**Experiment no. # 01**

**AIM:** Introduction to VHDL and Xilinx ISE Software.

VHDL: It is a description language that describes the digital circuits by their functions, data flow behaviour and structures. The VHDL stands for VHSIC (Very High Speed Integrated Circuits)Hardware Description Language. This means VHDL can be used to accelerate the design process of digital circuits. It is very important to note that VHDL is NOT a programming language but it is an HDL(Hardware Description Language) which describes asynchronous, synchronous, combinational and sequential digital circuits. This language is used to build a prototype to discover complexity, problems and faults in the design before actually implementing it in hardware. This hardware description is used to configure a Programmable Logic Device (PLD), such as a Field Programmable Gate Array (FPGA), with a custom logic design.

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

VHDL coding for the design:

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity haffadder is

Port ( a,b : in STD\_LOGIC;

c,s : out STD\_LOGIC);

end haffadder;

architecture Dataflow of haffadder is

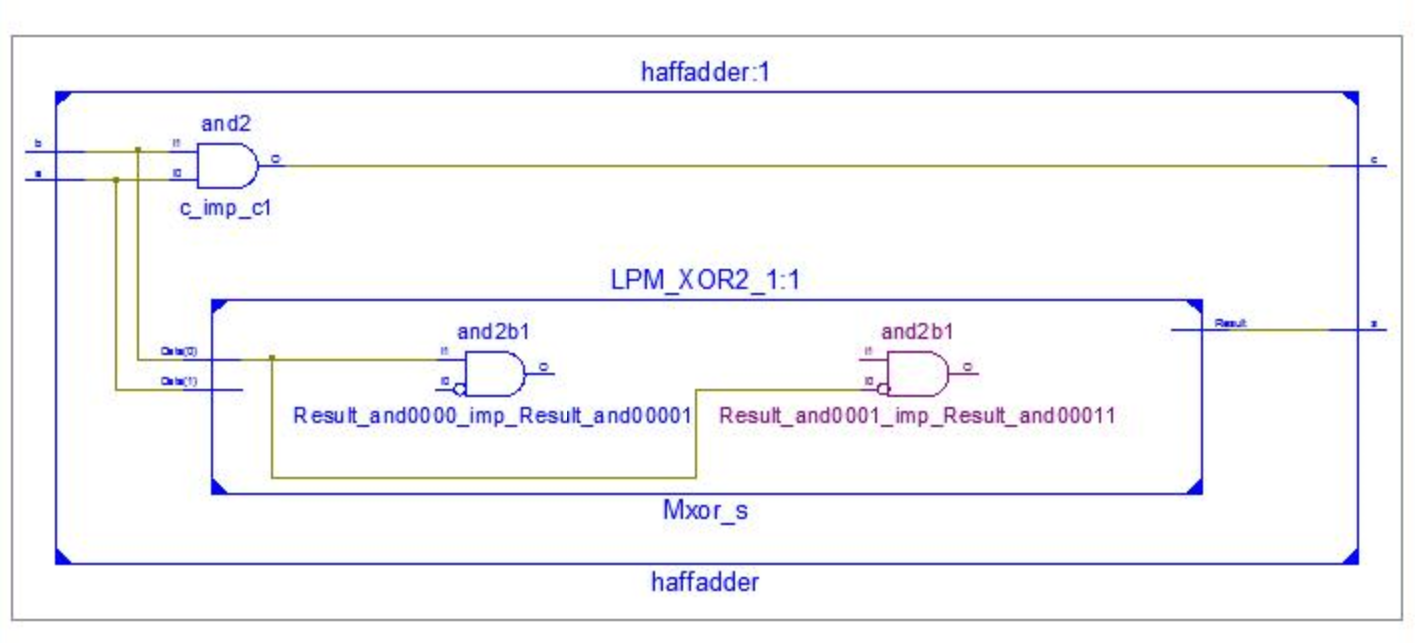
begin

s<=a xor b;

c<=a and b;

end Dataflow;

Register Transfer Logic (RTL) schematic of the design**:**



Test bench coding:

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY test\_haffadder IS

END test\_haffadder;

ARCHITECTURE Datflow OF test\_haffadder IS

COMPONENT haffadder

PORT(

a : IN std\_logic;

b : IN std\_logic;

c : OUT std\_logic;

s : OUT std\_logic

);

END COMPONENT;

signal a : std\_logic := '0';

signal b : std\_logic := '0';

signal c : std\_logic;

signal s : std\_logic ;

BEGIN

uut: haffadder PORT MAP (

a => a,

b => b,

c => c,

s => s

);

stim\_proc: process

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';

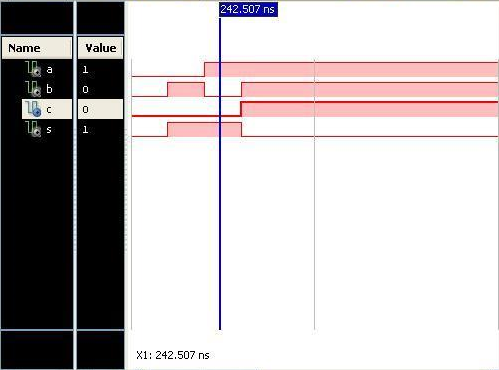
wait for 100 ns;

wait;

end process;

END;

Synthesis of the design to see logic resources and timing detail:



From the above figure we can clearly see that at 242.507 ns ,input values are a=1 ,b=0 and output values are c=0,s=1.

**Exercise #2:** Design full adder in data flow style of modelling.

VHDL coding for the design:

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity fulladdre171 is

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

s,c : out STD\_LOGIC);

end fulladdre171;

architecture Dataflow of fulladdre171 is

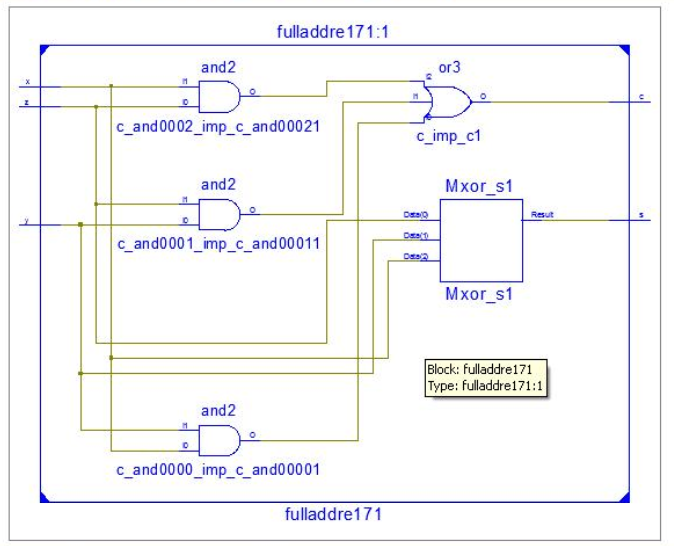
begin

s <= x xor y xor z;

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

end Dataflow;

Register Transfer Logic (RTL) schematic of the design**:**



Test bench coding:

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY test\_fulladder171 IS

END test\_fulladder171;

ARCHITECTURE Dataflow OF test\_fulladder171 IS

COMPONENT fulladdre171

PORT(

x : IN std\_logic;

y : IN std\_logic;

z : IN std\_logic;

s : OUT std\_logic;

c : OUT std\_logic

);

END COMPONENT;

signal x : std\_logic := '0';

signal y : std\_logic := '0';

signal z : std\_logic := '0';

signal s : std\_logic;

signal c : std\_logic;

BEGIN

uut: fulladdre171 PORT MAP (

x => x,

y => y,

z => z,

s => s,

c => c

);

stim\_proc: process

begin

x<='0';

y<='0';

z<='0';

wait for 100 ns;

x<='0';

y<='1';

z<='0';

wait for 100 ns;

x<='1';

y<='0';

z<='0';

wait for 100 ns;

x<='1';

y<='1';

z<='0';

wait for 100 ns;

x<='0';

y<='0';

z<='1';

wait for 100 ns;

x<='0';

y<='1';

z<='1';

wait for 100 ns;

x<='1';

y<='0';

z<='1';

wait for 100 ns;

x<='1';

y<='1';

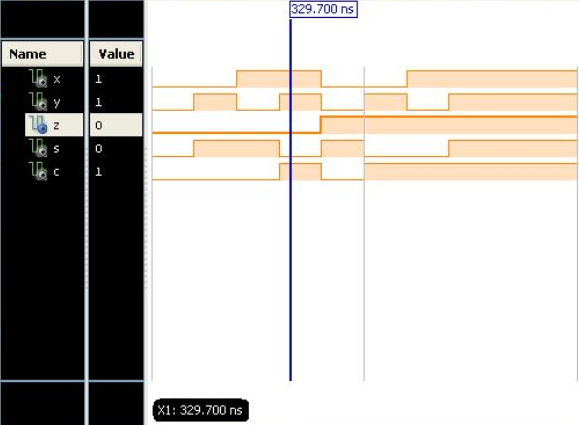
z<='1';

wait for 100 ns;

wait;

end process;END;

Synthesis of the design to see logic resources and timing detail:



From the above figure we can clearly see that at 329.700 ns ,input values are x=1 ,y=1,z=0 and output values are s=0,c=1 .

**Exercise #3:** Design full adder in **behavioral** style of modelling.

VHDL coding for the design:

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity fulladder171ar is

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

s,c : out STD\_LOGIC);

end fulladder171ar;

architecture Behavioral of fulladder171ar is

begin

process(x, y, z)

begin

if (x='0' and y='0' and z='0') then

s <= '0';

c <= '0';

elsif (x='0' and y='0' and z='1') then

s <= '1';

c <= '0';

elsif (x='0' and y='1' and z='0') then

s <= '1';

c <= '0';

elsif (x='0' and y='1' and z='1') then

s <= '0';

c <= '1';

elsif (x='1' and y='0' and z='0') then

s <= '1';

c <= '0';

elsif (x='1' and y='0' and z='1') then

s <= '0';

c <= '1';

elsif (x='1' and y='1' and z='0') then

s <= '0';

c <= '1';

else

s <= '1';

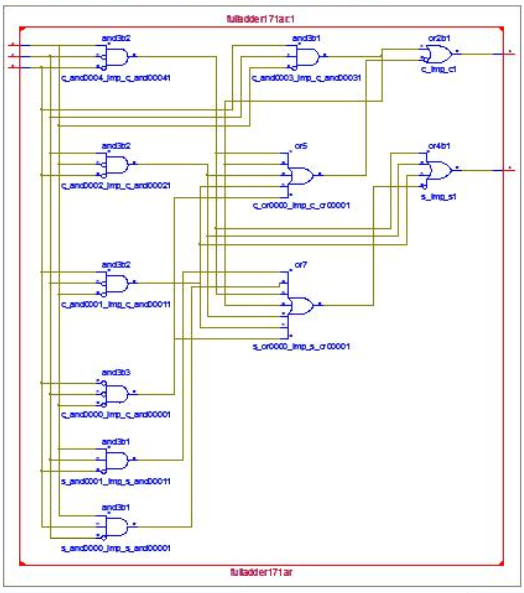
c <= '1';

end if;

end process;

end Behavioral;

Register Transfer Logic (RTL) schematic of the design**:**



Test bench coding:

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY test\_beh IS

END test\_beh;

ARCHITECTURE behavioral OF test\_beh IS

COMPONENT fulladder171ar

PORT(

x : IN std\_logic;

y : IN std\_logic;

z : IN std\_logic;

s : OUT std\_logic;

c : OUT std\_logic

);

END COMPONENT;

signal x : std\_logic := '0';

signal y : std\_logic := '0';

signal z : std\_logic := '0';

signal s : std\_logic;

signal c : std\_logic;

BEGIN

uut: fulladder171ar PORT MAP (

x => x,

y => y,

z => z,

s => s,

c => c

);

stim\_proc: process

begin

x<='0';

y<='0';

z<='0';

wait for 100 ns;

x<='0';

y<='1';

z<='0';

wait for 100 ns;

x<='1';

y<='0';

z<='0';

wait for 100 ns;

x<='1';

y<='1';

z<='0';

wait for 100 ns;

x<='0';

y<='0';

z<='1';

wait for 100 ns;

x<='0';

y<='1';

z<='1';

wait for 100 ns;

x<='1';

y<='0';

z<='1';

wait for 100 ns;

x<='1';

y<='1';

z<='1';

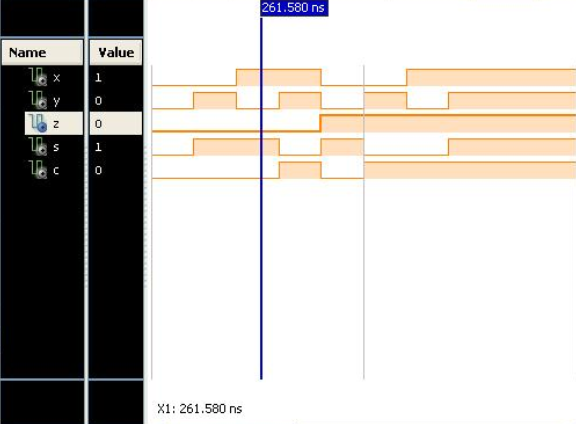
wait for 100 ns;

wait;

end process;

END;

Synthesis of the design to see logic resources and timing detail:



From the above figure we can clearly see that at 261.580 ns ,input values are x=1 ,y=0,z=0 and output values are s=1,c=0 .

Exercise#3: Design full adder using two half adders in structural style of modelling.

VHDL coding for the design:

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity fulladder171st is

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

s,c : out STD\_LOGIC);

end fulladder171st;

architecture Structural of fulladder171st is

signal s1, c1, c2 : std\_logic;

component HaffAdder

port (x, y : in std\_logic;

s, c : out std\_logic);

end component ;

begin

HA1: HaffAdder port map (x, y, s1, c1);

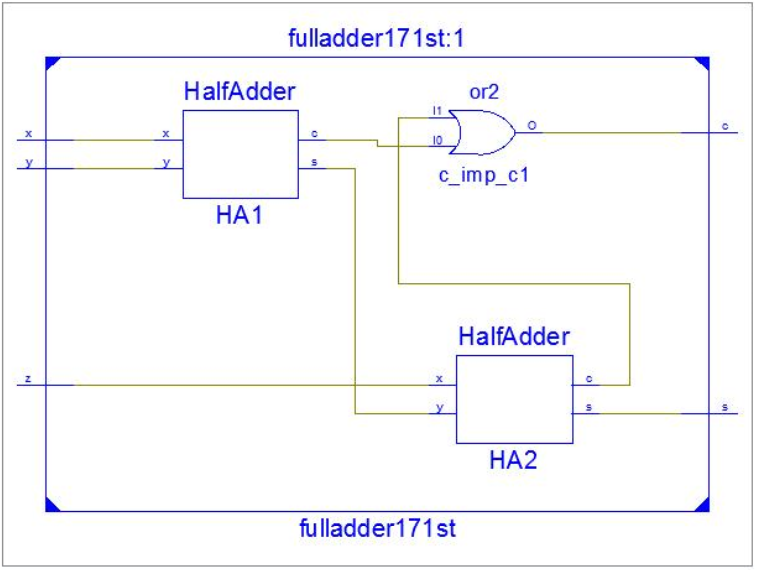
HA2: HaffAdder port map (z, s1, s, c2);

c <= c1 or c2;

c <= c1 or c2;

end Structural;

Register Transfer Logic (RTL) schematic of the design**:**



Test bench coding:

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY test\_st IS

END test\_st;

ARCHITECTURE behavior OF test\_st IS

COMPONENT fulladder171st

PORT(

x : IN std\_logic;

y : IN std\_logic;

z : IN std\_logic;

s : OUT std\_logic;

c : OUT std\_logic

);

END COMPONENT;

signal x : std\_logic := '0';

signal y : std\_logic := '0';

signal z : std\_logic := '0';

signal s : std\_logic;

signal c : std\_logic;

BEGIN

uut: fulladder171st PORT MAP (

x => x,

y => y,

z => z,

s => s,

c => c

);

stim\_proc: process

begin

x<='0';

y<='0';

z<='0';

wait for 100 ns;

x<='0';

y<='1';

z<='0';

wait for 100 ns;

x<='1';

y<='0';

z<='0';

wait for 100 ns;

x<='1';

y<='1';

z<='0';

wait for 100 ns;

x<='0';

y<='0';

z<='1';

wait for 100 ns;

x<='0';

y<='1';

z<='1';

wait for 100 ns;

x<='1';

y<='0';

z<='1';

wait for 100 ns;

x<='1';

y<='1';

z<='1';

wait for 100 ns;

wait;

end process;

END;

Synthesis of the design to see logic resources and timing detail:

