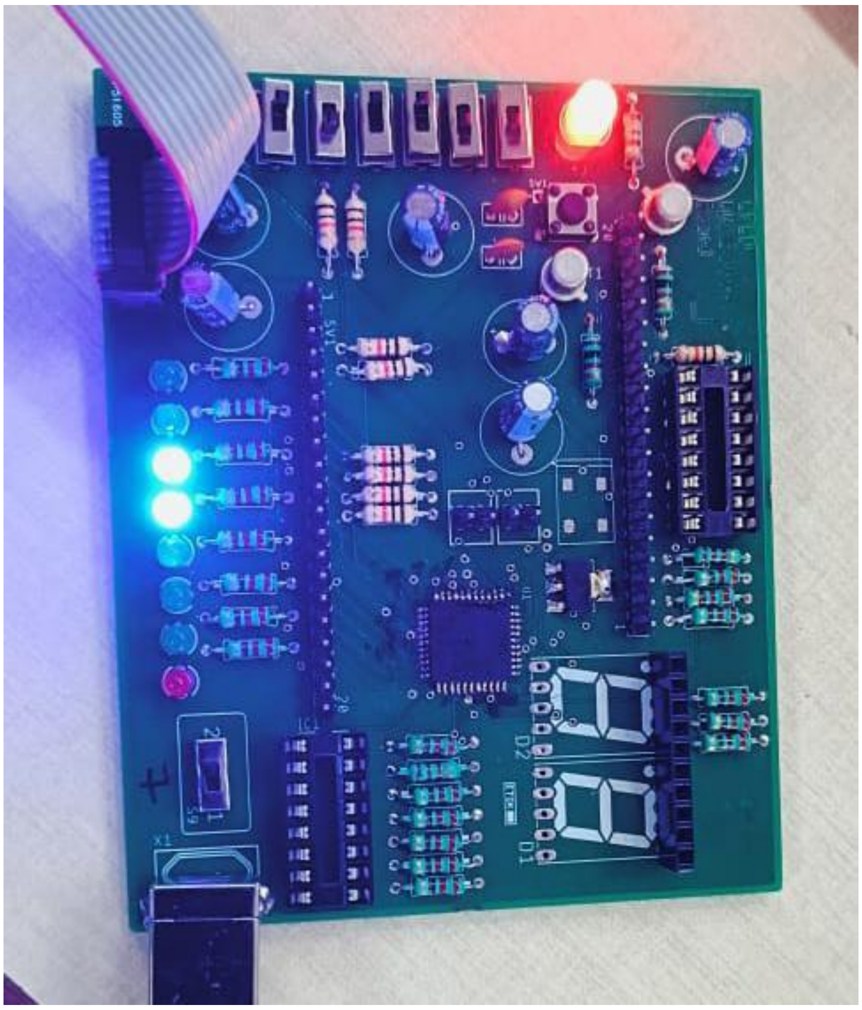


```
library ieee;
use ieee.std_logic_1164.all;
entity qq3 is
  port (E,in_0,in_1,in_2,in_3: in std_logic;
     out_1, out_0: out std_logic);
  end qq3;
architecture behavior of qq3 is
  Begin
     c1: process (E,in_0,in_1,in_2,in_3)
     begin
        if (E = '0') then
             out_1 <= E;
             out_0 <= E;
        Else
           if (in_0 = '1') then
             out_1 <= in_0;
             out_0 <= in_0;
           elsif (in_0 = '0' and in_1 = '1') then
             out_1 <= in_1;
             out_0 <= in_0;
           elsif (in_0 = '0' and in_1 = '0' and in_2 = '1') then
             out_1 <= in_1;
             out_0 <= in_2;
           elsif (in_0 = '0' and in_1 = '0' and in_2 = '0' and in_3 = '1') then
             out_1 <= not in_3;
             out_0 <= not in_3;
           end if;
        end if;
     end process c1;
  end behavior;
```

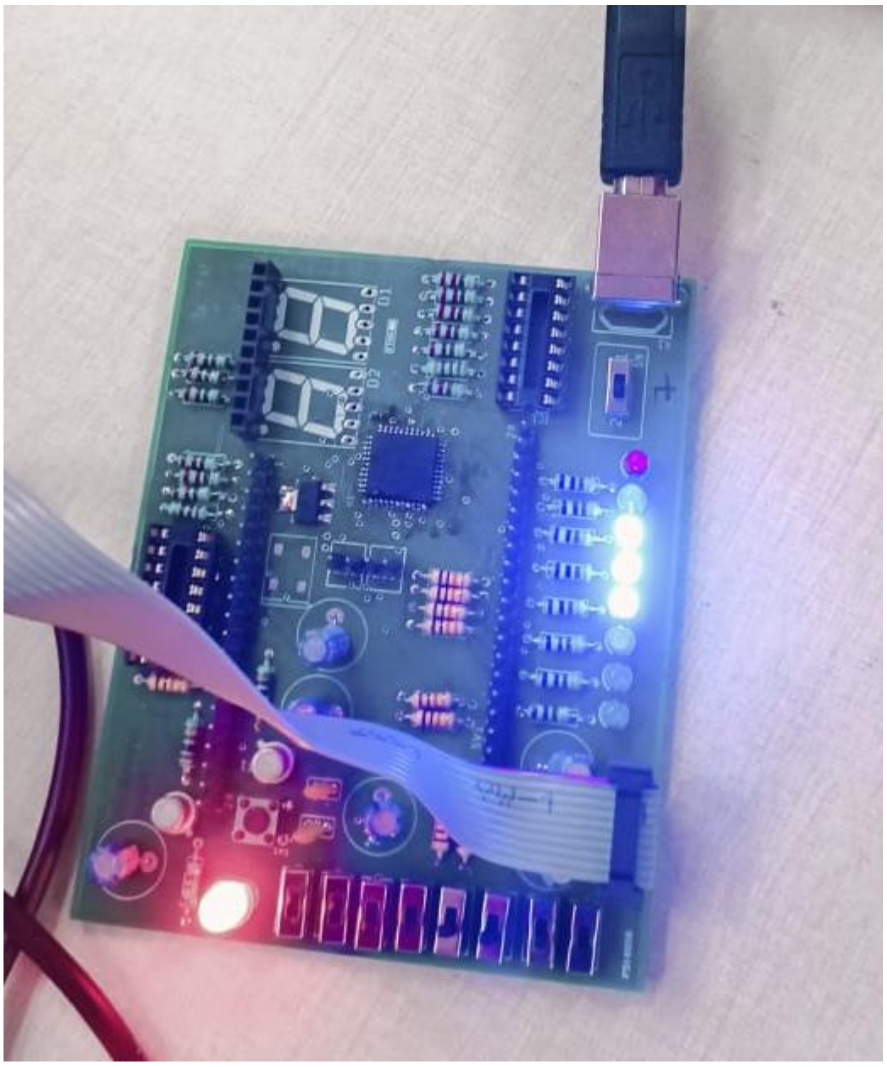


Snapshots of CPLD Board images:

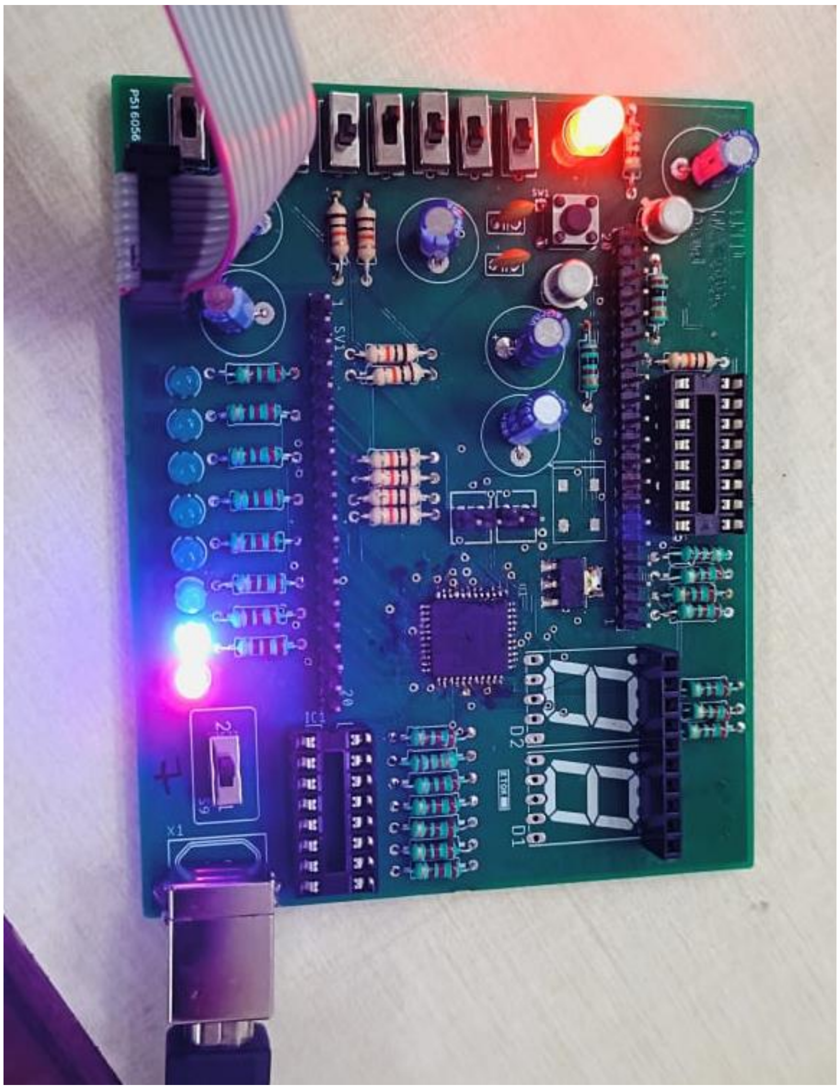
3-bit adder/subtractor



3-bit ALU



4:2 priority encoder



Results & Conclusion:

Recognized how to create VHDL code based on the circuit's logic. With this information, code was created for a 4:2 priority encoder, a 3-bit adder/subtractor, and a 3-bit ALU. The code's functioning was then validated by loading it onto a CPLD board.