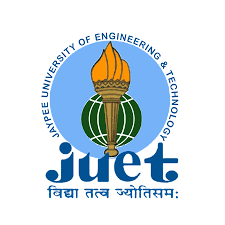
**JAYPEE UNIVERSITY OF ENGINEERING &TECHNOLOGY, GUNA**

**DEPARTMENT OF COMPUTER SCIENCE &ENGINEERING**



**A PRACTICAL WORK BOOK**

**of**

**COMPUTER ORGANIZATION & ARCHITECTURE LAB**

**(COURSE CODE: 18B17CI474)**

**SUBMITTED**

**TO**

**Dr. Rahul Pachauri**

**Name of student-** ViradChaurasia**ENRL NO.**201B354

**Class-** 3rd Year ( 6th Sem)**BATCH –**B12

**BRANCH-** CSE**SESSION--**2023

***INDEX***

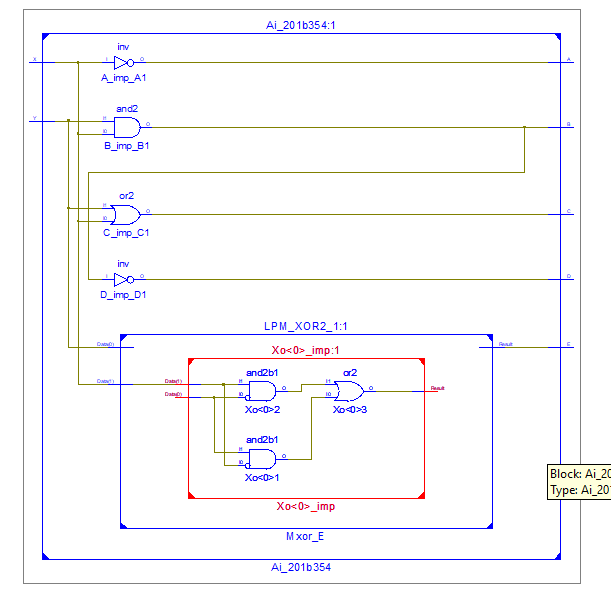
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S. No.** | **Aim of the experiment** | **Date of submission** | **signature** | **remarks** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Experiment -1**

**Aim: Design of All-in-One logic gate circuits.**

**Exercise#1:** Design two inputs and five outputs All-in-One logic gate circuit shown in Fig.1. Write theVHDL code in data flow style of modeling.

**Design Code—**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Ai\_201b354 is

Port ( X : in STD\_LOGIC;

Y : in STD\_LOGIC;

A : out STD\_LOGIC;

B : out STD\_LOGIC;

C : out STD\_LOGIC;

D : out STD\_LOGIC;

E : out STD\_LOGIC);

end Ai\_201b354;

architecture dataflow of Ai\_201b354 is

begin

A <= not X;

B <= X and Y;

C <= X or Y;

D <= X nand Y;

E <= X xor Y;

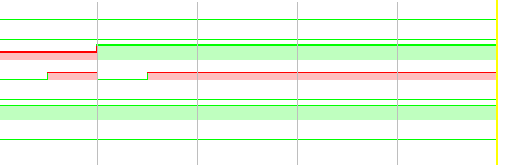
end dataflow;

**Test Bench Code-**

stim\_proc: process

begin

a<='0' ; b<='0';

 wait for 100 ns;

a<='0' ; b<='1'; A

wait for 100 ns;

a<='1' ; b<='0';

wait for 100 ns;

a<='1' ; b<='1';

wait for 100 ns;

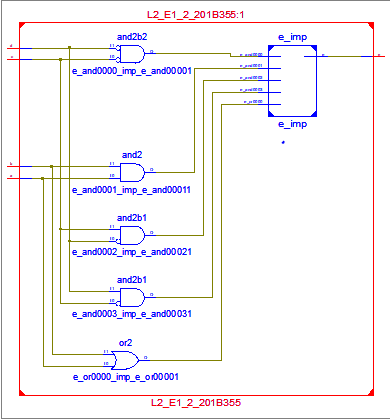
wait;

end process;

**Exercise#2:** Design two inputs and one output All-in-One logic gate diagram shown in Fig.2. Write the VHDL code in behavioral style of modeling using (i) if-then-else (ii) case-when.

**Design Code--**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Exper2\_354 is

Port ( a : in STD\_LOGIC;

b : in STD\_LOGIC;

c : in STD\_LOGIC;

d : in STD\_LOGIC;

x : out STD\_LOGIC);

end Exper2\_354;

architecture Behavioral of Exper2\_354 is

begin

process(a,b,c,d)

begin

if ( c='0' and d='0' )

then

x<= a and b;

elsif( c='1' and d='0' )

then

x<= a or b;

elsif( c='0' and d='1' )

then

x<= a nor b;

else

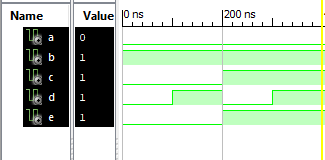
x<= a nand b;

end if;

end process;

end Behavioral;

**Test Bench Code-**



begin

a<='0';b<='1';

c<='0';d<='0';

wait for 100 ns;

c<='0'; d<='1';

wait for 100ns;

c<='1'; d<='0';

wait for 100 ns;

c<='1' ; d<='1';

wait for 100 ns;

wait;

end process;

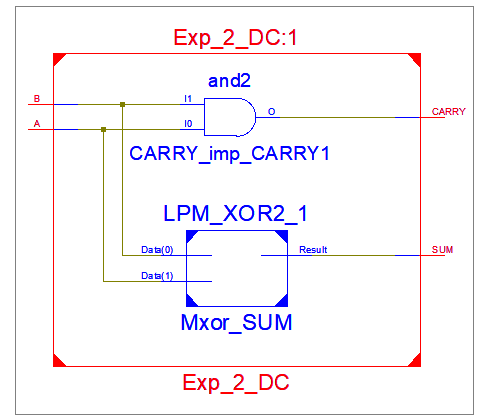
**Experiment # 2**

**Aim: Design of basic binary adders and subtractors.**

**Exercise#1:** Design half adder in **data flow style** of modeling.

**Design Code--**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Exp\_2\_DC is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

SUM : out STD\_LOGIC;

CARRY : out STD\_LOGIC);

end Exp\_2\_DC;

architecture DataFlow of Exp\_2\_DC is

begin

SUM <= A xor B;

CARRY <= A and B;

end DataFlow;

**Test Bench Code---**

begin

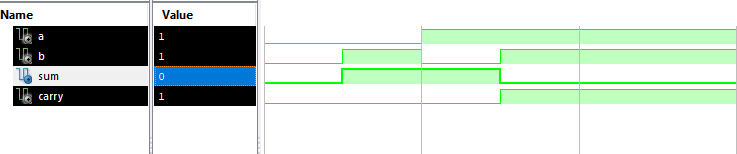
A<='0' ; B<= '0';

wait for 100 ns;

A<='0' ; B<= '1';

wait for 100 ns;

A<='1' ; B<= '0';

 wait for 100 ns;

A<='1' ; B<= '1';

wait for 100 ns;

wait;

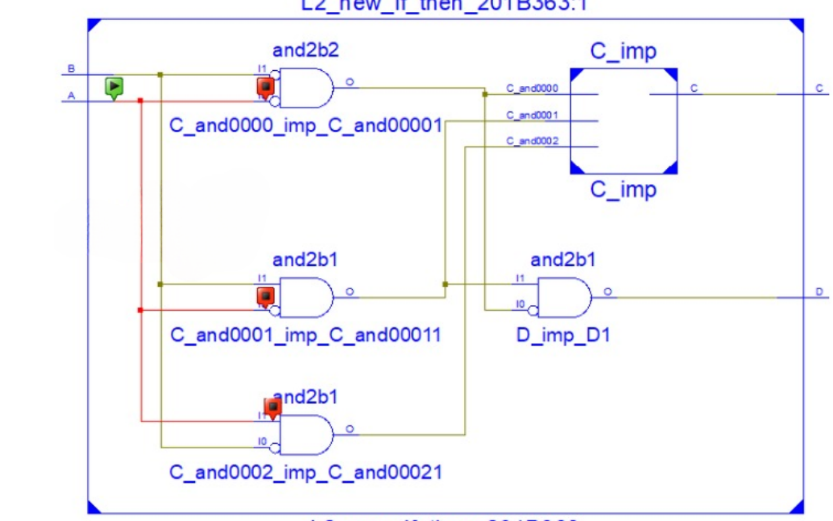
end process;

END;

**Exercise#2:** Design half subtractor in **behavioral style** (using either **if-then** or **case when**) of

modeling.

Design Code—

entity HS\_201B354 is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : out STD\_LOGIC;

D : out STD\_LOGIC);

end HS\_201B354;

architecture Behavioral of HS\_201B354is

begin

process(a,b)

begin

if(A='0' and B='0')then

C<='0';

D<='0';

elsif(A='0' and B='1')then

C<='1';

D<='1';

elsif(A='1' and B='0')then

C<='1';

D<='0';

else

C<='0';

D<='0';

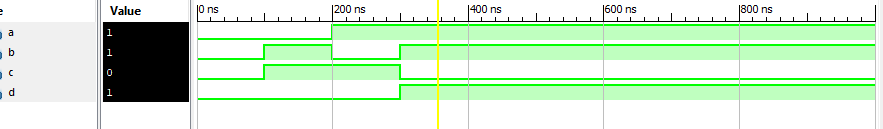
end if;

end process;

**Test Bench Code:-**

begin

A<='0';B<='0';

 wait for 100 ns;

A<='0';B<='1';

wait for 100 ns;

A<='1';B<='0';

wait for 100 ns;

A<='1';B<='1';

wait for 100 ns;

**Experiment # 3**

**Aim**: Design of 4-bit adder-subtractor circuits**.**

**Exercise#1:** Design 4-bit ripple carry adder shown in Fig.1 using (i) structural style of architecture

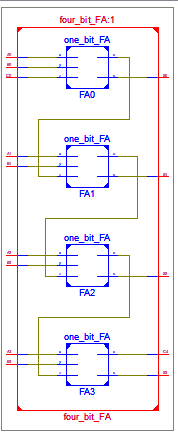
(ii) **generic** and **for-generate** statements instructural style of architecture

**Design Code:**

**Component Code:**

library IEEE; use IEEE.STD\_LOGIC\_1164.ALL;

entity one\_bit\_FA is

Port ( x : in STD\_LOGIC; y : in STD\_LOGIC;

z : in STD\_LOGIC; s : out STD\_LOGIC;

c : out STD\_LOGIC); end one\_bit\_FA;

architecture DataFlow of one\_bit\_FA is

begin

s<=x xor y xor z;

c<=(x and y ) or (x and z) or ( y and z);

end DataFlow;

**Structural Code:-**

entity four\_bit\_FA is Port ( A0 : in STD\_LOGIC;

B0 : in STD\_LOGIC; C0 : in STD\_LOGIC;

A1 : in STD\_LOGIC; S0 : out STD\_LOGIC;

B1 : in STD\_LOGIC; S1 : out STD\_LOGIC;

A2 : in STD\_LOGIC; S2 : out STD\_LOGIC;

B2 : in STD\_LOGIC; S3 : out STD\_LOGIC;

A3 : in STD\_LOGIC; C4 : out STD\_LOGIC;

B3 : in STD\_LOGIC; )

end four\_bit\_FA;

architecture Structural of four\_bit\_FA is

signal C1,C2,C3:std\_logic;

component one\_bit\_FA

Port ( x : in STD\_LOGIC; y : in STD\_LOGIC;

z : in STD\_LOGIC; s : out STD\_LOGIC; c : out STD\_LOGIC);

end component;

begin

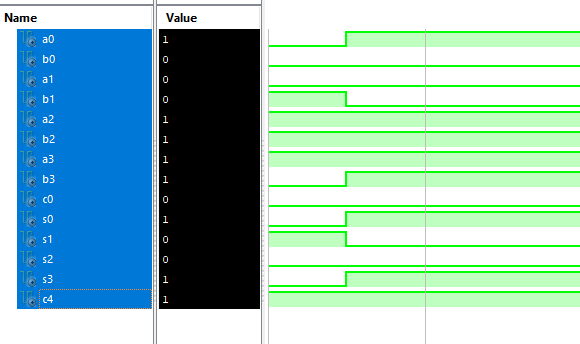
FA0: one\_bit\_FA port map(A0,B0,C0,S0,C1);

FA1: one\_bit\_FA port map(A1,B1,C1,S1,C2);

FA2: one\_bit\_FA port map(A2,B2,C2,S2,C3);

FA3: one\_bit\_FA port map(A3,B3,C3,S3,C4);

end Structural**;**

**Test Bench Code:**

stim\_proc: process

begin

C0<='0';

A3<='1'; A2<='1'; A1<='0'; A0<='0';

B3<='0'; B2<='1'; B1<='1'; B0<='0';

wait for 100 ns;

A3<='1'; A2<='1'; A1<='0'; A0<='1';

B3<='1'; B2<='1'; B1<='0'; B0<='0';

wait for 100 ns;

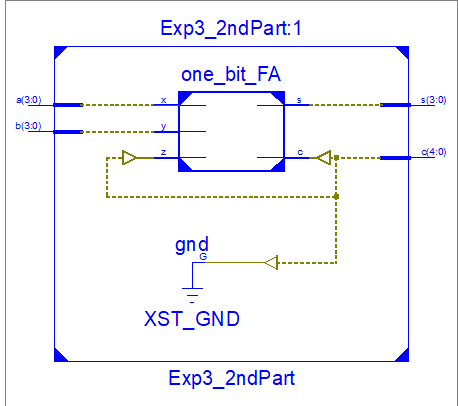
wait;

end process;

END;

(ii) **generic** and **for-generate** statements instructural style of architecture

**Design Code:**

entity Exp3\_2ndPart is

generic (N: integer := 4);

Port ( a : in STD\_LOGIC\_VECTOR (N-1 downto 0);

b : in STD\_LOGIC\_VECTOR (N-1 downto 0);

c :inout STD\_LOGIC\_VECTOR (N downto 0);

s : out STD\_LOGIC\_VECTOR (N-1 downto 0));

end Exp3\_2ndPart;

architecture Structural of Exp3\_2ndPart is

component one\_bit\_FA

Port ( x : in STD\_LOGIC;

y : in STD\_LOGIC;

z : in STD\_LOGIC;

s : out STD\_LOGIC;

c : out STD\_LOGIC);end component;

**begin**

**c(0)<='0';**

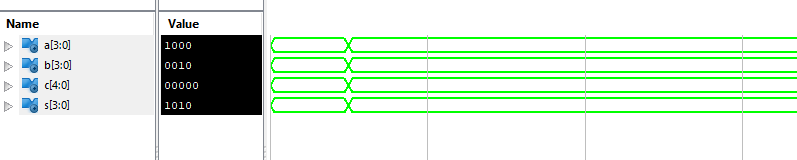
**F: for i in 0 to N-1 generate**

**FA: one\_bit\_FA port map (A(i), B(i), C(i), S(i), C(i+1));**

**end generate;**

**end Structural;**

**TestBench:**

**stim\_proc: process**

**begin**

**A<="1100";**

**B<="0110";**

**wait for 100 ns;**

**A<="1000";**

**B<="0010";**

**wait for 100 ns;**

**wait;**

**end process;**

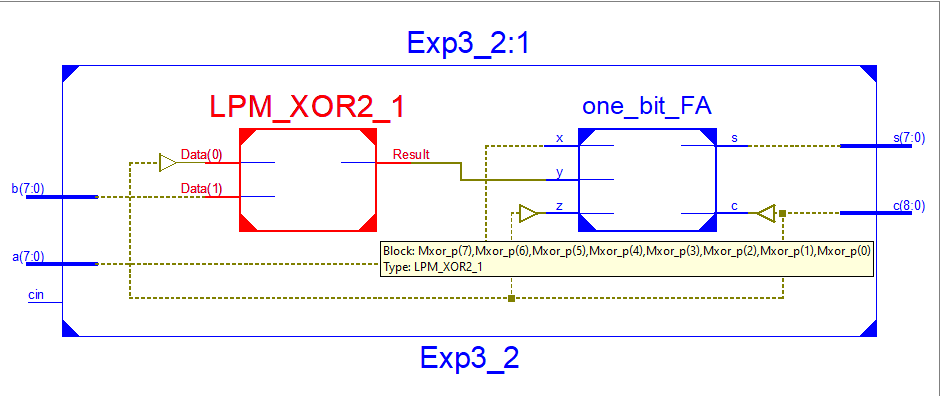
**Exercise#2: Design 4-bit adder-subtractor shown in Fig.2 using structural style of architecture.**

**Design Code:**

entity Exp3\_2 is

generic (N: integer := 8);

Port ( a : in STD\_LOGIC\_VECTOR (N-1 downto 0);

b : in STD\_LOGIC\_VECTOR (N-1 downto 0);

c :inout STD\_LOGIC\_VECTOR (N downto 0);

s : out STD\_LOGIC\_VECTOR (N-1 downto 0);

cin : in STD\_LOGIC);

end Exp3\_2;

architecture Structural of Exp3\_2 is

component one\_bit\_FA

Port ( x : in STD\_LOGIC;

y : in STD\_LOGIC;

z : in STD\_LOGIC;

s : out STD\_LOGIC;

c : out STD\_LOGIC);

end component;

signal p: STD\_LOGIC\_VECTOR (N-1 downto 0);

**begin**

**c(0)<= cin;**

**F: for i in 0 to N-1 generate**

**p(i)<=B(i) xorc(0);**

**FA: one\_bit\_FA port map (A(i), p(i), C(i), S(i), C(i+1));**

**end generate;**

**end Structural;**

**Test Bench:**

**stim\_proc: process**

**begin**

**cin<=’0’;**

** A<="1100";**

**B<="0110";**

**wait for 100 ns;**

**cin<=’1’;**

**A<="1000";**

**B<="0010";**

**wait for 100 ns;**

**wait;**

**end process;**

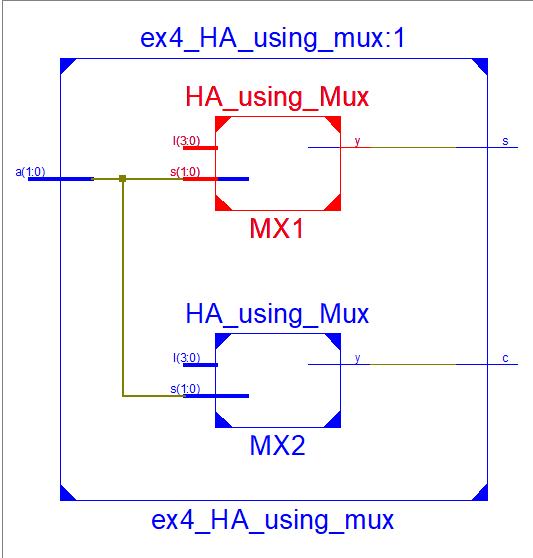
**Experiment # 4**

**Aim: Design of combinational logic circuits.**

**Exercise#1: Design a half adder using two 4:1 multiplexers (shown in Fig.1) in structural style of architecture**

**Design**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity ex4\_HA\_using\_mux is

Port ( a : in STD\_LOGIC\_VECTOR (1 downto 0);

s : out STD\_LOGIC;

c : out STD\_LOGIC);

end ex4\_HA\_using\_mux;

architecture Structural of ex4\_HA\_using\_mux is

component HA\_using\_Mux is

Port ( I : in STD\_LOGIC\_VECTOR (3 downto 0);

s : in STD\_LOGIC\_VECTOR (1 downto 0);

y : out STD\_LOGIC);

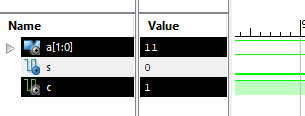
end component;

begin

MX1: HA\_using\_Mux port map("0110" ,a,s);

MX2:HA\_using\_Mux port map("0001",a,c);

End Structural;

**Test** **bench :**

begin

a<="00";

wait for 100 ns;

a<="01";

wait for 100 ns;

a<="10";

wait for 100 ns;

a<="11";

wait for 100 ns;

wait;

end process;

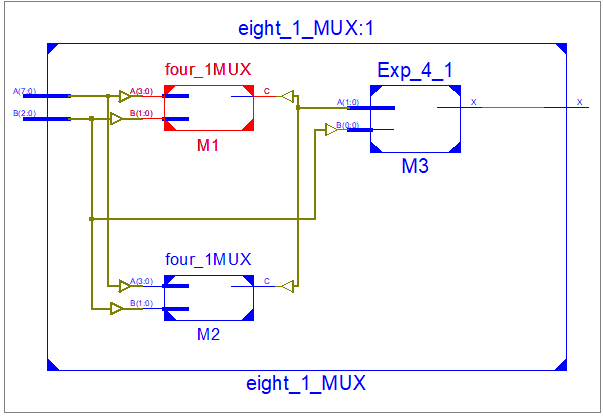
**Exercise#2:** Design 8:1 multiplexer using two 4:1 and one 2:1 multiplexers (shown in Fig.2) in

**structural style of architecture**. Use inputs and selection lines in the form of bus.

**Design Code :**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity eight\_1\_MUX is

Port ( A : in STD\_LOGIC\_VECTOR (7 downto 0);

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

X : out STD\_LOGIC);

end eight\_1\_MUX;

architecture structural of eight\_1\_MUX is

signal m:STD\_LOGIC\_VECTOR(1 downto 0);

component four\_1MUX is

Port ( A : in STD\_LOGIC\_VECTOR (3 downto 0);

B : in STD\_LOGIC\_VECTOR (1 downto 0);

C : out STD\_LOGIC);

end component;

component Exp\_4\_1 is

Port ( A : in STD\_LOGIC\_VECTOR (1 downto 0);

B : in STD\_LOGIC\_VECTOR (0 downto 0);

X : out STD\_LOGIC);

end component;

begin

M1 :four\_1MUX port map(A( 3 downto 0) ,B(1 downto 0),m(0));

M2 :four\_1MUX port map(A( 7 downto 4) ,B(1 downto 0),m(1));

M3 :Exp\_4\_1 port map(m,B(2 downto 2),X);

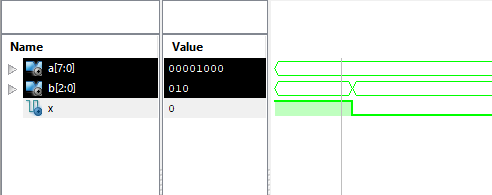
end structural;

**Test Bench code**

stim\_proc: process

begin

-- hold reset state for 100 ns.

 A<= "00001000" ; B<= "011";

wait for 100 ns;

A<= "00001000" ; B<= "010";

wait for 100 ns;

wait;

end process;

END;