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;
```