**VHDL code for full adder using Behavioral modeling**

library IEEE;  
use IEEE.STD\_LOGIC\_1164.ALL;  
use IEEE.STD\_LOGIC\_ARITH.ALL;  
use IEEE.STD\_LOGIC\_UNSIGNED.ALL;  
entity full\_adder is  
Port ( a : in STD\_LOGIC;  
b : in STD\_LOGIC;  
c : in STD\_LOGIC;  
sum : out STD\_LOGIC;  
cout : out STD\_LOGIC);  
end full\_adder;  
architecture test\_fa of full\_adder is  
begin  
process(a,b,c)  
begin  
if(a='0' and b='0' and c='0') then  
sum <= '0';cout <= '0';  
elsif( a='0' and b='0' and c='1')then  
sum <= '1' ;  
cout <= '0' ;  
elsif ( a='0' and b='1' and c='0') then  
sum <= '1';  
cout <= '0 ';  
elsif( a='0' and b='1' and c='1')then  
sum <= '0';  
cout <= '1';  
elsif( a='1' and b='0' and c='0')then  
sum <= '1';  
cout <= '0';  
elsif( a='1' and b='0' and c='1')then  
sum <= '0';  
cout <= '1';  
elsif( a='1' and b='1' and c='0')then  
sum <= '0';  
cout <= '1';  
else  
sum <= '1' ;  
cout <= '1';  
end if;  
end process;  
end test\_fa;  
  
  
**Test bench Code for Full Adder:**  
  
LIBRARY ieee;  
USE ieee.std\_logic\_1164.ALL;  
ENTITY tb\_test\_fa IS  
END tb\_test\_fa;  
ARCHITECTURE behavior OF tb\_test\_fa IS  
COMPONENT test\_Full\_Adder  
PORT(  
x : IN std\_logic;  
y : IN std\_logic;  
z : IN std\_logic;  
sum : OUT std\_logic;  
cout : OUT std\_logic  
);  
END COMPONENT;  
--Inputs  
signal x : std\_logic := '0';  
signal y : std\_logic := '0';  
signal z : std\_logic := '0';  
--Outputs  
signal sum : std\_logic;  
signal cout : std\_logic;  
-- No clocks detected in port list. Replace <clock> below with  
-- appropriate port name  
BEGIN  
-- Instantiate the Unit Under Test (UUT)  
uut: test\_Full\_Adder PORT MAP (  
x => x,  
y => y,  
z => z,  
sum => sum,  
cout => cout  
);-- Stimulus process  
process  
begin  
x <= '0';  
y <= '0';  
z <= '0';  
wait for 10 ns;  
x <= '0';  
y <= '0';  
z <= '1';  
wait for 10 ns;  
x <= '0';  
y <= '1';  
z <= '0';  
wait for 10 ns;  
x <= '0';  
y <= '1';  
z <= '1';  
wait for 10 ns;  
x <= '1';  
y <= '0';  
z <= '0';  
wait for 10 ns;  
x <= '1';  
y <= '0';  
z <= '1';  
wait for 10 ns;  
x <= '1';  
y <= '1';  
z <= '0';  
wait for 10 ns;  
x <= '1';  
y <= '1';  
z <= '1';  
wait for 10 ns;  
end process;  
END;

**VHDL code for full adder using Data flow modeling**  
  
library IEEE;  
use IEEE.STD\_LOGIC\_1164.ALL;  
entity full\_adder is  
Port ( a : in STD\_LOGIC;  
b : in STD\_LOGIC;  
c : in STD\_LOGIC;  
sum : out STD\_LOGIC;  
cout : out STD\_LOGIC);  
end full\_adder;  
architecture test\_fa of full\_adder is  
begin  
sum <= A XOR B XOR Cin ;  
cout <= (A AND B) OR (Cin AND A) OR (Cin AND B) ;  
end test\_fa;

**Test bench Code for Full Adder:**  
  
LIBRARY ieee;  
USE ieee.std\_logic\_1164.ALL;  
ENTITY tb\_test\_fa IS  
END tb\_test\_fa;  
ARCHITECTURE behavior OF tb\_test\_fa IS  
COMPONENT test\_Full\_Adder  
PORT(  
x : IN std\_logic;  
y : IN std\_logic;  
z : IN std\_logic;  
sum : OUT std\_logic;  
cout : OUT std\_logic  
);  
END COMPONENT;  
--Inputs  
signal x : std\_logic := '0';  
signal y : std\_logic := '0';  
signal z : std\_logic := '0';  
--Outputs  
signal sum : std\_logic;  
signal cout : std\_logic;  
-- No clocks detected in port list. Replace <clock> below with  
-- appropriate port name  
BEGIN  
-- Instantiate the Unit Under Test (UUT)  
uut: test\_Full\_Adder PORT MAP (  
x => x,  
y => y,  
z => z,  
sum => sum,  
cout => cout  
);-- Stimulus process  
process  
begin  
wait for 5 ns;  
x <= '0';  
y <= '0';  
z <= '0';  
wait for 10 ns;  
x <= '0';  
y <= '0';  
z <= '1';  
wait for 10 ns;  
x <= '0';  
y <= '1';  
z <= '0';  
wait for 10 ns;  
x <= '0';  
y <= '1';  
z <= '1';  
wait for 10 ns;  
x <= '1';  
y <= '0';  
z <= '0';  
wait for 10 ns;  
x <= '1';  
y <= '0';  
z <= '1';  
wait for 10 ns;  
x <= '1';  
y <= '1';  
z <= '0';  
wait for 10 ns;  
x <= '1';  
y <= '1';  
z <= '1';  
wait for 10 ns;  
end process;  
END;

**VHDL code for full adder using structural modeling**  
  
  
  
**Step1: VHDL code for Half Adder:**  
  
  
  
library IEEE;  
use IEEE.STD\_LOGIC\_1164.ALL;  
entity test\_HA is  
Port ( a : in STD\_LOGIC;  
b : in STD\_LOGIC;  
sum : out STD\_LOGIC;  
cout : out STD\_LOGIC);  
end test\_HA;  
architecture data\_flow\_test of test\_HA is  
begin  
sum<= a xor b;  
cout<= a and b;  
end data\_flow\_test;  
  
  
  
**Step 2: VHDL code for OR gate:**  
  
  
  
library IEEE;  
use IEEE.STD\_LOGIC\_1164.ALL;  
entity test\_or is  
Port ( p : in STD\_LOGIC;  
q : in STD\_LOGIC;  
r : out STD\_LOGIC);  
end test\_or;  
architecture data\_flow\_test of test\_or is  
begin  
r<= p or q;  
end data\_flow\_test;  
  
  
  
**Step3: VHDL code for full adder using structural modeling:**  
  
  
  
library IEEE;  
use IEEE.STD\_LOGIC\_1164.ALL;  
entity test\_Full\_Adder is  
Port ( x : in STD\_LOGIC;  
y : in STD\_LOGIC;  
z : in STD\_LOGIC;  
sum : out STD\_LOGIC;cout : out STD\_LOGIC);  
end test\_Full\_Adder;  
architecture Structural\_test of test\_Full\_Adder is  
component test\_HA is  
Port ( a : in STD\_LOGIC;  
b : in STD\_LOGIC;  
sum : out STD\_LOGIC;  
cout : out STD\_LOGIC);  
end component;  
component test\_or is  
Port ( p : in STD\_LOGIC;  
q : in STD\_LOGIC;  
r : out STD\_LOGIC);  
end component;  
signal sum1,carry1,carry2:STD\_LOGIC;  
begin  
comp1:test\_HA port map(x,y,sum1,carry1);  
comp2:test\_HA port map(sum1,z,sum,carry2);  
comp3:test\_or port map(carry1,carry2,cout);  
end Structural\_test;  
  
  
  
**Test bench Code for Full Adder:**  
  
  
  
LIBRARY ieee;  
USE ieee.std\_logic\_1164.ALL;  
ENTITY tb\_test\_fa IS  
END tb\_test\_fa;  
ARCHITECTURE behavior OF tb\_test\_fa IS  
COMPONENT test\_Full\_Adder  
PORT(  
x : IN std\_logic;  
y : IN std\_logic;  
z : IN std\_logic;  
sum : OUT std\_logic;  
cout : OUT std\_logic  
);  
END COMPONENT;  
--Inputs  
signal x : std\_logic := '0';  
signal y : std\_logic := '0';  
signal z : std\_logic := '0';  
--Outputs  
signal sum : std\_logic;  
signal cout : std\_logic;  
-- No clocks detected in port list. Replace <clock> below with  
-- appropriate port name  
BEGIN  
-- Instantiate the Unit Under Test (UUT)  
uut: test\_Full\_Adder PORT MAP (  
x => x,  
y => y,  
z => z,  
sum => sum,  
cout => cout  
);-- Stimulus process  
process  
begin  
x <= '0';  
y <= '0';  
z <= '0';  
wait for 10 ns;  
x <= '0';  
y <= '0';  
z <= '1';  
wait for 10 ns;  
x <= '0';  
y <= '1';  
z <= '0';  
wait for 10 ns;  
x <= '0';  
y <= '1';  
z <= '1';  
wait for 10 ns;  
x <= '1';  
y <= '0';  
z <= '0';  
wait for 10 ns;  
x <= '1';  
y <= '0';  
z <= '1';  
wait for 10 ns;  
x <= '1';  
y <= '1';  
z <= '0';  
wait for 10 ns;  
x <= '1';  
y <= '1';  
z <= '1';  
wait for 10 ns;  
end process;  
END

**VHDL code for 4:1 MUX**

library IEEE;  
use IEEE.STD\_LOGIC\_1164.all;  
  
entity mux\_test is  
port(  
A,B,C,D : in STD\_LOGIC;  
S0,S1: in STD\_LOGIC;  
Z: out STD\_LOGIC  
);  
end mux\_test;  
  
architecture beh\_test of mux\_test is  
begin  
process (A,B,C,D,S0,S1) is  
begin  
if (S0 ='0' and S1 = '0') then  
Z <= A;  
elsif (S0 ='1' and S1 = '0') then  
Z <= B;  
elsif (S0 ='0' and S1 = '1') then  
Z <= C;  
else  
Z <= D;  
end if;  
end process;  
end beh\_test;  
  
**Test Bench Code for 4 to 1 Multiplexer:**  
  
LIBRARY ieee;  
USE ieee.std\_logic\_1164.ALL;  
ENTITY tb\_mux\_test IS  
END tb\_mux\_test;  
ARCHITECTURE behavior OF tb\_mux\_test IS  
-- Component Declaration for the Unit Under Test (UUT)  
COMPONENT mux\_test  
PORT(  
A : IN std\_logic;  
B : IN std\_logic;  
C : IN std\_logic;  
D : IN std\_logic;  
S0 : IN std\_logic;  
S1 : IN std\_logic;  
Z : OUT std\_logic  
);  
END COMPONENT;  
--Inputs  
signal A : std\_logic := '0';  
signal B : std\_logic := '0';  
signal C : std\_logic := '0';  
signal D : std\_logic := '0';  
signal S0 : std\_logic := '0';  
signal S1 : std\_logic := '0';  
--Outputs  
signal Z : std\_logic;  
BEGIN  
uut: mux\_test PORT MAP (  
A => A,  
B => B,  
C => C,  
D => D,  
S0 => S0,  
S1 => S1,  
Z => Z  
);  
  
process  
begin  
wait for 5 ns;  
  
A <= '1';  
B <= '0';  
C <= '1';  
D <= '0';  
  
S0 <= '0'; S1 <= '0';  
wait for 10 ns;  
  
S0 <= '0'; S1 <= '1';  
wait for 10 ns;  
  
S0 <= '1'; S1 <= '0';  
wait for 10 ns;  
  
S0 <= '1'; S1 <= '1';  
wait for 10 ns;  
  
end process;  
END;