**AIM: TO IMPLEMENT HALF ADDER AND FULLADDER**

**HALFADDER**

library IEEE;

use IEEE.std\_logic\_1164.all;

entity ha is

port(

a,b : IN std\_logic;

HA\_sum, HA\_carry : OUT std\_logic);

end ha;

architecture dataflow of ha is

begin

HA\_sum <= a xor b;

HA\_carry <= a and b;

end dataflow;

**HALFADDER\_TB CODE**

library IEEE;

use IEEE.std\_logic\_1164.all;

entity ha1\_tb is

end entity;

architecture tb of ha1\_tb is

component ha is

port(a,b : IN std\_logic;

HA\_sum, HA\_carry : OUT std\_logic);

end component;

signal a, b, HA\_sum, HA\_carry : std\_logic;

begin

uut: ha port map(

a => a, b => b,

HA\_sum => HA\_sum,

HA\_carry => HA\_carry);

stim: process

begin

a <= '0';

b <= '0';

wait for 10 ns;

a <= '0';

b <= '1';

wait for 10 ns;

a <= '1';

b <= '0';

wait for 10 ns;

a <= '1';

b <= '1';

wait for 10 ns;

wait;

end process;

end tb;

**FULL ADDER :**

library IEEE;

use IEEE.std\_logic\_1164.all;

entity fa is

port(

a,b,cin : IN std\_logic;

FA\_sum, FA\_carry : OUT std\_logic);

end fa;

architecture dataflow of fa is

begin

FA\_sum <= (a xor b) xor cin;

FA\_carry <= (a and b) or (b and cin) or (cin and a);

end dataflow;

Fa\_ tb code

library IEEE;

use IEEE.std\_logic\_1164.all;

entity fa1\_tb is

end entity;

architecture tb of fa1\_tb is

component fa is

port(a,b,cin : IN std\_logic;

FA\_sum, FA\_carry : OUT std\_logic);

end component;

signal a, b, cin,FA\_sum, FA\_carry : std\_logic;

begin

uut: fa port map(

a => a, b => b,

cin => cin,

FA\_sum => FA\_sum,

FA\_carry => FA\_carry);

stim: process

begin

a <= '0';

b <= '0';

cin <= '0';

wait for 10 ns;

a <= '0';

b <= '0';

cin <= '1';

wait for 10 ns;

a <= '0';

b <= '1';

cin <= '0';

wait for 10 ns;

a <= '0';

b <= '1';

cin <= '1';

wait for 10 ns;

a <= '1';

b <= '0';

cin <= '0';

wait for 10 ns;

a <= '1';

b <= '0';

cin <= '1';

wait for 10 ns;

a <= '1';

b <= '1';

cin <= '0';

wait for 10 ns;

a <= '1';

b <= '1';

cin <= '1';

wait for 10 ns;

wait;

end process;

end tb;