**INVERTOR VHDL**

library IEEE;

use IEEE.std\_logic\_1164.all;

use IEEE.std\_logic\_arith.all;

use IEEE.std\_logic\_unsigned.all;

entity invertor is

port(A: in std\_logic;

B: out std\_logic;

end invertor;

architecture BEHAVIORAL of invertor is

begin

b <= NOT A;

end BEHAVIORAL;

**INVERTOR TESTBENCH**

library IEEE;

use IEEE.std\_logic\_1164.all;

entity inverterbench is

end inverterbench;

architecture tb of inverterbench is

component inverter\_gate is

port(

a: in std\_logic;

q: out std\_logic);

end component;

signal a\_in, q\_out: std\_ulogic;

begin

DUT: inverter\_gate port map(a\_in, q\_out);

process

begin

a\_in <= '0';

wait for 100 ns;

a\_in <= '1';

wait for 100 ns;

a\_in <= '0';

wait;

end process;

end tb;

**OR-GATE VHDL**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity orgate is

Port ( x: in STD\_LOGIC;

y: in STD\_LOGIC;

z: out STD\_LOGIC);

end OR\_gate;

architecture data\_and of orgate is

begin

z<= x OR y ;

end data\_and;

**OR-GATE TESTBENCH**

library ieee;

use ieee.std\_logic\_1164.all;

entity orgate\_tb id

end orgate\_tb;

architecture behavioral or orgate\_tb is

component orgate is

port(x,y : in std\_logic;

z : out std\_logic);

end component;

signal X,Y,Z : std\_ulogic;

begin

DUT : orgate port map(x,y,z);

stim\_proc : process

begin

wait for 10ns;

x\_in <= '0';

y\_in <= '0';

wait for 10 ns;

x\_in <= '0';

y\_in <= '1';

wait for 10 ns;

x\_in <= '1';

y\_in <= '0';

wait for 10 ns;

x\_in <= '1';

y\_in <= '1';

wait for 10 ns;

end process;

end behavioral;

**XNOR VHDL**

library IEEE;

use IEEE.std\_logic\_1164.all;

entity xnorgate is

port(

a: in std\_logic;

b: in std\_logic;

q: out std\_logic );

end xnorgate;

architecture art1 of xnorgate is

begin

process(a,b) is

begin

q <= a xnor b;

end process;

end art1;

**XNOR TESTBENCH**

library IEEE;

use IEEE.std\_logic\_1164.all;

entity xnorbench is

--empty

end xnorbench;

architecture tb of xnorbench is

component xnorgate is

port(

a: in std\_logic;

b: in std\_logic;

q: out std\_logic);

end component;

signal a\_in, b\_in, q\_out: std\_ulogic;

begin

DUT : xnorgate port map(a\_in, b\_in , q\_out);

process

begin

a\_in <= '0';

b\_in <= '0';

wait for 100 ns;

a\_in <= '0';

b\_in <= '1';

wait for 100 ns;

a\_in <= '1';

b\_in <= '0';

wait for 100 ns;

a\_in <= '1';

b\_in <= '1';

wait for 100 ns;

a\_in <= '0';

b\_in <= '0';

wait;

end process;

end tb;

**AND-GATE VHDL**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity and\_gate is

Port ( A: in STD\_LOGIC;

B: in STD\_LOGIC;

Y: out STD\_LOGIC);

end and\_gate;

architecture Behavioural of and\_gate is

begin

Y <= A and B ;

end Behavioural;

**AND-GATE TESTBENCH**

library IEEE;

use IEEE.std\_logic\_1164.all;

entity andbench is

--empty

end andbench;

architecture tb of andbench is

component and\_gate is

port(

A: in std\_logic;

B: in std\_logic;

Y: out std\_logic);

end component;

signal a\_in, b\_in, q\_out: std\_ulogic;

begin

DUT : and\_gate port map(a\_in, b\_in ,q\_out);

process

begin

a\_in <= '0';

b\_in <= '0';

wait for 100 ns;

a\_in <= '0';

b\_in <= '1';

wait for 100 ns;

a\_in <= '1';

b\_in <= '0';

wait for 100 ns;

a\_in <= '1';

b\_in <= '1';

wait for 100 ns;

a\_in <= '0';

b\_in <= '0';

wait;

end process;

end tb;

**NAND-GATE VHDL**

library IEEE;

use IEEE.std\_logic\_1164.all;

entity nandgate is

port(

a: in std\_logic;

b: in std\_logic;

q: out std\_logic );

end nandgate;

architecture art1 of nandgate is

begin

process(a,b) is

begin

q <= a nand b;

end process;

end art1;

**NAND-GATE TESTBENCH**

library IEEE;

use IEEE.std\_logic\_1164.all;

entity nandbench is

--empty

end nandbench;

architecture tb of nandbench is

component nandgate is

port(

a: in std\_logic;

b: in std\_logic;

q: out std\_logic);

end component;

signal a\_in, b\_in, q\_out: std\_ulogic;

begin

DUT : nandgate port map(a\_in, b\_in , q\_out);

process

begin

a\_in <= '0';

b\_in <= '0';

wait for 100 ns;

a\_in <= '0';

b\_in <= '1';

wait for 100 ns;

a\_in <= '1';

b\_in <= '0';

wait for 100 ns;

a\_in <= '1';

b\_in <= '1';

wait for 100 ns;

a\_in <= '0';

b\_in <= '0';

wait;

end process;

end tb;

**NOR-GATE VHDL**

library IEEE;

use IEEE.std\_logic\_1164.all;

entity norgate is

port(

a: in std\_logic;

b: in std\_logic;

q: out std\_logic );

end norgate;

architecture art1 of norgate is

begin

process(a,b) is

begin

q <= a nor b;

end process;

end art1;

**NOR-GATE TESTBENCH**

library IEEE;

use IEEE.std\_logic\_1164.all;

entity norbench is

--empty

end norbench;

architecture tb of norbench is

component norgate is

port(

a: in std\_logic;

b: in std\_logic;

q: out std\_logic);

end component;

signal a\_in, b\_in, q\_out: std\_ulogic;

begin

DUT : norgate port map(a\_in, b\_in , q\_out);

process

begin

a\_in <= '0';

b\_in <= '0';

wait for 100 ns;

a\_in <= '0';

b\_in <= '1';

wait for 100 ns;

a\_in <= '1';

b\_in <= '0';

wait for 100 ns;

a\_in <= '1';

b\_in <= '1';

wait for 100 ns;

a\_in <= '0';

b\_in <= '0';

wait;

end process;

end tb;

**XOR-GATE VHDL**

library IEEE;

use IEEE.std\_logic\_1164.all;

entity xor\_gate is

port(

a: in std\_logic;

b: in std\_logic;

q: out std\_logic );

end xor\_gate;

architecture rt1 of xor\_gate is

begin

process(a,b) is

begin

q <= a xor b;

end process;

end rt1;

**XOR-GATE TESTBENCH**

library IEEE;

use IEEE.std\_logic\_1164.all;

entity xorbench is

--empty

end xorbench;

architecture tb of xorbench is

component xor\_gate is

port(

a: in std\_logic;

b: in std\_logic;

q: out std\_logic);

end component;

signal a\_in, b\_in, q\_out: std\_ulogic;

begin

DUT : xor\_gate port map(a\_in, b\_in , q\_out);

process

begin

a\_in <= '0';

b\_in <= '0';

wait for 100 ns;

a\_in <= '0';

b\_in <= '1';

wait for 100 ns;

a\_in <= '1';

b\_in <= '0';

wait for 100 ns;

a\_in <= '1';

b\_in <= '1';

wait for 100 ns;

a\_in <= '0';

b\_in <= '0';

wait;

end process;

end tb;

**ADDER-VHDL**

library IEEE;

use IEEE.std\_logic\_1164.all;

entity full\_adder is

port(

A : in STD\_LOGIC;

B : in STD\_LOGIC;

Cin : in STD\_LOGIC;

S : out STD\_LOGIC;

Cout : out STD\_LOGIC );

end full\_adder;

architecture rt1 of full\_adder is

begin

process(A,B, Cin) is

begin

S <= A XOR B XOR Cin ;

Cout <= (A AND B) OR (Cin AND A) OR (Cin AND B) ;

end process;

end rt1;

**ADDER TESTBENCH**

library IEEE;

use IEEE.std\_logic\_1164.all;

entity adderbench is

--empty

end adderbench;

architecture tb of adderbench is

component full\_adder is

port(

A : in STD\_LOGIC;

B : in STD\_LOGIC;

Cin : in STD\_LOGIC;

S : out STD\_LOGIC;

Cout : out STD\_LOGIC);

end component;

--signal a\_in, b\_in, c\_in, q\_out, c\_out: std\_ulogic;

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

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

signal c\_in : std\_logic := '0';

signal q\_out : std\_ulogic;

signal c\_out : std\_ulogic;

begin

DUT : full\_adder port map(a\_in, b\_in , c\_in, q\_out, c\_out);

process

begin

a\_in <= '0';

b\_in <= '1';

c\_in <= '0';

wait for 100 ns;

a\_in <= '1';

b\_in <= '0';

c\_in <= '0';

wait for 100 ns;

a\_in <= '1';

b\_in <= '1';

c\_in <= '0';

wait for 100 ns;

a\_in <= '0';

b\_in <= '0';

c\_in <= '1';

wait for 100 ns;

a\_in <= '0';

b\_in <= '1';

c\_in <= '1';

wait for 100 ns;

a\_in <= '1';

b\_in <= '0';

c\_in <= '1';

wait for 100 ns;

a\_in <= '1';

b\_in <= '1';

c\_in <= '1';

wait for 100 ns;

a\_in <= '0';

b\_in <= '0';

c\_in <= '0';

wait;

end process;

end tb;

**COMPARATOR VHDL**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity comparator is

Port ( A,B : in std\_logic;

G,S,E: out std\_logic);

end comparator;

architecture comp\_arch of comparator is

begin

G <= A and (not B);

S <= (not A) and B;

E <= A xnor B;

end comp\_arch;

**COMPARATOR TESTBENCH**

library IEEE;

use IEEE.std\_logic\_1164.all;

entity comparatorbench is

--empty

end comparatorbench;

architecture tb of comparatorbench is

component comparator is

port(

A : in STD\_LOGIC;

B : in STD\_LOGIC;

G : out STD\_LOGIC;

S : out STD\_LOGIC;

E : out STD\_LOGIC);

end component;

--signal a\_in, b\_in, c\_in, q\_out, c\_out: std\_logic;

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

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

signal g\_out : std\_ulogic;

signal s\_out : std\_ulogic;

signal e\_out : std\_ulogic;

begin

DUT : comparator port map(a\_in, b\_in , g\_out, s\_out, e\_out);

process

begin

a\_in <= '0';

b\_in <= '0';

wait for 100 ns;

a\_in <= '0';

b\_in <= '1';

wait for 100 ns;

a\_in <= '1';

b\_in <= '0';

wait for 100 ns;

a\_in <= '1';

b\_in <= '1';

wait for 100 ns;

a\_in <= '0';

b\_in <= '0';

wait;

end process;

end tb;

**DECODER VHDL**

library IEEE;

use IEEE.STD\_LOGIC\_1164.all;

entity decoder is

port(

a : in STD\_LOGIC\_VECTOR(1 downto 0);

b : out STD\_LOGIC\_VECTOR(3 downto 0)

);

end decoder;

architecture bhv of decoder is

begin

b(0) <= not a(0) and not a(1);

b(1) <= not a(0) and a(1);

b(2) <= a(0) and not a(1);

b(3) <= a(0) and a(1);

end bhv;

**DOCODER TESTBENCH**

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY decoderbench IS

END decoderbench;

ARCHITECTURE behavior OF decoderbench IS

COMPONENT decoder

PORT(

a : IN std\_logic\_vector(1 downto 0);

b : OUT std\_logic\_vector(3 downto 0)

);

END COMPONENT;

signal a : std\_logic\_vector(1 downto 0) := (others => '0');

signal b : std\_logic\_vector(3 downto 0);

BEGIN

uut: decoder PORT MAP (

a => a,

b => b

);

stim\_proc: process

begin

wait for 100 ns;

a <= "00";

wait for 100 ns;

a <= "01";

wait for 100 ns;

a <= "10";

wait for 100 ns;

a <= "11";

wait;

end process;

END;

**MULTIPLIER VHDL**

library ieee;

use ieee.std\_logic\_1164.all;

entity multiply\_behav is

port (A, B : in bit\_vector(1 downto 0);

P : out bit\_vector(3 downto 0)

);

end multiply\_behav;

architecture behavioral of multiply\_behav is

begin

process(A,B) is

begin

case A is

when "00" =>

if B="00" then P<="0000";

elsif B="01" then P<="0000";

elsif B="10" then P<="0000";

else P<="0000";

end if;

when "01" =>

if B="00" then P<="0000";

elsif B="01" then P<="0001";

elsif B="10" then P<="0010";

else P<="0011";

end if;

when "10" =>

if B="00" then P<="0000";

elsif B="01" then P<="0010";

elsif B="10" then P<="0100";

else P<="0110";

end if;

when "11" =>

if B="00" then P<="0000";

elsif B="01" then P<="0011";

elsif B="10" then P<="0110";

else P<="1001";

end if;

end case;

end process;

end architecture;

**MULTIPLIER TESTBENCH**

library ieee;

use ieee.std\_logic\_1164.all;

entity multiply\_behav\_tb is

end multiply\_behav\_tb;

architecture tb of multiply\_behav\_tb is

component multiply\_behav is

port (A, B : in bit\_vector(1 downto 0);

P : out bit\_vector(3 downto 0)

);

end component;

signal A, B : bit\_vector(1 downto 0);

signal P : bit\_vector(3 downto 0);

begin

UUT : multiply\_behav port map (

A => A,

B => B,

P => P);

Force:process

constant period: time := 20 ns;

begin

A <= "00";

B <= "00";

wait for period;

A <= "00";

B <= "01";

wait for period;

A <= "00";

B <= "10";

wait for period;

A <= "00";

B <= "11";

wait for period;

A <= "01";

B <= "00";

wait for period;

A <= "01";

B <= "01";

wait for period;

A <= "01";

B <= "10";

wait for period;

A <= "01";

B <= "11";

wait for period;

A <= "10";

B <= "00";

wait for period;

A <= "10";

B <= "01";

wait for period;

A <= "10";

B <= "10";

wait for period;

A <= "10";

B <= "11";

wait for period;

A <= "11";

B <= "00";

wait for period;

A <= "11";

B <= "01";

wait for period;

A <= "11";

B <= "10";

wait for period;

A <= "11";

B <= "11";

wait for period;

wait;

end process;

end tb;

**MULTIPLEXER VHDL**

library IEEE;

use IEEE.STD\_LOGIC\_1164.all;

entity multiplexer is

port(

A,B,C,D : in STD\_LOGIC;

S0,S1: in STD\_LOGIC;

Z: out STD\_LOGIC

);

end multiplexer;

architecture bhv of multiplexer 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 bhv;

**MULTIPLEXER TESTBENCH**

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY multibench IS

END multibench;

ARCHITECTURE behavior OF multibench IS

COMPONENT multiplexer

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;

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';

signal Z : std\_ulogic;

BEGIN

uut: multiplexer PORT MAP (

A => A,

B => B,

C => C,

D => D,

S0 => S0,

S1 => S1,

Z => Z

);

stim\_proc: process

begin

wait for 100 ns;

A <= '1';

B <= '0';

C <= '1';

D <= '0';

S0 <= '0'; S1 <= '0';

wait for 100 ns;

S0 <= '1'; S1 <= '0';

wait for 100 ns;

S0 <= '0'; S1 <= '1';

wait for 100 ns;

S0 <= '0'; S1 <= '1';

wait for 100 ns;

end process;

END;

**ALU VHDL**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity alu is

Port ( inp\_a : in signed(3 downto 0);

inp\_b : in signed(3 downto 0);

sel : in STD\_LOGIC\_VECTOR (2 downto 0);

out\_alu : out signed(3 downto 0));

end alu;

architecture Behavioral of alu is

begin

process(inp\_a, inp\_b, sel)

begin

case sel is

when "000" =>

out\_alu<= inp\_a + inp\_b; --addition

when "001" =>

out\_alu<= inp\_a - inp\_b; --subtraction

when "010" =>

out\_alu<= inp\_a - 1; --sub 1

when "011" =>

out\_alu<= inp\_a + 1; --add 1

when "100" =>

out\_alu<= inp\_a and inp\_b; --AND gate

when "101" =>

out\_alu<= inp\_a or inp\_b; --OR gate

when "110" =>

out\_alu<= not inp\_a ; --NOT gate

when "111" =>

out\_alu<= inp\_a xor inp\_b; --XOR gate

when others =>

NULL;

end case;

end process;

end Behavioral;

**ALU TESTBENCH**

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

USE ieee.numeric\_std.ALL;

ENTITY Tb\_alu IS

END Tb\_alu;

ARCHITECTURE behavior OF Tb\_alu IS

-- Component Declaration for the Unit Under Test (UUT)

COMPONENT alu

PORT(

inp\_a : IN signed(3 downto 0);

inp\_b : IN signed(3 downto 0);

sel : IN std\_logic\_vector(2 downto 0);

out\_alu : OUT signed(3 downto 0)

);

END COMPONENT;

--Inputs

signal inp\_a : signed(3 downto 0) := (others => '0');

signal inp\_b : signed(3 downto 0) := (others => '0');

signal sel : std\_logic\_vector(2 downto 0) := (others => '0');

--Outputs

signal out\_alu : signed(3 downto 0);

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: alu PORT MAP (

inp\_a => inp\_a,

inp\_b => inp\_b,

sel => sel,

out\_alu => out\_alu

);

-- Stimulus process

stim\_proc: process

begin

-- hold reset state for 100 ns.

wait for 100 ns;

-- insert stimulus here

inp\_a <= "1001";

inp\_b <= "1111";

sel <= "000";

wait for 100 ns;

sel <= "001";

wait for 100 ns;

sel <= "010";

wait for 100 ns;

sel <= "011";

wait for 100 ns;

sel <= "100";

wait for 100 ns;

sel <= "101";

wait for 100 ns;

sel <= "110";

wait for 100 ns;

sel <= "111";

end process;

END;