CPLD MAX3000A

ALU and Priority Encoder

15th March, 2023

Implementing 3-bit adder/subtractor, ALU, 4:2 priority encoder in VHDL

Components used:

Quartus software,CPLD Board and USB connector

Summary of the Experiment:

Used VHDL to implement circuits and also used CPLD MAX3000A Board show the desired output.

Truth Table(s):

1)

Inputs			Outputs			
A	B	C	Sum	Carry	Difference	Borrow
0	0	0	0	0	0	0
0	0	1	1	0	1	1
0	1	0	1	0	1	1
0	1	1	0	1	0	1
1	0	0	1	0	1	0
1	0	1	0	1	0	0
1	1	0	0	1	0	0
1	1	1	1	1	1	1

Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;--define the entity with ports
 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 structure of adsub_3bit is
component fulladder
port(i1, i2, i3: in std_logic;
o1, o2: out std_logic);
end component;
signal C0, C1: std_logic;
signal b0, b1, b2: std_logic;
begin
b0 <= b(0) xor m;
b1 <= b(1) xor m;
b2 <= b(2) xor m;
u1: fulladder port map (i1 => a(0), i2 => b0, i3 => m, o1 => sum(0), o2 =>
C0);
u2: fulladder port map (i1 => a(1), i2 => b1, i3 => C0, o1 => sum(1), o2 => C1); u:
fulladder port map (i1 => a(2), i2 => b2, i3 => C1, o1 => sum(2), o2 => cout);  end
structure;
--VHDL CODE FOR single bit full adder --in same VHDL file-----
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
 entity fulladder is
port(i1, i2, i3: in bit; o1, o2 : out bit);
end fulladder;
--Defining the architecture of full adder in dataflow modelling style---
architecture dataflow of fulladder is
begin
 o1 <= i1 xor i2 xor i3;
 o2 <= (i1 and i2) or ((i1 xor i2) and i3);
end dataflow;
```

2)

Code:

```
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(3 downto 0); -- 2 inputs 8-bit
ALU_Sel : in STD_LOGIC_VECTOR(1 downto 0);
 ALU_Out : out STD_LOGIC_VECTOR(3 downto 0));
 end ALU;
architecture Behavioral of ALU is
begin
c1: process(A,B,ALU_Sel)
begin
if(ALU_Sel(0)='0') then
      if(ALU_Sel(1)='0') then
           ALU_Out<=A+B;
      else ALU_Out<=A-B;
      end if;
else if(ALU_Sel(1)='0') then
     Alu_Out(0)<=A(0) and B(0);
     Alu_Out(1)<=A(1) and B(1);
      Alu_Out(2)<=A(2) and B(2); Alu_Out(3)<='0';
 else Alu_Out(0)<=A(0) xor B(0);
       Alu_Out(1)<=A(1) xor B(1);
        Alu_Out(2)<=A(2) xor B(2);
      Alu_Out(3)<='0';
end if;
end if;
```

```
end process c1;
end Behavioral;
```

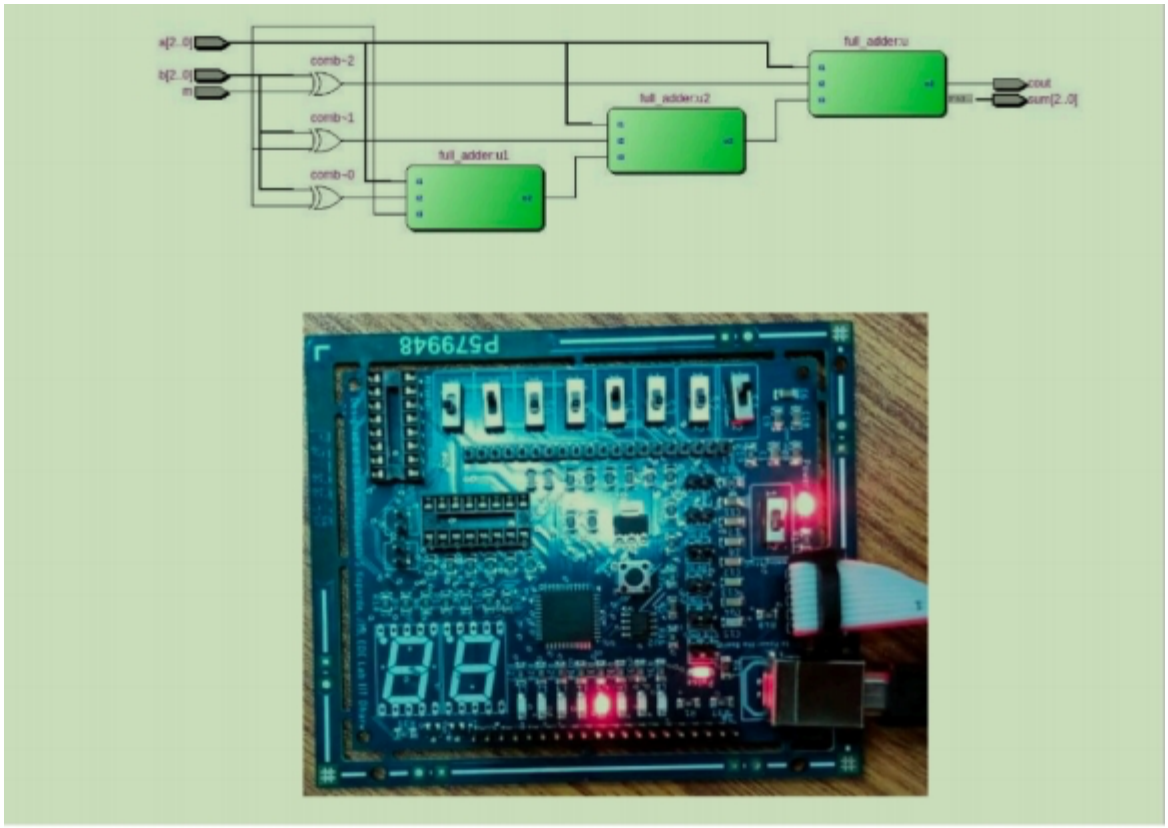
Q3)

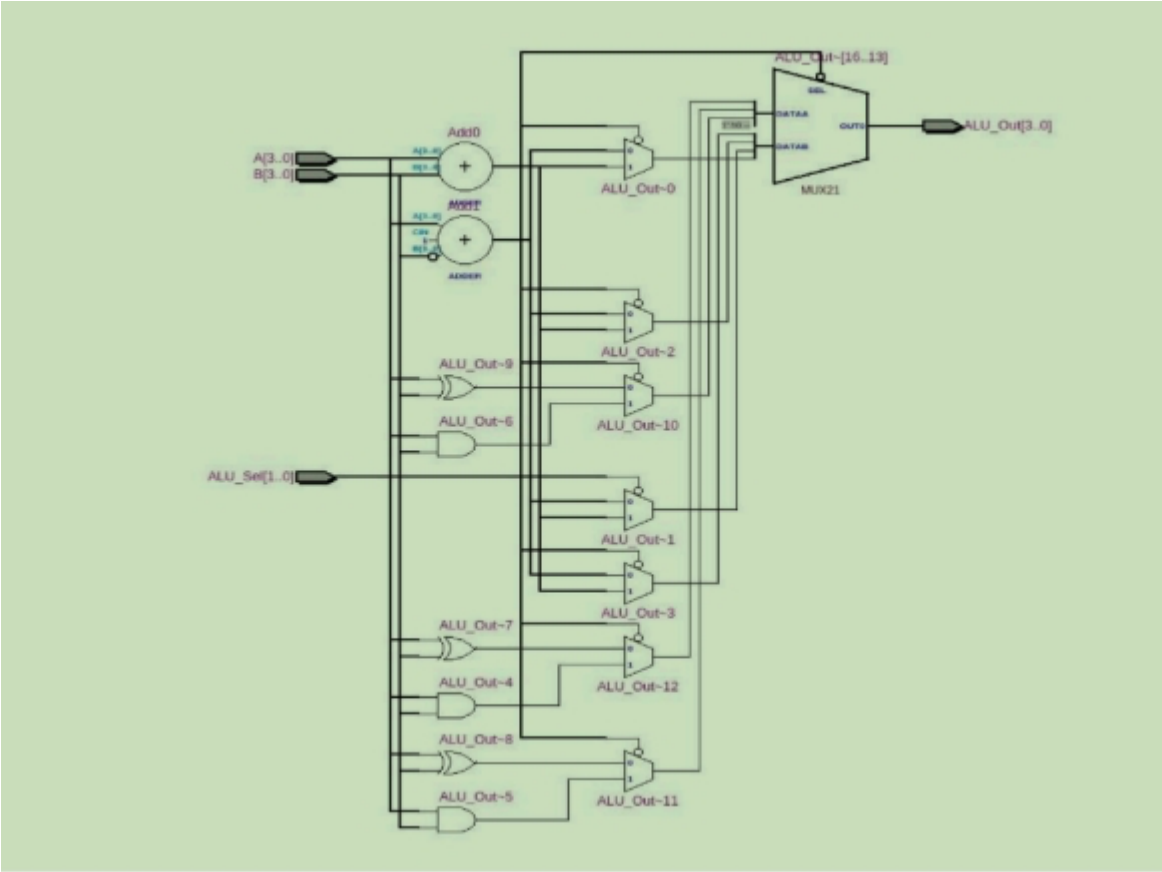
Code:

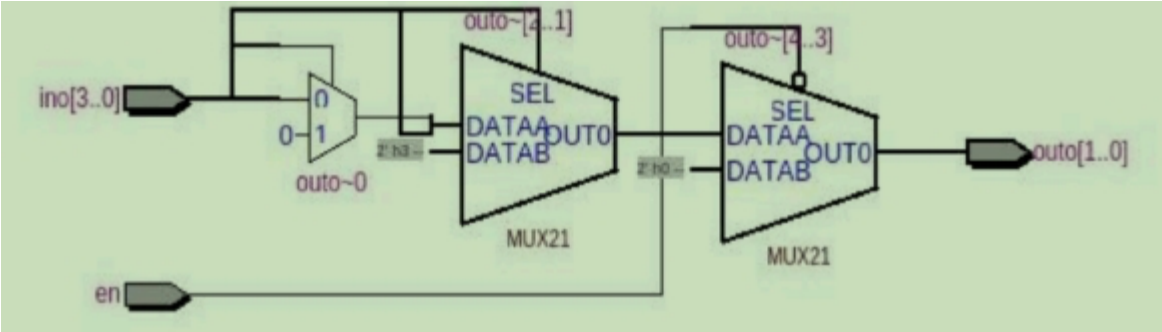
```
library IEEE;
 use IEEE.STD_LOGIC_1164.all;
 entity p_enc is
port (ino : in STD_LOGIC_VECTOR (3 downto 0);
 en : in STD_LOGIC;
outo :out STD_LOGIC_VECTOR (1 downto 0));
 end p_enc;
 architecture behavior of p_enc is
begin
c1: process (ino, en)
begin
  if(en='0') then
     outo(0)<='0';
      outo(1)<='0';
else if(ino(0)='1') then
      outo(0)<='1';
       outo(1)<='1';
else if(ino(1)='1') then
      outo(0)<='0';
     outo(1)<='1';
 else if(ino(2)='1') then
    outo(0)<='1';
   outo(1)<='0';
```

```
else
  outo(0)<='0';
   outo(1)<='0';
      end if;
    end if;
  end if;
end if;
end process c1;
 end behavior;
```

Circuit Snapshots:







Conclusion:

The constructed CPLD simulations show the same outputs as theoretically expected from the combination of those circuits.

Thank You.