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;
```