**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)

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 testbench entity

end adderbench;

architecture tb of adderbench is

-- Component Declaration of Full Adder

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;

-- Testbench Signals

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

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

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

signal q\_out : std\_logic;

signal c\_out : std\_logic;

begin

-- Instantiate the Device Under Test (DUT)

DUT : full\_adder

port map (

A => a\_in,

B => b\_in,

Cin => c\_in,

S => q\_out,

Cout => c\_out

);

-- Stimulus Process

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;

**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)

variable temp : signed(3 downto 0);

begin

case sel is

when "000" => -- Addition

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

when "001" => -- Subtraction

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

when "010" => -- Subtract 1

out\_alu <= inp\_a - 1;

when "011" => -- Add 1

out\_alu <= inp\_a + 1;

when "100" => -- AND

out\_alu <= signed(std\_logic\_vector(inp\_a) and std\_logic\_vector(inp\_b));

when "101" => -- OR

out\_alu <= signed(std\_logic\_vector(inp\_a) or std\_logic\_vector(inp\_b));

when "110" => -- NOT (only on inp\_a)

out\_alu <= signed(not std\_logic\_vector(inp\_a));

when "111" => -- XOR

out\_alu <= signed(std\_logic\_vector(inp\_a) xor std\_logic\_vector(inp\_b));

when others =>

out\_alu <= (others => '0'); -- Default value

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');

-- Output

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

-- Wait for global reset

wait for 100 ns;

-- Test all operations

inp\_a <= "1001"; -- -7 (signed)

inp\_b <= "1111"; -- -1 (signed)

sel <= "000"; -- Addition

wait for 100 ns;

sel <= "001"; -- Subtraction

wait for 100 ns;

sel <= "010"; -- Subtract 1

wait for 100 ns;

sel <= "011"; -- Add 1

wait for 100 ns;

sel <= "100"; -- AND

wait for 100 ns;

sel <= "101"; -- OR

wait for 100 ns;

sel <= "110"; -- NOT

wait for 100 ns;

sel <= "111"; -- XOR

wait;

end process;

END behavior;

**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(1) and not a(0); -- when a = "00"

b(1) <= not a(1) and a(0); -- when a = "01"

b(2) <= a(1) and not a(0); -- when a = "10"

b(3) <= a(1) and a(0); -- when a = "11"

end bhv;

**Decoder Testbench**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity decoderbench is

end decoderbench;

architecture behavior of decoderbench is

-- Component Declaration

component decoder

port (

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

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

);

end component;

-- Signals for input and output

signal a : STD\_LOGIC\_VECTOR(1 downto 0) := (others => '0');

signal b : STD\_LOGIC\_VECTOR(3 downto 0);

begin

-- Instantiate the Unit Under Test (UUT)

uut: decoder port map (

a => a,

b => b

);

-- Stimulus Process

stim\_proc: process

begin

wait for 100 ns;

a <= "00"; -- Expect b = "0001"

wait for 100 ns;

a <= "01"; -- Expect b = "0010"

wait for 100 ns;

a <= "10"; -- Expect b = "0100"

wait for 100 ns;

a <= "11"; -- Expect b = "1000"

wait;

end process;

end behavior;

**D FlipFlop VHDL**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity D\_FLIPFLOP\_SOURCE is

port (

D : in STD\_LOGIC;

CLOCK : in STD\_LOGIC;

Q : out STD\_LOGIC;

Qb : out STD\_LOGIC

);

end D\_FLIPFLOP\_SOURCE;

architecture Behavioral of D\_FLIPFLOP\_SOURCE is

begin

process(CLOCK)

begin

if rising\_edge(CLOCK) then

Q <= D;

Qb <= not D;

end if;

end process;

end Behavioral;

**D FlipFlop Testbench**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity DFF\_tb is

end DFF\_tb;

architecture tb of DFF\_tb is

-- Component Declaration

component D\_FLIPFLOP\_SOURCE is

port (

D : in STD\_LOGIC;

CLOCK : in STD\_LOGIC;

Q : out STD\_LOGIC;

Qb : out STD\_LOGIC

);

end component;

-- Signals

signal D, CLK, Q, Qb : STD\_LOGIC;

begin

-- Instantiate the Unit Under Test (UUT)

uut: D\_FLIPFLOP\_SOURCE

port map (

D => D,

CLOCK => CLK,

Q => Q,

Qb => Qb

);

-- Clock generation: 100ns HIGH, 100ns LOW

clock\_process: process

begin

while true loop

CLK <= '0'; wait for 100 ns;

CLK <= '1'; wait for 100 ns;

end loop;

end process;

-- Stimulus

stim\_process: process

begin

D <= '0'; wait for 150 ns;

D <= '1'; wait for 150 ns;

D <= '0'; wait for 200 ns;

D <= '1'; 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 : out STD\_LOGIC; -- A > B

S : out STD\_LOGIC; -- A < B

E : out STD\_LOGIC -- A = B

);

end comparator;

architecture comp\_arch of comparator is

begin

G <= A and (not B); -- Greater

S <= (not A) and B; -- Smaller

E <= A xnor B; -- Equal

end comp\_arch;

**Comparator Testbench**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity comparatorbench is

end comparatorbench;

architecture tb of comparatorbench is

component comparator

port(

A : in STD\_LOGIC;

B : in STD\_LOGIC;

G : out STD\_LOGIC;

S : out STD\_LOGIC;

E : out STD\_LOGIC

);

end component;

-- Signals for inputs

signal a\_in : STD\_LOGIC := '0';

signal b\_in : STD\_LOGIC := '0';

-- Signals for outputs

signal g\_out : STD\_LOGIC;

signal s\_out : STD\_LOGIC;

signal e\_out : STD\_LOGIC;

begin

-- Instantiate the Unit Under Test (UUT)

DUT: comparator

port map (

A => a\_in,

B => b\_in,

G => g\_out,

S => s\_out,

E => e\_out );

-- Stimulus Process

process

begin

-- Test Case 1: A = 0, B = 0

a\_in <= '0'; b\_in <= '0'; wait for 100 ns;

-- Test Case 2: A = 0, B = 1

a\_in <= '0'; b\_in <= '1'; wait for 100 ns;

-- Test Case 3: A = 1, B = 0

a\_in <= '1'; b\_in <= '0'; wait for 100 ns;

-- Test Case 4: A = 1, B = 1

a\_in <= '1'; b\_in <= '1'; wait for 100 ns;

wait; -- Stop simulation

end process;

end tb;

**JK FlipFlop VHDL**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity JK\_FF\_tb is

end JK\_FF\_tb;

architecture testbench of JK\_FF\_tb is

component JK\_FF

port(

J, K, clk : in STD\_LOGIC;

Q, Qbar : out STD\_LOGIC );

end component;

signal J, K, clk : STD\_LOGIC := '0';

signal Q, Qbar : STD\_LOGIC;

begin

uut: JK\_FF

port map(

J => J,

K => K,

clk => clk,

Q => Q,

Qbar => Qbar );

clock: process

begin

while true loop

clk <= '0';

wait for 200 ns;

clk <= '1';

wait for 200 ns;

end loop;

end process;

stim\_proc: process

begin

-- Test 1: J = 0, K = 0 → Hold state

J <= '0'; K <= '0'; wait for 400 ns;

-- Test 2: J = 0, K = 1 → Reset

J <= '0'; K <= '1'; wait for 400 ns;

-- Test 3: J = 1, K = 0 → Set

J <= '1'; K <= '0'; wait for 400 ns;

-- Test 4: J = 1, K = 1 → Toggle

J <= '1'; K <= '1'; wait for 400 ns;

-- Test 5: Toggle again

J <= '1'; K <= '1'; wait for 400 ns;

wait; -- Stop simulation

end process;

end testbench;

**JK FlipFlop Testbench**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity JK\_FF is

port(

J, K, clk : in std\_logic;

Q, Qbar : out std\_logic );

end JK\_FF;

architecture behavioral of JK\_FF is

signal qn : std\_logic := '0';

begin

process(clk)

begin

if rising\_edge(clk) then

if (J = '0' and K = '0') then -- Hold state

qn <= qn;

elsif (J = '0' and K = '1') then -- Reset

qn <= '0';

elsif (J = '1' and K = '0') then -- Set

qn <= '1';

elsif (J = '1' and K = '1') then -- Toggle

qn <= not qn;

end if;

end if;

end process;

Q <= qn;

Qbar <= not qn;

end behavioral;

**Multiplier VHDL**

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.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)

variable a\_int, b\_int, p\_int : integer;

begin

-- Convert bit\_vector to integer

a\_int := to\_integer(unsigned(A));

b\_int := to\_integer(unsigned(B));

-- Perform multiplication

p\_int := a\_int \* b\_int;

-- Convert result back to 4-bit bit\_vector

P <= std\_logic\_vector(to\_unsigned(p\_int, 4));

end process;

end behavioral;

**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

);

stimulus: process

constant period: time := 20 ns;

begin

for i in 0 to 3 loop

for j in 0 to 3 loop

A <= std\_logic\_vector(to\_unsigned(i, 2));

B <= std\_logic\_vector(to\_unsigned(j, 2));

wait for period;

end loop;

end loop;

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)

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, B, C, D : in std\_logic;

S0, S1 : in std\_logic;

Z : out std\_logic

);

end component;

-- Signals

signal A, B, C, D : std\_logic := '0';

signal S0, S1 : std\_logic := '0';

signal Z : std\_logic;

begin

uut: multiplexer

port map (

A => A,

B => B,

C => C,

D => D,

S0 => S0,

S1 => S1,

Z => Z );

stim\_proc: process

begin

-- Set inputs

A <= '1'; B <= '0'; C <= '1'; D <= '0';

-- Select A

S0 <= '0'; S1 <= '0'; wait for 100 ns;

-- Select B

S0 <= '1'; S1 <= '0'; wait for 100 ns;

-- Select C

S0 <= '0'; S1 <= '1'; wait for 100 ns;

-- Select D

S0 <= '1'; S1 <= '1'; wait for 100 ns;

wait; -- End simulation

end process;

end behavior;

**COUNTER VHDL**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity counter is

port (

clk : in std\_logic; reset : in std\_logic;

count : out std\_logic\_vector(3 downto 0) );

end counter;

architecture behavioral of counter is

signal cnt : std\_logic\_vector(3 downto 0) := "0000";

begin

process(clk, reset)

begin

if reset = '1' then

cnt <= "0000";

elsif rising\_edge(clk) then

cnt <= cnt + 1;

end if;

end process;

count <= cnt;

end behavioral;

**COUNTER Testbench**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity counter\_tb is

end counter\_tb;

architecture testbench of counter\_tb is

-- Component declaration

component counter

port (

clk : in std\_logic;

reset : in std\_logic;

count : out std\_logic\_vector(3 downto 0) );

end component;

-- Testbench signals

signal clk : std\_logic := '0';

signal reset : std\_logic := '0';

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

constant clk\_period : time := 10 ns;

begin

-- Instantiate the Unit Under Test (UUT)

uut: counter

port map (

clk => clk,

reset => reset,

count => count );

-- Clock process

clk\_process : process

begin

while true loop

clk <= '0';

wait for clk\_period / 2;

clk <= '1';

wait for clk\_period / 2;

end loop;

end process;

-- Stimulus process

stim\_proc : process

begin

-- Initial reset

reset <= '1'; wait for 20 ns; reset <= '0';

-- Run the counter for several clock cycles

wait for 200 ns;

-- Apply reset again

reset <= '1'; wait for 20 ns; reset <= '0';

wait for 100 ns;

wait;

end process;

end testbench;