1. Problem/Objective

During this lab we will be taking a closer look at combinational circuits. We will be programming the Diligent Spartan 3 FPGA board to simulate these circuits. We will be using Boolean algebra on selected inputs to display the working circuits. This will include a 4-1 MUX, 2-4 Decoder, 4-bit Adder, simple Comparator, 2-bit ALU, and a 4-bit signed multiplier. The circuits we will be simulating help reinforce the concepts discussed in lecture and will make us more familiar with VHDL, using Xilinx ISE, and the Spartan 3 boards.

2. Methodology

Read about and describe here each component listed below:

a) Multiplexers (MUX) - pg. 388 - Section 11.3

A multiplexer (or mux) is a device that selects one of several input signals and forwards the selected input into a single line. A multiplexer of inputs has select lines, which are used to select which input line to send to the output.

b) Decoders - pg. 390 - Section 11.3

A decoder is a combinational logic circuit that converts binary information from the n coded inputs to a maximum of 2^n unique outputs. They are used in a wide variety of applications, including data demultiplexing, seven segment displays, and memory address decoding.

c) Adders - pg. 392 - Section 11.3

An adder is a digital circuit that performs addition of numbers. In many computers and other kinds of processors adders are used in the arithmetic logic units or ALU. They are also utilized in other parts of the processor, where they are used to calculate addresses, table indices, increment and decrement operators, and similar operations.

d) ALU - pg. 329 - Section 10.1

An arithmetic logic unit (ALU) is a combinational digital electronic circuit that performs arithmetic and bitwise operations on integer binary numbers. An ALU is a fundamental building block of many types of computing circuits, including the central processing unit (CPU) of computers, FPUs, and graphics processing units (GPUs). A single CPU, FPU or GPU may contain multiple ALUs.

e) Comparators and **Multipliers**

A comparator is a device that compares two voltages or currents and outputs a digital signal indicating which is larger. It has two analog input terminals and one binary digital output.

A binary multiplier is an electronic circuit used in digital electronics, such as a computer, to multiply two binary numbers. It is built using binary adders.

3. Question(s)

VHDL description has two domains: a sequential domain and a concurrent domain. The sequential domain is represented by a process or subprogram that contains sequential statements. These statements are executed in the order in which they appear within the process or subprogram, as in programming languages. The concurrent domain is represented by an architecture that contains processes, concurrent procedure calls, concurrent signal assignments, and component instantiations. These statements are executed simultaneously.

4. Program Code

-- 4-1 Mux:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity Mux41 is
    port (
        a, b, c, d : in std_logic;
        sel : in std_logic_vector( 1 downto 0);
        z : out std_logic);
end Mux41;

architecture Behavioral of Mux41 is

begin
    process(a, b, c, d, sel)
        begin
            -- Determine output based on select input
            if (sel = "00") then
                z <= a;
            elsif(sel = "01") then
                z <= b;
            elsif(sel = "10") then
                z <= c;
            else
                z <= d;
            end if;
    end process;
end Behavioral;
```

--2-4 Decoder

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity decoder24 is
    port(
        a, b : in STD_LOGIC;
        c : out STD_LOGIC_VECTOR(3 downto 0));
end decoder24;

architecture Behavioral of decoder24 is

begin
```

```
    process(a, b)
            begin
            --Cycle available inputs
             if (b = '0' and a = '0') then
                    c <= "0001";
             elsif (b = '0' and a = '1') then
                    c <= "0010";
             elsif (b = '1' and a = '0') then
                    c <= "0100";
             else
                    c <= "1000";
             end if;
    end process;
end Behavioral;
```

-- 4 bit Adder

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Ripple_Adder is
Port (
    A : in STD_LOGIC_VECTOR (3 downto 0);
    B : in STD_LOGIC_VECTOR (3 downto 0);
    Cin : in STD_LOGIC;
    S : out STD_LOGIC_VECTOR (3 downto 0);
    Cout : out STD_LOGIC);
end Ripple_Adder;

architecture Behavioral of Ripple_Adder is

-- Full Adder VHDL Code Component Decalaration
component full_adder_vhdl_code
Port (
    A : in STD_LOGIC;
    B : in STD_LOGIC;
    Cin : in STD_LOGIC;
    S : out STD_LOGIC;
    Cout : out STD_LOGIC);
end component;

-- Carry declaration
signal c1,c2,c3: STD_LOGIC;

begin

-- Port Mapping Full Adder 4 times
FA1: full_adder_vhdl_code port map( A(0), B(0), Cin, S(0), c1);
FA2: full_adder_vhdl_code port map( A(1), B(1), c1, S(1), c2);
```

```
FA3: full_adder_vhdl_code port map( A(2), B(2), c2, S(2), c3);
FA4: full_adder_vhdl_code port map( A(3), B(3), c3, S(3), Cout);

end Behavioral;
```

-- Full Adder (required for 4 bit adder)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity full_adder_vhdl_code 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_vhdl_code;

architecture gate_level of full_adder_vhdl_code is
begin

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

end gate_level;
```

-- 2 bit Comparator:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity comparator2bit is
    port(
            a : in std_logic_vector(1 downto 0);
            b : in std_logic_vector(1 downto 0);
            equal : out std_logic;
            greater : out std_logic;
            less : out std_logic);
end comparator2bit;

architecture Behavioral of comparator2bit is
begin
    process(a, b)
            variable equal_tmp : std_logic;
            variable greater_tmp : std_logic;
            variable less_tmp : std_logic;
    begin
```

```
		equal_tmp := '0';
		greater_tmp := '0';
		less_tmp := '0';
		if a > b then
			greater_tmp := '1';
		elsif a = b then
			equal_tmp := '1';
		else
			less_tmp := '1';
		end if;
		greater <= greater_tmp;
		equal <= equal_tmp;
		less <= less_tmp;
	end process;
end Behavioral;
```

```
-- 2 bit ALU
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity alu is
	Port (
		inp_a : in signed(2 downto 0);
		inp_b : in signed(2 downto 0);
		sel : in STD_LOGIC_VECTOR (2 downto 0);
		out_alu : out signed(2 downto 0));
end alu;

architecture Behavioral of alu is
begin
	process(inp_a, inp_b, sel)
		begin
			case sel is
				when "00" =>
					--Addition
					out_alu<= inp_a + inp_b;
				when "01" =>
					--A + Not B + 1
					out_alu<= inp_a + (not inp_b) + 1;
				when "10" =>
					--A and B
					out_alu<= inp_a and inp_b;
				when "11" =>
					--A or B
					out_alu<= inp_a or inp_b;
				when others =>
					NULL;
```

```
                end case;
    end process;
end Behavioral;
```

-- 2 bit ALU User Constraint File

```
# Switches
NET "inp_b<0>" LOC = "F12";
NET "inp_b<1>" LOC = "G12";
NET "inp_a<0>" LOC = "H14";
NET "inp_a<1>" LOC = "H13";
NET "sel<0>" LOC = "K14";
NET "sel<1>" LOC = "K13";
# LEDs
NET "out_alu<0>" LOC = "K12";
NET "out_alu<1>" LOC = "P14";
```

-- 4 bit Signed Multiplier

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Multiplier is
    port(
        inp_a, inp_b : in std_logic_vector(3 downto 0);
        prod : out std_logic_vector(7 downto 0));
end Multiplier;

architecture Behavioral of Multiplier is
begin

    process(inp_a,inp_b)
    begin
        -- cast logic vectors to signed before multiplying and output to product
        prod <= signed(inp_a) * signed(inp_b);
    end process;
end Behavioral;
```

---

5. Test Bench

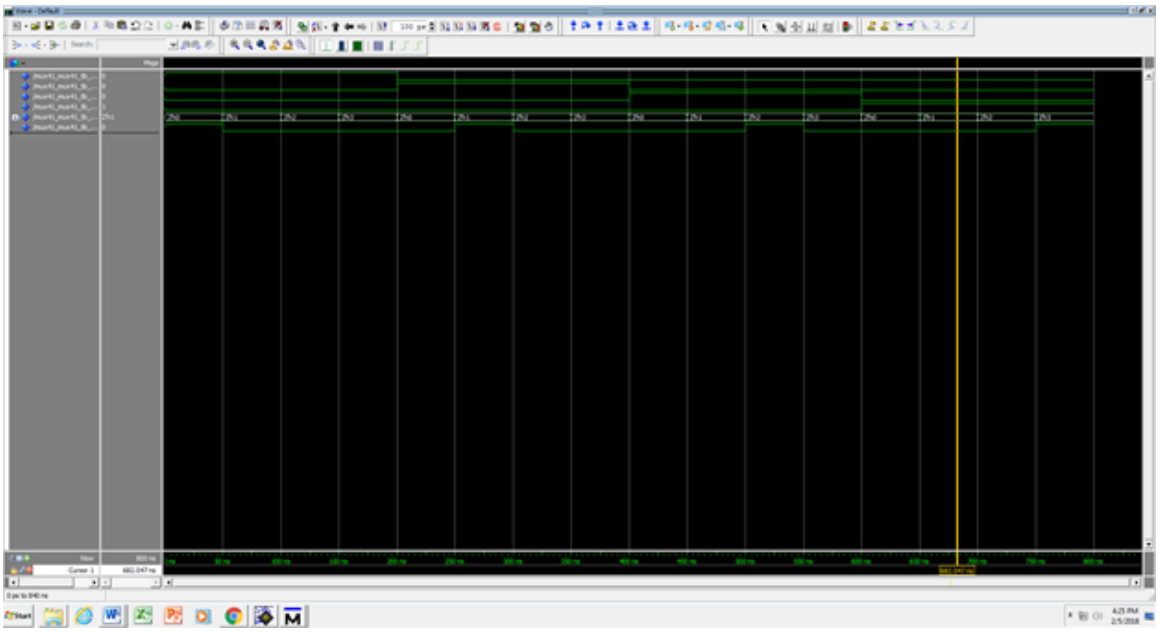
-- 4-1 Mux TB:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_arith.ALL;
USE ieee.numeric_std.ALL;

ENTITY mux41_Mux41_tb_vhd_tb IS
```

```
END mux41_Mux41_tb_vhd_tb;

ARCHITECTURE behavior OF mux41_Mux41_tb_vhd_tb IS
    COMPONENT mux41
    PORT(
            a : IN std_logic;
            b : IN std_logic;
            c : IN std_logic;
            d : IN std_logic;
            sel : IN std_logic_vector(1 downto 0);
            z : OUT std_logic);
    END COMPONENT;

    --Inputs
    SIGNAL a :  std_logic;
    SIGNAL b :  std_logic;
    SIGNAL c :  std_logic;
    SIGNAL d :  std_logic;
    SIGNAL sel :  std_logic_vector(1 downto 0);
    --Outputs
    SIGNAL z :  std_logic;

BEGIN
    --Instantiate unit under test(uut)
    uut: mux41 PORT MAP(
            a => a,
            b => b,
            c => c,
            d => d,
            sel => sel,
            z => z);

-- *** Test Bench - User Defined Section ***
  tb : PROCESS
  BEGIN
    --Assign 1st inputs and loop for each select case
    a <= '1';
    b <= '0';
    c <= '0';
    d <= '0';
    for i1 in integer range 0 to 3 loop
            sel <= conv_std_logic_vector(i1,2);
    wait for 50 ns;
    end loop;
    --Assign 2nd inputs and loop for each select case
    a <= '0';
    b <= '1';
    c <= '0';
```

```
        d <= '0';
        for i1 in integer range 0 to 3 loop
                sel <= conv_std_logic_vector(i1,2);
        wait for 50 ns; -- will wait forever
        end loop;
        --Assign 3rd inputs and loop for each select case
        a <= '0';
        b <= '0';
        c <= '1';
        d <= '0';
        for i1 in integer range 0 to 3 loop
                sel <= conv_std_logic_vector(i1,2);
        wait for 50 ns; -- will wait forever
        end loop;
        --Assign final inputs and loop for each select case
        a <= '0';
        b <= '0';
        c <= '0';
        d <= '1';
        for i1 in integer range 0 to 3 loop
                sel <= conv_std_logic_vector(i1,2);
        wait for 50 ns; -- will wait forever
        end loop;
        --Alert user when simulation is complete
        assert false report "End of Simulation" severity failure;
    END PROCESS;
    -- *** End Test Bench - User Defined Section ***
    END;
```

-- Mux Wave

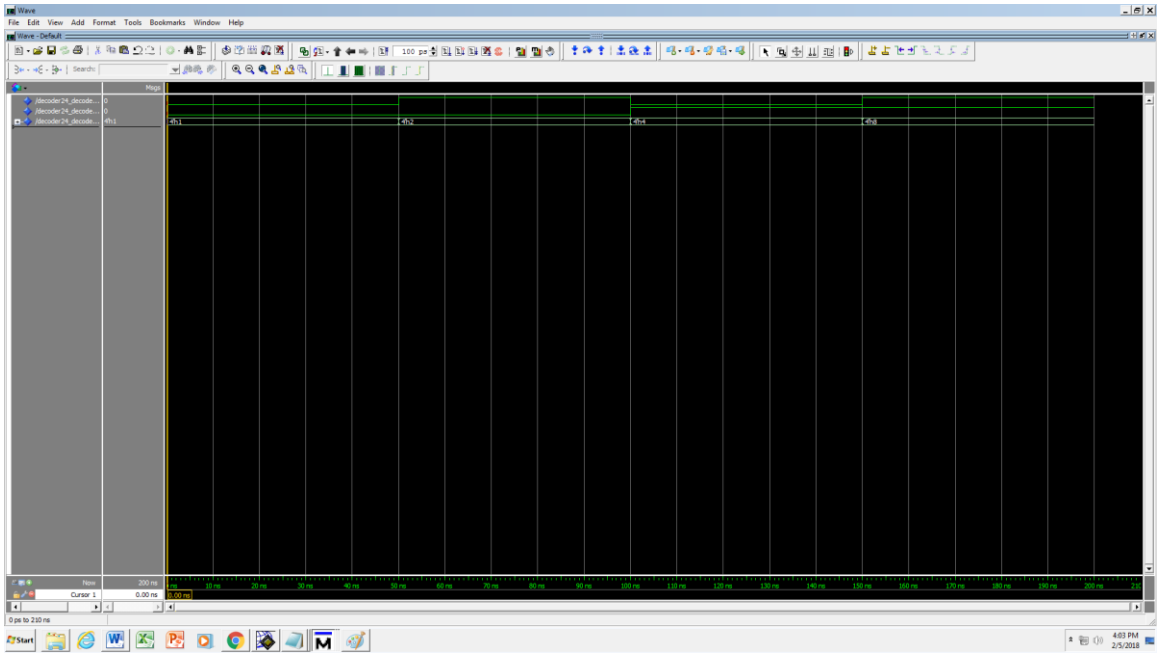


--2-4 Decoder Test Bench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY decoder24_decoder24_tb_vhd_tb IS
END decoder24_decoder24_tb_vhd_tb;

ARCHITECTURE behavior OF decoder24_decoder24_tb_vhd_tb IS
    COMPONENT decoder24
    PORT(
            a : IN std_logic;
            b : IN std_logic;
            c : OUT std_logic_vector(3 downto 0));
    END COMPONENT;

    --Inputs
    SIGNAL a :  std_logic;
    SIGNAL b :  std_logic;
    --Outputs
    SIGNAL c :  std_logic_vector(3 downto 0);

BEGIN
    uut: decoder24 PORT MAP(
            a => a,
            b => b,
            c => c);
```

```
-- *** Test Bench - User Defined Section ***
  tb : PROCESS
  BEGIN
     --Cycle test inputs from "00" to "11"
     a <= '0';
     b <= '0';
     wait for 50 ns;
     a <= '1';
     b <= '0';
     wait for 50 ns;
     a <= '0';
     b <= '1';
     wait for 50 ns;
     a <= '1';
     b <= '1';
     wait for 50 ns;
     --Notify when finished running test bench
     assert false report "End of Simulation" severity failure;
  END PROCESS;
-- *** End Test Bench - User Defined Section ***
END;
```

-- Decoder Wave



--4 bit Adder TB

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
use ieee.std_logic_arith.ALL;
```

```
ENTITY Tb_Ripple_Adder IS
END Tb_Ripple_Adder;

ARCHITECTURE behavior OF Tb_Ripple_Adder IS
-- Component Declaration for the Unit Under Test (UUT)

COMPONENT Ripple_Adder
PORT(
	A : IN std_logic_vector(3 downto 0);
	B : IN std_logic_vector(3 downto 0);
	Cin : IN std_logic;
	S : OUT std_logic_vector(3 downto 0);
	Cout : OUT std_logic);
END COMPONENT;

--Inputs
signal A : std_logic_vector(3 downto 0) := (others => '0');
signal B : std_logic_vector(3 downto 0) := (others => '0');
signal Cin : std_logic := '0';
--Outputs
signal S : std_logic_vector(3 downto 0);
signal Cout : std_logic;

BEGIN
-- Instantiate the Unit Under Test (UUT)
uut: Ripple_Adder PORT MAP (
	A => A,
	B => B,
	Cin => Cin,
	S => S,
	Cout => Cout);

-- Stimulus process
stim_proc: process
begin
	--Assign A input and loop for all possible B inputs
	A <= "0000";
	for i1 in integer range 0 to 15 loop
		B <= conv_std_logic_vector(i1,4);
		wait for 50 ns;
	end loop;
	--Assign A input and loop for all possible B inputs
	A <= "0001";
	for i1 in integer range 0 to 15 loop
		B <= conv_std_logic_vector(i1,4);
		wait for 50 ns;
	end loop;
	--Assign A input and loop for all possible B inputs
```

```
A <= "0010";
for i1 in integer range 0 to 15 loop
        B <= conv_std_logic_vector(i1,4);
        wait for 50 ns;
end loop;
--Assign A input and loop for all possible B inputs
A <= "0011";
for i1 in integer range 0 to 15 loop
        B <= conv_std_logic_vector(i1,4);
        wait for 50 ns;
end loop;
--Assign A input and loop for all possible B inputs
A <= "0100";
for i1 in integer range 0 to 15 loop
        B <= conv_std_logic_vector(i1,4);
        wait for 50 ns;
end loop;
--Assign A input and loop for all possible B inputs
A <= "0101";
for i1 in integer range 0 to 15 loop
        B <= conv_std_logic_vector(i1,4);
        wait for 50 ns;
end loop;
--Assign A input and loop for all possible B inputs
A <= "0110";
for i1 in integer range 0 to 15 loop
        B <= conv_std_logic_vector(i1,4);
        wait for 50 ns;
end loop;
--Assign A input and loop for all possible B inputs
A <= "0111";
for i1 in integer range 0 to 15 loop
        B <= conv_std_logic_vector(i1,4);
        wait for 50 ns;
end loop;
--Assign A input and loop for all possible B inputs
A <= "1000";
for i1 in integer range 0 to 15 loop
        B <= conv_std_logic_vector(i1,4);
        wait for 50 ns;
end loop;
--Assign A input and loop for all possible B inputs
A <= "1001";
for i1 in integer range 0 to 15 loop
        B <= conv_std_logic_vector(i1,4);
        wait for 50 ns;
end loop;
--Assign A input and loop for all possible B inputs
```

```
	A <= "1010";
	for i1 in integer range 0 to 15 loop
		B <= conv_std_logic_vector(i1,4);
		wait for 50 ns;
	end loop;
	--Assign A input and loop for all possible B inputs
	A <= "1011";
	for i1 in integer range 0 to 15 loop
		B <= conv_std_logic_vector(i1,4);
		wait for 50 ns;
	end loop;
	--Assign A input and loop for all possible B inputs
	A <= "1100";
	for i1 in integer range 0 to 15 loop
		B <= conv_std_logic_vector(i1,4);
		wait for 50 ns;
	end loop;
	--Assign A input and loop for all possible B inputs
	A <= "1101";
	for i1 in integer range 0 to 15 loop
		B <= conv_std_logic_vector(i1,4);
		wait for 50 ns;
	end loop;
	--Assign A input and loop for all possible B inputs
	A <= "1110";
	for i1 in integer range 0 to 15 loop
		B <= conv_std_logic_vector(i1,4);
		wait for 50 ns;
	end loop;
	--Assign A input and loop for all possible B inputs
	A <= "1111";
	for i1 in integer range 0 to 15 loop
		B <= conv_std_logic_vector(i1,4);
		wait for 50 ns;
	end loop;
	--Notify when finished running test bench
	assert false report "End of Simulation" severity failure;
end process;
END;
```

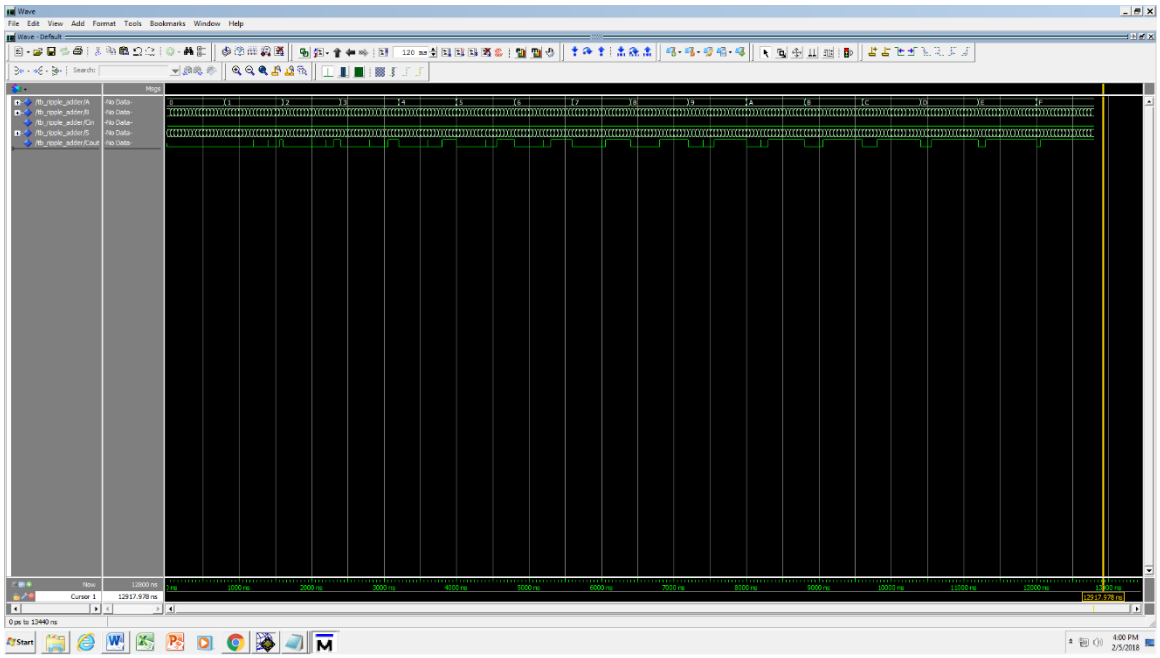
-- Full Adder TB

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY Testbench_full_adder IS
END Testbench_full_adder;

ARCHITECTURE behavior OF Testbench_full_adder IS
```

```
-- Component Declaration for the Unit Under Test (UUT)
COMPONENT full_adder_vhdl_code
PORT(
	A : IN std_logic;
	B : IN std_logic;
	Cin : IN std_logic;
	S : OUT std_logic;
	Cout : OUT std_logic);
 END COMPONENT;

--Inputs
signal A : std_logic := '0';
signal B : std_logic := '0';
signal Cin : std_logic := '0';
--Outputs
signal S : std_logic;
signal Cout : std_logic;

BEGIN
-- Instantiate the Unit Under Test (UUT)
uut: full_adder_vhdl_code PORT MAP (
	A => A,
	B => B,
	Cin => Cin,
	S => S,
	Cout => Cout);

-- Stimulus process
stim_proc: process
begin
	-- Simulate different inputs to determine if full adder works properly
	A <= '1';
	B <= '0';
	Cin <= '0';
	wait for 10 ns;

	A <= '0';
	B <= '1';
	Cin <= '0';
	wait for 10 ns;

	A <= '1';
	B <= '1';
	Cin <= '0';
	wait for 10 ns;

	A <= '0';
```

```
        B <= '0';
        Cin <= '1';
        wait for 10 ns;

        A <= '1';
        B <= '0';
        Cin <= '1';
        wait for 10 ns;

        A <= '0';
        B <= '1';
        Cin <= '1';
        wait for 10 ns;

        A <= '1';
        B <= '1';
        Cin <= '1';
        wait for 10 ns;
        --Notify when finished running test bench
        assert false report "End of Simulation" severity failure;

end process;
END;
```

-- 4 bit Adder Wave



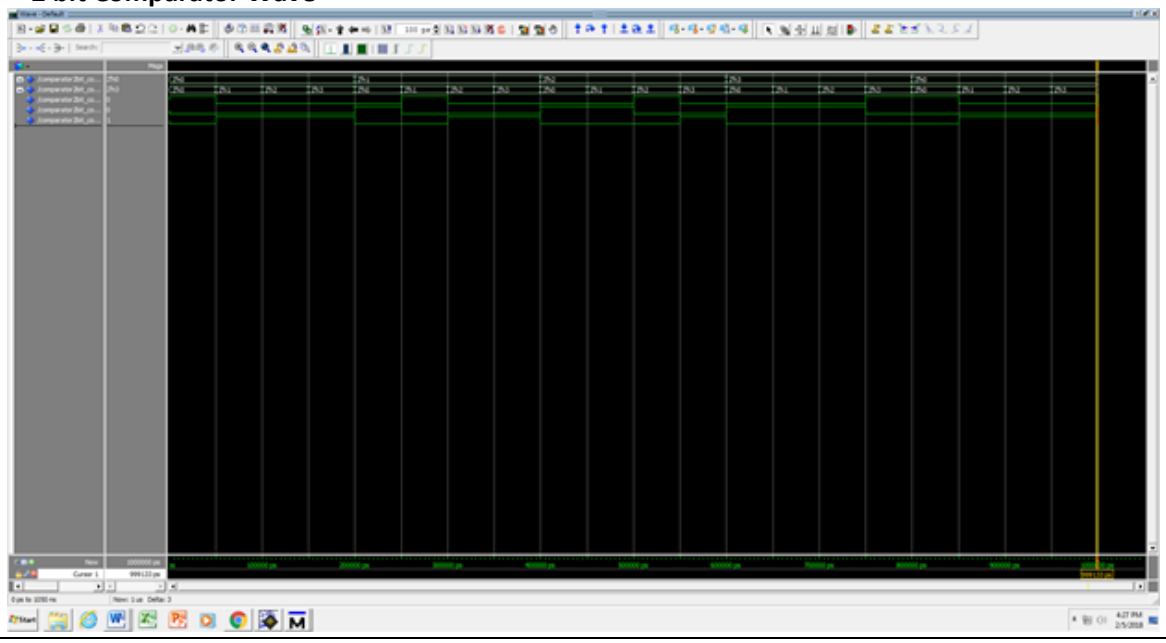
--2 bit Comparator TB:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_arith.ALL;
```

```
USE ieee.numeric_std.ALL;

ENTITY comparator2bit_comparator2bit_tb_vhd_tb IS
END comparator2bit_comparator2bit_tb_vhd_tb;

ARCHITECTURE behavior OF comparator2bit_comparator2bit_tb_vhd_tb IS
--Component Declaration for the Unit Under Test(UUT)
COMPONENT comparator2bit
PORT(
	a : IN std_logic_vector(1 downto 0);
	b : IN std_logic_vector(1 downto 0);
	equal : OUT std_logic;
	greater : OUT std_logic;
	less : OUT std_logic);
END COMPONENT;

--Inputs
SIGNAL a :  std_logic_vector(1 downto 0);
SIGNAL b :  std_logic_vector(1 downto 0);
--Outputs
SIGNAL equal :  std_logic;
SIGNAL greater :  std_logic;
SIGNAL less :  std_logic;

BEGIN
	uut: comparator2bit PORT MAP(
			a => a,
			b => b,
			equal => equal,
			greater => greater,
			less => less);

-- *** Test Bench - User Defined Section ***
tb : PROCESS
BEGIN
	--Assign A input and loop for possible B inputs
	a <= "00";
	for i1 in integer range 0 to 3 loop
			b <= conv_std_logic_vector(i1,2);
	wait for 50 ns;
	end loop;
	--Assign A input and loop for possible B inputs
	a <= "01";
	for i1 in integer range 0 to 3 loop
			b <= conv_std_logic_vector(i1,2);
	wait for 50 ns;
	end loop;
	--Assign A input and loop for possible B inputs
```

```
        a <= "10";
        for i1 in integer range 0 to 3 loop
                b <= conv_std_logic_vector(i1,2);
        wait for 50 ns;
        end loop;
        --Assign A input and loop for possible B inputs
        a <= "11";
        for i1 in integer range 0 to 3 loop
                b <= conv_std_logic_vector(i1,2);
        wait for 50 ns;
        end loop;
        --Notify when finished running test bench
        assert false report "End of Simulation" severity failure;
END PROCESS;
-- *** End Test Bench - User Defined Section ***
END;
```

-- 2 bit Comparator Wave



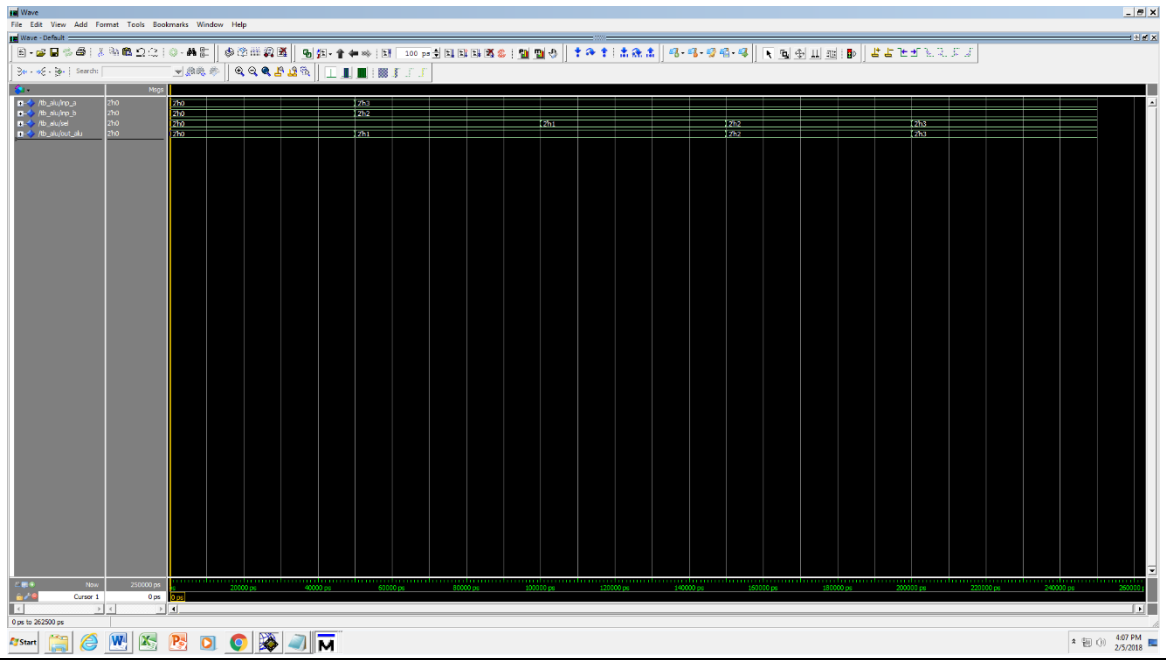
-- 2 bit ALU TB

```
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(2 downto 0);
      inp_b : IN signed(2 downto 0);
      sel : IN std_logic_vector(2 downto 0);
      out_alu : OUT signed(2 downto 0));
END COMPONENT;


-- Inputs
signal inp_a : signed(2 downto 0) := (others => '0');
signal inp_b : signed(2 downto 0) := (others => '0');
signal sel : std_logic_vector(2 downto 0) := (others => '0');

-- Outputs
signal out_alu : signed(2 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 50 ns.
wait for 50 ns;

-- set inputs and then cycle select cases to see different comparisons
inp_a <= "11";
inp_b <= "10";

sel <= "00";
     wait for 50 ns;
sel <= "01";
     wait for 50 ns;
sel <= "10";
     wait for 50 ns;
sel <= "11";
     wait for 50 ns;
--Notify when finished running test bench
     assert false report "End of Simulation" severity failure;
```

```
end process;
END;
```

-- 2 bit ALU Wave



-- Signed Multiplier TB

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY multiplier_multi_tb_vhd_tb IS
END multiplier_multi_tb_vhd_tb;

ARCHITECTURE behavior OF multiplier_multi_tb_vhd_tb IS

    COMPONENT multiplier
    PORT(
            inp_a : IN std_logic_vector(3 downto 0);
            inp_b : IN std_logic_vector(3 downto 0);
            prod : OUT std_logic_vector(7 downto 0));
    END COMPONENT;

    --Inputs
    SIGNAL inp_a :  std_logic_vector(3 downto 0);
    SIGNAL inp_b :  std_logic_vector(3 downto 0);
    --Outputs
    SIGNAL prod :  std_logic_vector(7 downto 0);

BEGIN

    uut: multiplier PORT MAP(
```

```
                inp_a => inp_a,
                inp_b => inp_b,
                prod => prod);

-- *** Test Bench - User Defined Section ***
  tb : PROCESS
  BEGIN
      -- set inputs to determine validity of multiplication output
      inp_a <= "0001";
      inp_b <= "0101";
      wait for 50 ns;
      -- test case for (-3*7)
      inp_a <= "1101";
      inp_b <= "0111";
      wait for 50 ns;
      -- test case for (6*5)
      inp_a <= "0110";
      inp_b <= "0101";
      wait for 50 ns;
      --Signal user when simulation is complete
     assert false report "End of Simulation" severity failure;
  END PROCESS;
-- *** End Test Bench - User Defined Section ***

END;
```