**VHDL code to implement FULL ADDER using Structural, Data flow and Behavioral modeling**

**T Raja Aadhithan**

**602162021**

**Behavioral modeling of Full Adder:**

Design Code:

--behavioral style of coding:

library ieee;

use ieee.std\_logic\_1164.all;

entity full\_adder is

    port (a, b, cin : in std\_logic; sum, cout : out std\_logic); --signal instatiation

end entity;

architecture behavioral of full\_adder is begin --architecture of full adder

    process(a,b,cin) begin -- sensiivity list is given

        if (a = '0' and b = '0' and cin = '0') then sum <= '0'; cout <= '0';

        elsif (a = '0' and b = '0' and cin = '1') then sum <= '1'; cout <= '0';

        elsif (a = '0' and b = '1' and cin = '0') then sum <= '1'; cout <= '0';

        elsif (a = '0' and b = '1' and cin = '1') then sum <= '0'; cout <= '1';

        elsif (a = '1' and b = '0' and cin = '0') then sum <= '1'; cout <= '0';

        elsif (a = '1' and b = '0' and cin = '1') then sum <= '0'; cout <= '1';

        elsif (a = '1' and b = '1' and cin = '0') then sum <= '0'; cout <= '1';

        elsif (a = '1' and b = '1' and cin = '1') then sum <= '1'; cout <= '1';

        else sum <= 'X'; cout <= 'X'; --for unknown value of inputs

        end if;

    end process;

end behavioral; -- end architechture

Test bench Code:

library ieee;

use ieee.std\_logic\_1164.all;

entity full\_adder\_tb is

end full\_adder\_tb;

architecture behavior of full\_adder\_tb is

    component full\_adder

        port(a,b,cin : IN  std\_logic; sum,cout : OUT  std\_logic);

    end component;

     signal a,b,cin,cout,sum : std\_logic;

     begin

   uut: full\_adder port map( a => a, b => b, cin => cin, sum => sum, cout => cout );

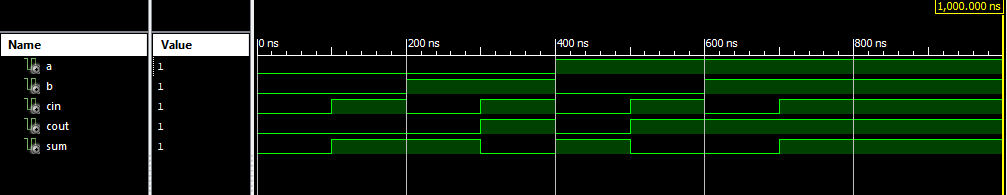
    a <= '0', '1' after 400 ns;

    b <= '0', '1' after 200 ns, '0' after 400 ns, '1' after 600 ns;

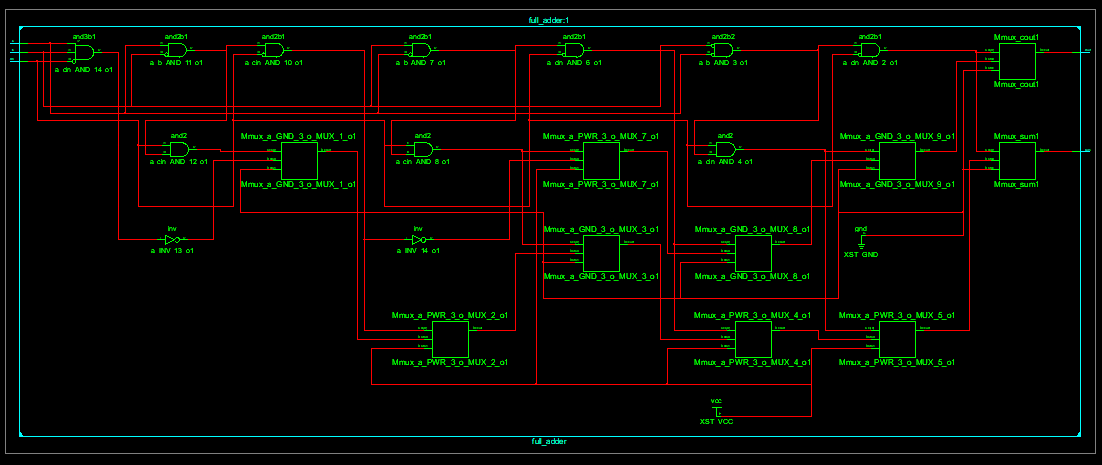
 cin<='0','1' after 100 ns,'0' after 200 ns,'1' after 300 ns,'0' after 400ns,'1' after 500 ns,'0' after 600 ns,'1' after 700 ns;

end behavior;

Simulation result:



RTL of full adder:



**Structural modeling of Full Adder:**

Design Code:

--structural style of coding:

--code for or gate:

library ieee; --library instantiation

use ieee.std\_logic\_1164.all;

entity or\_gate is port ( a,b : in std\_logic; y : out std\_logic); end or\_gate; -- entity of or gate

architecture model\_or of or\_gate is --architecture of or gate

    begin y <= a or b;

end model\_or; -- end architecture

--code for xor gate:

library ieee; --library instantiation

use ieee.std\_logic\_1164.all;

entity xor\_gate is port ( a,b : in std\_logic; y : out std\_logic); end xor\_gate; --entity of xor gate

architecture model\_xor of xor\_gate is --architecture of xor gate

    begin y <= a xor b;

end model\_xor; -- end architecture

--code for and gate:

library ieee; --library instantiation

use ieee.std\_logic\_1164.all;

entity and\_gate is port ( a,b : in std\_logic; y : out std\_logic); end and\_gate; --entity of and gate

architecture model\_and of and\_gate is --architecture of and gate

    begin y <= a and b;

end model\_and; -- end architecture

--code for full adder:

library ieee; --library instantiation

use ieee.std\_logic\_1164.all;

entity Full\_Adder is --entity of full adder

    Port ( a, b, cin : in  STD\_LOGIC; -- input bits

           sum, cout : out  STD\_LOGIC ); -- output bins

end Full\_Adder; -- end entity

architecture structural of Full\_Adder is -- architecture of full adder

    component or\_gate port( a,b : in std\_logic ; y : out std\_logic); end component;--instantiating or gate inside FA

    component and\_gate port( a,b : in std\_logic ; y : out std\_logic); end component; --instantiating and gate inside FA

    component xor\_gate port( a,b : in std\_logic ; y : out std\_logic); end component; --instantiating xor gate inside FA

    signal x,w,u,v,p : std\_logic; --instantiating wires to be used

    begin

        x1: xor\_gate port map(a,b,x); --portmapping the components to signals

        x2: xor\_gate port map(cin,x,sum);

        a1: and\_gate port map(a,b,w);

        a2: and\_gate port map(cin,b,u);

        a3: and\_gate port map(a,cin,v);

        o1: or\_gate port map(w,u,p);

        o2: or\_gate port map(v,p,cout);

end structural; -- end architecture

Test bench code for Full adder:

-- testbench of full adder

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY full\_adder\_tb IS

END full\_adder\_tb;

ARCHITECTURE behavior OF full\_adder\_tb IS

    COMPONENT full\_adder

    PORT(a,b,cin : IN  std\_logic;

         sum,cout : OUT  std\_logic);

    END COMPONENT;

    signal a,b,cin,cout,sum : std\_logic;

BEGIN

  uut: Full\_Adder PORT MAP ( -- Instantiate the Unit Under Test (UUT)

a => a, b => b, cin => cin, sum => sum, cout => cout );

    process begin

    a <= '0';

    b <= '0';

    cin <= '0';

    wait for 100 ns;

    a <= '0';

    b <= '0';

    cin <= '1';

    wait for 100 ns;

    a <= '0';

    b <= '1';

    cin <= '0';

    wait for 100 ns;

    a <= '0';

    b <= '1';

    cin <= '1';

    wait for 100 ns;

    a <= '1';

    b <= '0';

    cin <= '0';

    wait for 100 ns;

    a <= '1';

    b <= '0';

    cin <= '1';

    wait for 100 ns;

    a <= '1';

    b <= '1';

    cin <= '0';

    wait for 100 ns;

    a <= '1';

    b <= '1';

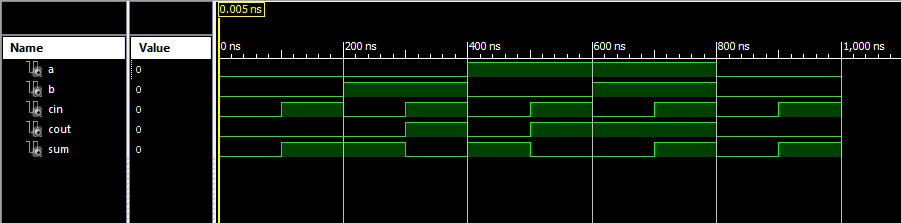
    cin <= '1';

    wait for 100 ns;

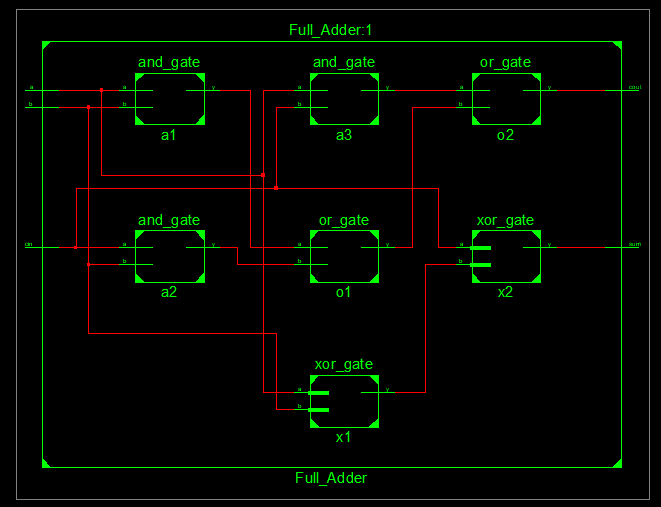
    end process;

END;

Stimulation output:



RTL of Full adder:



**Data Flow modeling of Full Adder:**

Design Code:

library ieee; -- library instantiation

use ieee.std\_logic\_1164.all;

entity full\_adder is -- entity

    port (a, b, cin : in std\_logic; sum, cout : out std\_logic); -- input and output instantiation

end entity;

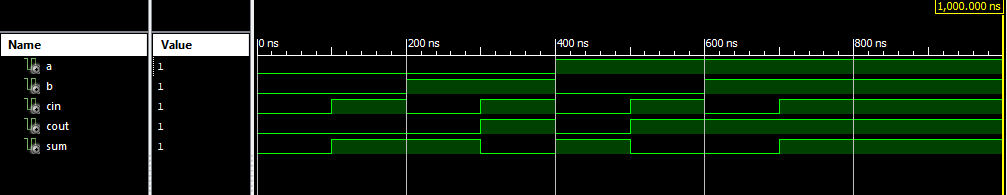
architecture data\_flow of full\_adder is begin -- architecture of full adder

    sum <= a xor b xor cin; -- logic for sum

    cout <= ( a and b ) or ( a and cin ) or ( b and cin ); -- logic for C out

end data\_flow;

Simulation Result:



RTL of Full adder:

