Assignment 3: Building an ALU.

ALU is a unit performing arithmetic and logic operations. VHDL codes provided in this assignment are to investigate gate-level construction of ALU and to verify the ALU behavior through simulation.

VHDL is a hardware description language. It is to explore design such that hardware schematic can be synthesized from it, a real hardware can be implemented based on it, and hardware behavior can be examined with simulation provided by many of its vendors, including by open source community.

Tasks

0a. Set up VHDL synthesis and simulation tools.
The assignment is prepared and tested with Xilinx ISE Design Suite 13.2, which can be downloaded from:
http://www.xilinx.com/support/download/index.htm.

The choice of synthesis and simulation tools is yours. However, the submitted assignment will be tested with Xilinx ISE Design Suite 13.2.

0b. Familiarize with VHDL tools and language itself. Synthesize and simulate CH04 project provided in the appendix.

The CH04 project is to implement a one-bit ALU, as shown in Fig. 4.17 of the class textbook, Computer Organization & Design by Patterson and Hennessy. The figure is re-illustrated here in Figure 1. The upper schematic (schematic a) is what is provided. The lower schematic (schematic b) is what you will work for your task 2. For schematic a, the one-bit ALU takes `a`, `b`, `CarryIn` as input, `Binvert` and `Operation` signals as control signals, and then provides `Result` and `CarryOut` as output.

1. Make an ALU for the most significant bit.
The provided BitALU (see Appendix) is for 31 least significant bits. It does not have overflow detection. The one-bit ALU for the most significant bit has to be able to detect overflow occurrence from either addition or subtraction.

a) When does overflow occur? Complete the truth table (Table 1), when a_m and b_m, are two most significant bits of two adding operands; $Binv_m$ is a signal controlling inversion of b_m; and sum_m is the result from a one-bit full adder at the most significant bit.
Using notation that overflow = 1 means that overflow occurs and overflow = 0 means otherwise.
Note: $Binv_m$ = 1 indicates that the operation is actually subtraction.

b) Write boolean expression of Overflow in terms of sums of products (SOP) of a_m, b_m, $Binv_m$, and sum_m.

c) Create a module detecting overflow. Synthesize, Simulate, and manually verify its functionality. (Your report should provide code, synthesized schematic, simulation waveform, and concluding hand verification.)

d) Integrate an overflow detection module into a one-bit ALU and create a one-bit ALU for MSB (according to schematic b of Figure 1). Synthesize, Simulate, and manually verify its functionality. (Your report should provide code, synthesized schematic, simulation waveform, and concluding hand verification.)

Note: in addition to overflow detection unit, a one-bit ALU for MSB has a set signal as an extra output. Don't forget to have it in your design.

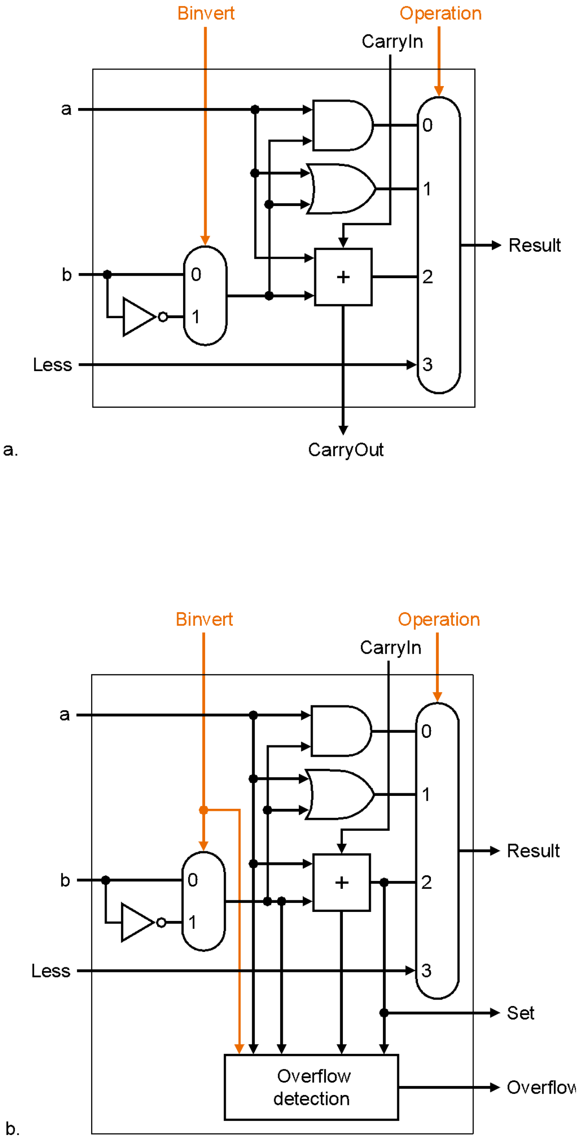


Figure 1: A one-bit ALU (from Fig. 4.17 of Patterson and Hennessy 2nd Ed. textbook)

	Binv_m	a_m	b_m	sum_m	Overflow
A + B	0	0	0	0	
	0	0	0	1	
	0	0	1	0	
	0	0	1	1	
	0	1	0	0	
	0	1	0	1	
	0	1	1	0	
	0	1	1	1	
A – B	1	0	0	0	
	1	0	0	1	
	1	0	1	0	
	1	0	1	1	
	1	1	0	0	
	1	1	0	1	
	1	1	1	0	
	1	1	1	1	

Table 1: Truth table of overflow given operation (Binv), operands (a and b), and result (sum) of the MSB.

2. Put 32 one-bit ALUs together to make a 32-bit ALU.

Put together 31 regular one-bit ALUs of the 31 least significant bits and a one-bit ALU for MSB of the MSB. Synthesize, Simulate, and manually verify its functionalities.
(Your report should provide code, synthesized schematic, simulation waveform, and concluding hand verification.)

a) What are Less and Set signals for?
You explain it and provide an example (or examples). Clues: it facilitates the set on less than instruction (`slt`).

b) Put 32 one-bit ALUs together as shown in Figure 2 (a copy of Fig. 4.18 P&H 2e.).

c) Add zero detection as shown in Figure 3.

B1. * Bonus*: Improve the ALU efficiency with Carry Look-Ahead.

B2. * Bonus*: Create a multiplication module and other related modules, as necessary.

B3. * Bonus*: Create a division module and other related modules, as necessary.

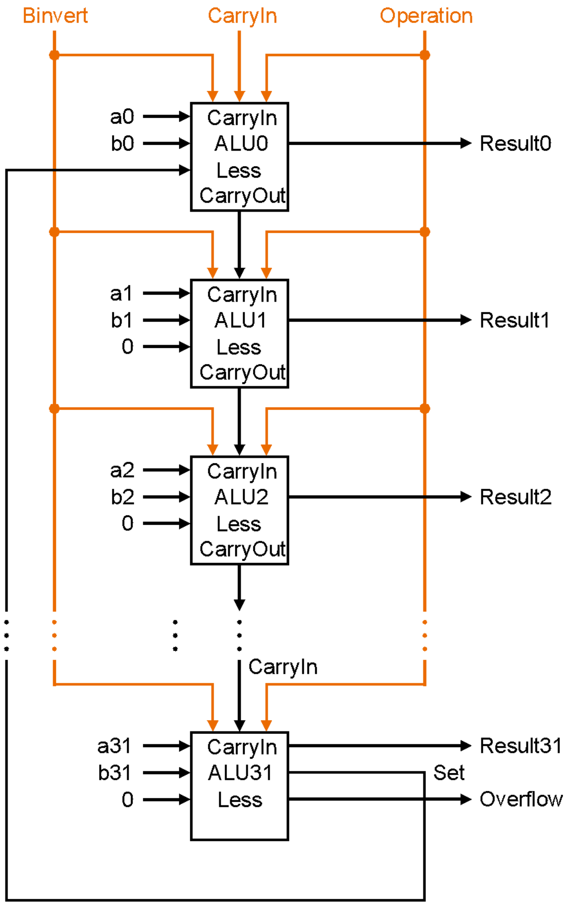


Figure 2: A 32-bit ALU constructed from 32 1-bit ALUs.

Tips:

Bits 1 to 30 can be modeled with the same pattern. You can write each one out like other single component. Alternatively, you can write a for loop to generate components for these 30 bits, as the following code segment:

```
    bits1to30: for i in 1 to 30 generate
        ibit: BitALU port map(
            a => a(i),
            b => b(i),
            cin => c(i),
            Binvert => Binvert,
            Lessin => '0',
            Qo => qr(i),
            cout => c(i+1),
            operation => Operation
        );
    end generate;
```

, where `a(i)`, `b(i)`, `c(i)`, …, `Operation` are corresponding input or internal signals.

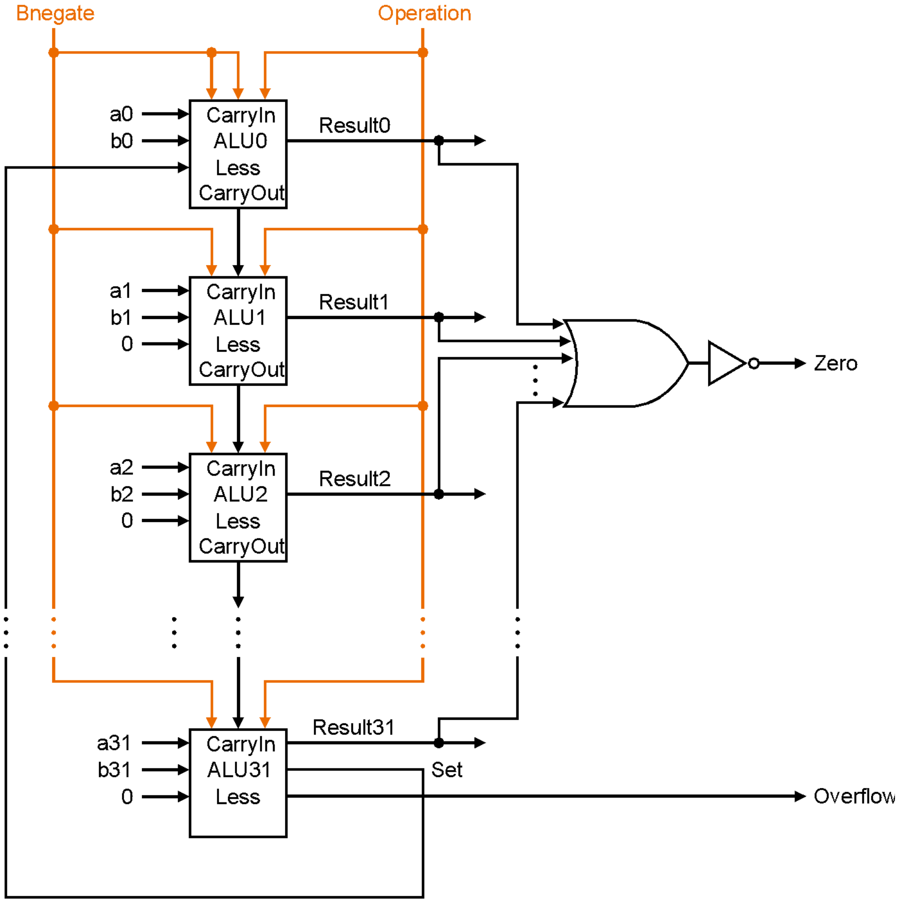


Figure 3: (Fig. 4.19 P&H 2e) ALU with zero detection

Appendix

Project CH04 is to build a one-bit ALU, as shown in Fig. 4.17 of Patterson and Hennessy textbook.

<u>Source file: **BitALU.vhd**</u>

```
----------------------------------------------------------------------------------
-- Company:  comp. engr. KKU.
-- Engineer: TK
--
-- Create Date:    08:07:26 08/18/2011
-- Design Name:
-- Module Name:    BitALU - Behavioral
-- Project Name:   CH04
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments: 
-- "Try not to become a man of success, but rather try to become a man of value."
-- Albert Einstein.
---------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity BitALU is
    Port ( a : in  STD_LOGIC;
           b : in  STD_LOGIC;
           cin : in  STD_LOGIC;
           Qo : out  STD_LOGIC;
           cout : out  STD_LOGIC;
           operation : in STD_LOGIC_VECTOR (1 downto 0);
           Binvert : in  STD_LOGIC;
                     Lessin : in STD_LOGIC);
end BitALU;

architecture Behavioral of BitALU is

      component mux4TO1
            port ( sel : in STD_LOGIC_VECTOR(1 downto 0); in0, in1, in2, in3 : in
STD_LOGIC; Q : out STD_LOGIC);
      end component;

      component fulladder
            port (a, b, ci: in STD_LOGIC; sum, co : out STD_LOGIC);
      end component;

      signal ANDout             : STD_LOGIC;
      signal ORout              : STD_LOGIC;
      signal ADDERout     : STD_LOGIC;
      signal bi                 : STD_LOGIC;

begin
            bi <= ((not Binvert) and b ) or (Binvert and not b);

            ANDout <= a and bi;
            ORout <= a or bi;
            FA: fulladder port map (a, bi, cin, ADDERout, cout);

            MX: mux4TO1 port map (sel => operation, in0 => ANDout, in1 => ORout,
                                                                in2 => ADDERout, in3 =>
Lessin, Q => Qo);

end Behavioral;
```

Source file: **fulladder.vhd**

```
----------------------------------------------------------------------------------
-- Company:  comp. engr. KKU.
-- Engineer: TK
-- 
-- Create Date:    08:19:31 08/18/2011 
-- Design Name: 
-- Module Name:    fulladder - Behavioral 
-- Project Name:   Ch04 
-- Target Devices: 
-- Tool versions: 
-- Description: 
--
-- Dependencies: 
--
-- Revision: 
-- Revision 0.01 - File Created
-- Additional Comments: 
-- "You must be the change you wish to see in the world." - Gandhi
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity fulladder is
    Port ( a : in  STD_LOGIC;
           b : in  STD_LOGIC;
           ci : in  STD_LOGIC;
           sum : out  STD_LOGIC;
           co : out  STD_LOGIC);
end fulladder;

architecture Behavioral of fulladder is
begin
      sum   <= (a xor b) xor ci;
      co    <= (a and b) or (b and ci) or (a and ci);
end Behavioral;
```

Source file: **mux4TO1.vhd**

```
----------------------------------------------------------------------------------
-- Company:  comp. engr. KKU.
-- Engineer: TK
-- 
```

```
-- Create Date:    08:33:14 08/18/2011
-- Design Name:
-- Module Name:    mux4TO1 - Behavioral
-- Project Name:    CH04
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
-- "Watch your thoughts; they become words. Watch your words; they become actions.
-- Watch your actions; they become habits. Watch your habits; they become character.
-- Watch your character; it becomes your destiny." -- Lao-Tze

----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity mux4TO1 is
    Port ( sel : in  STD_LOGIC_VECTOR (1 downto 0);
           in0 : in  STD_LOGIC;
           in1 : in  STD_LOGIC;
           in2 : in  STD_LOGIC;
           in3 : in  STD_LOGIC;
           Q : out  STD_LOGIC);
end mux4TO1;

architecture Behavioral of mux4TO1 is

begin
   process (sel, in0, in1, in2, in3) is
   begin
      case Sel is
         when "00"  => Q <= in0;
         when "01"  => Q <= in1;
         when "10"  => Q <= in2;
         when "11"  => Q <= in3;
         when others => Q <= '0';
      end case;
   end process;

end Behavioral;
```

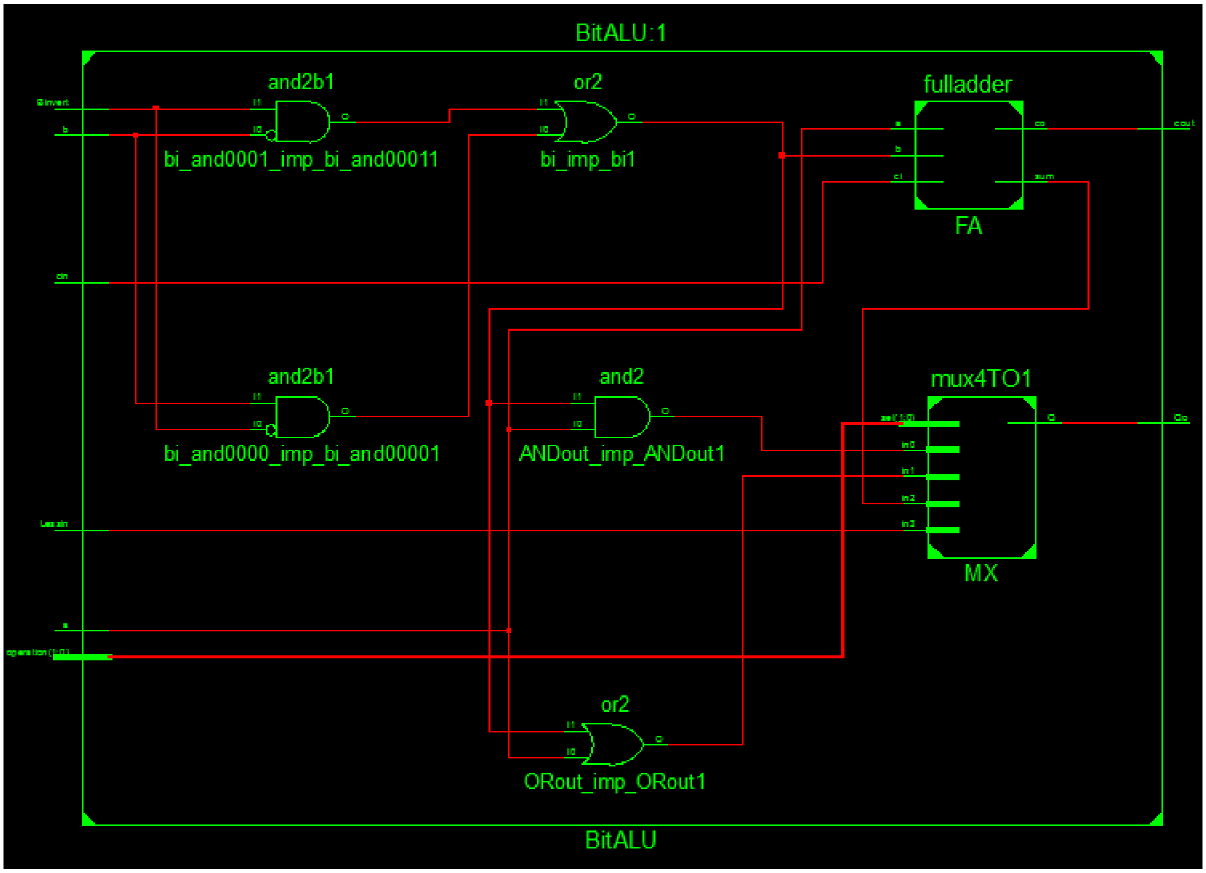
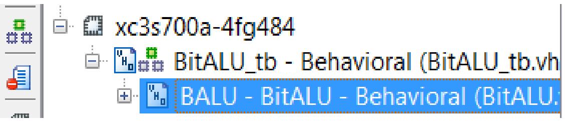


Figure 4: Synthesized RTL schematic of A 1-bit ALU.

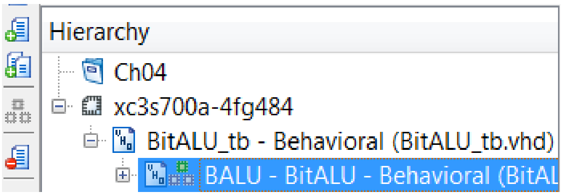
Remark:

Verify that BitALU is the top module. Otherwise, select BitALU and `Souce > Set as Top Module`.



xc3s700a-4fg484
BitALU_tb - Behavioral (BitALU_tb.vh
BALU - BitALU - Behavioral (BitALU.

Figure 5: A test bench (BitALU_tb) is selected as a top module. Synthesis may not work.



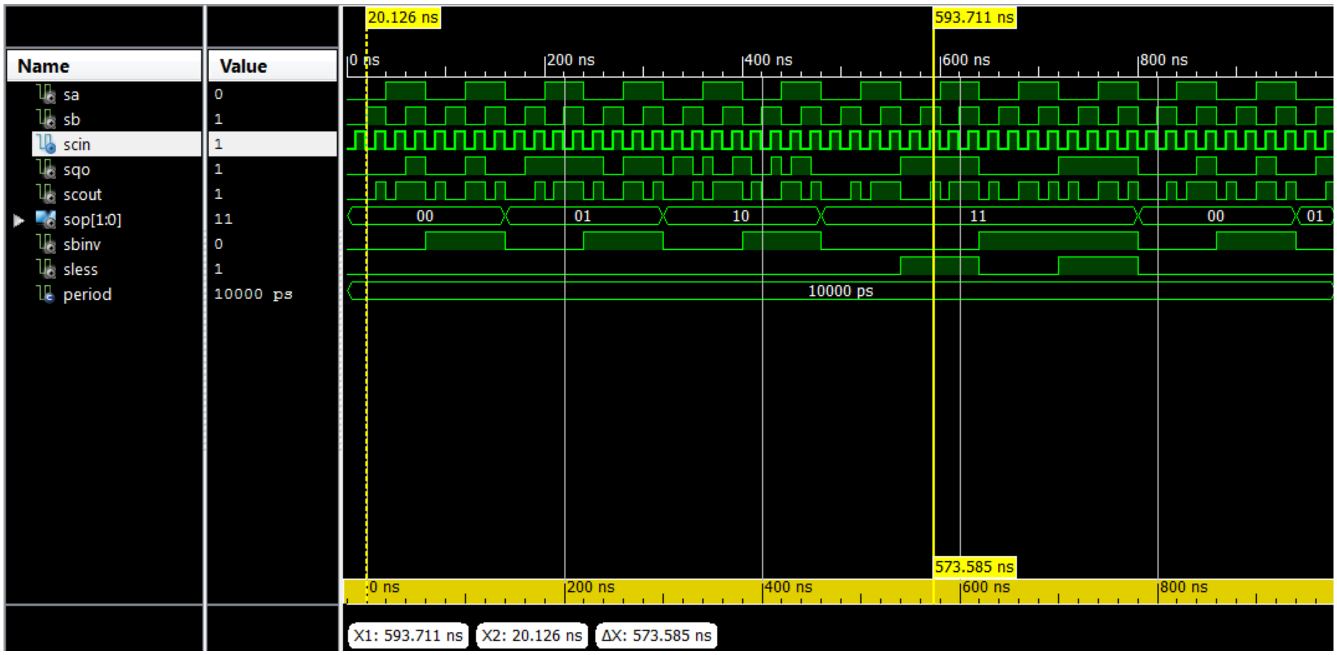
Hierarchy
Ch04
xc3s700a-4fg484
BitALU_tb - Behavioral (BitALU_tb.vhd)
BALU - BitALU - Behavioral (BitAL

Figure 6: A BitALU is selected as the top module.

Test bench file: **BitALU_tb.vhd**

```
----------------------------------------------------------------------------------
-- Company:  comp. engr. KKU.
-- Engineer:       TK
--
-- Create Date:    13:33:08 08/18/2011
-- Design Name:
-- Module Name:    BitALU_tb - Behavioral
-- Project Name:   CH04
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
-- "Success is not final. Failure is not fatal. Only courage to continue that counts." -
-- Winston Churchill
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity BitALU_tb is
end BitALU_tb;

architecture Behavioral of BitALU_tb is

    COMPONENT BitALU
    Port ( a : in  STD_LOGIC;
           b : in  STD_LOGIC;
           cin : in  STD_LOGIC;
           Qo : out  STD_LOGIC;
           cout : out  STD_LOGIC;
           operation : in STD_LOGIC_VECTOR (1 downto 0);
           Binvert : in  STD_LOGIC;
                     Lessin : in STD_LOGIC );
    END COMPONENT;

    SIGNAL sa             : std_logic := '0';
    SIGNAL sb             : std_logic := '0';
    SIGNAL scin     : std_logic := '0';
    SIGNAL sQo      : std_logic;
    SIGNAL scout    : std_logic;
    SIGNAL sop      : std_logic_vector (1 downto 0) := "00";
    SIGNAL sBinv   : std_logic := '0';
    SIGNAL sLess   : std_logic := '0';

    constant PERIOD : time := 10 ns;

begin

        BALU : BitALU
        PORT MAP (a => sa, b => sb, cin => scin,
                   Qo => sQo, cout => scout,
                   operation => sop,
                   Binvert => sBinv, Lessin => sLess);

        PROCESS    -- running a, b, cin
        BEGIN
            RUN_LOOP : LOOP
                   sa <= '0';
                   sb <= '0';
                   scin <= '0';
                WAIT FOR PERIOD;
                   sa <= '0';
                   sb <= '0';
                   scin <= '1';
                WAIT FOR PERIOD;
                   sa <= '0';
                   sb <= '1';
                   scin <= '0';
                WAIT FOR PERIOD;
                   sa <= '0';
                   sb <= '1';
                   scin <= '1';
                WAIT FOR PERIOD;
                   sa <= '1';
                   sb <= '0';
                   scin <= '0';
                WAIT FOR PERIOD;
                   sa <= '1';
                   sb <= '0';
                   scin <= '1';
                WAIT FOR PERIOD;
                   sa <= '1';
                   sb <= '1';
                   scin <= '0';
                WAIT FOR PERIOD;
                   sa <= '1';
                   sb <= '1';
                   scin <= '1';
                WAIT FOR PERIOD;
            END LOOP RUN_LOOP;
      END PROCESS;

             PROCESS      -- running Operation, Binv, Less
      BEGIN
                                sop          <= "00";
                                sBinv        <= '0';
```

```
                                 sLess        <= '0';
                                 WAIT FOR 8*PERIOD;
                                 sop          <= "00";
                                 sBinv        <= '1';
                                 sLess        <= '0';
                                 WAIT FOR 8*PERIOD;
                                 sop          <= "01";
                                 sBinv        <= '0';
                                 sLess        <= '0';
                                 WAIT FOR 8*PERIOD;
                                 sop          <= "01";
                                 sBinv        <= '1';
                                 sLess        <= '0';
                                 WAIT FOR 8*PERIOD;
                                 sop          <= "10";
                                 sBinv        <= '0';
                                 sLess        <= '0';
                                 WAIT FOR 8*PERIOD;
                                 sop          <= "10";
                                 sBinv        <= '1';
                                 sLess        <= '0';
                                 WAIT FOR 8*PERIOD;
                                 sop          <= "11";
                                 sBinv        <= '0';
                                 sLess        <= '0';
                                 WAIT FOR 8*PERIOD;
                                 sop          <= "11";
                                 sBinv        <= '0';
                                 sLess        <= '1';
                                 WAIT FOR 8*PERIOD;
                                 sop          <= "11";
                                 sBinv        <= '1';
                                 sLess        <= '0';
                                 WAIT FOR 8*PERIOD;
                                 sop          <= "11";
                                 sBinv        <= '1';
                                 sLess        <= '1';
                                 WAIT FOR 8*PERIOD;

             END PROCESS;

end Behavioral;
```



Name	Value
sa	0
sb	1
scin	1
sqo	1
scout	1
sop[1:0]	11
sbinv	0
sless	1
period	10000 ps

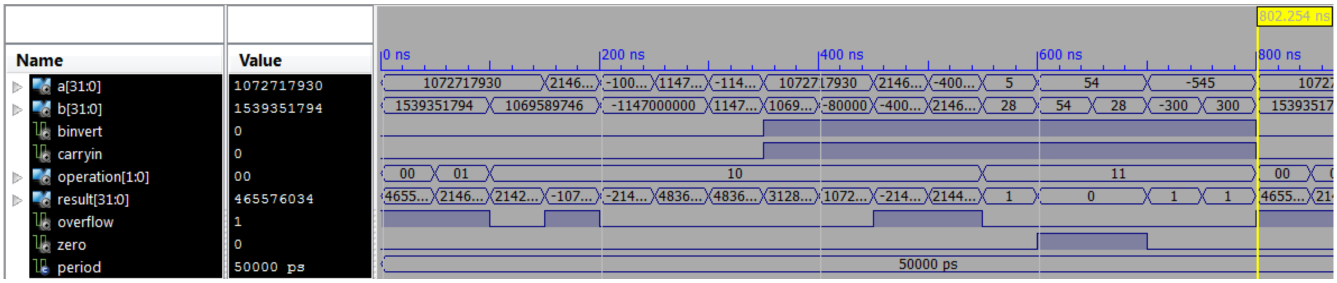
Figure 7: Simulation results (Simulation Run Time is set to 1000 ns).

Test bench: **WordALU_tb.vhd**

```
----------------------------------------------------------------------------------
-- Company:         comp. engr. KKU.
-- Engineer:               TK
--
-- Create Date:    09:55:02 08/22/2011
-- Design Name:
-- Module Name:    WordALU_tb - Behavioral
-- Project Name:    Ch04
-- Revision 0.01 - File Created
-- Additional Comments:
--   If you think you can do a thing or think you can't do a thing, you're right.
--   Henry Ford

----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity WordALU_tb is
end WordALU_tb;
```

```
architecture Behavioral of WordALU_tb is

component WordALU is
    Port ( a, b : in  STD_LOGIC_VECTOR (31 downto 0);
           Binvert, CarryIn : in  STD_LOGIC;
           Operation : in  STD_LOGIC_VECTOR (1 downto 0);
           Result : out  STD_LOGIC_VECTOR (31 downto 0);
           Overflow, Zero : out  STD_LOGIC);
end component;

      signal a, b : STD_LOGIC_VECTOR (31 downto 0);
      signal Binvert, CarryIn : STD_LOGIC;
      signal Operation : STD_LOGIC_VECTOR (1 downto 0);
      signal Result : STD_LOGIC_VECTOR (31 downto 0);
      signal Overflow, Zero : STD_LOGIC;

   constant PERIOD : time := 50 ns;

begin

      walu: WordALU
      port map (
             a => a,
             b => b,
             Binvert => Binvert,
             CarryIn => CarryIn,
             Operation => Operation,
             Result => Result,
             Overflow => Overflow,
             Zero => Zero);

      PROCESS
   BEGIN

             Binvert <= '0';
             CarryIn <= '0';


      -- AND --
             Operation <= "00";  -- 00: AND, 01: OR, 10: ADDITION, 11: Less (slt)
             a <= "00111111111100000110000001101010";
             b <= "01011011110000001010010011110010";
             -- expect:  "00011011110000000010000001100010" = '0x1bc02062'
             WAIT FOR PERIOD;

      -- OR --
             Operation <= "01";  -- 00: AND, 01: OR, 10: ADDITION, 11: Less (slt)
             a <= "00111111111100000110000001101010";
             b <= "01011011110000001010010011110010";
             -- expect:  "01111111111100001110010011111010" = '0x7ff0e4fa'
             WAIT FOR PERIOD;

      -- ADDITION --
             -- A + B; A > 0, B > 0, no overflow
```

```
            Binvert <= '0';          -- A + B
            CarryIn <= '0';
            Operation <= "10";  -- 00: AND, 01: OR, 10: ADDITION, 11: Less (slt)

            a <= "00111111111100000110000001101010";
            b <= "00111111110000001010010011110010";
            -- expect:   "01111111101100010000010101011100" = 0x7fb1055c

            WAIT FOR PERIOD;

            -- A + B; A > 0, B > 0, overflow

            Binvert <= '0';          -- A + B
            CarryIn <= '0';
            Operation <= "10";  -- 00: AND, 01: OR, 10: ADDITION, 11: Less (slt)

            a <= "01111111111100000110000001101010";
            b <= "00111111110000001010010011110010";
            -- expect:   "10111111101100010000010101011100" = 0xbfb1055c

            WAIT FOR PERIOD;

            -- A + B; A < 0, B < 0, no overflow

            Binvert <= '0';          -- A + B
            CarryIn <= '0';
            Operation <= "10";  -- 00: AND, 01: OR, 10: ADDITION, 11: Less (slt)

            a <= "11000100011001010011011000000000"; -- = 0xc4653600 = -1000000000
            b <= "10111011101000100010101101000000"; -- = 0xbba22b40 = -1147000000
            -- expect: "d10000000000001110110000101000000"; -- = 0x80076140 = -2147000000

            WAIT FOR PERIOD;

            -- A + B; A > 0, B < 0

            Binvert <= '0';          -- A + B
            CarryIn <= '0';
            Operation <= "10";  -- 00: AND, 01: OR, 10: ADDITION, 11: Less (slt)

            a <= "01000100011001010011011000000000"; -- = 0x44653600 =  1147483648
            b <= "10111011101000100010101101000000"; -- = 0xbba22b40 = -1147000000
            -- expect: "?00000000000001110110000101000000"; -- = 0x00076140 = 483648

            WAIT FOR PERIOD;

            -- A + B; A < 0, B > 0

            Binvert <= '0';          -- A + B
            CarryIn <= '0';
            Operation <= "10";  -- 00: AND, 01: OR, 10: ADDITION, 11: Less (slt)

            a <= "10111011101000100010101101000000"; -- = 0xbba22b40 = -1147000000
            b <= "01000100011001010011011000000000"; -- = 0x44653600 =  1147483648
            -- expect: "?00000000000001110110000101000000"; -- = 0x00076140 = 483648
            WAIT FOR PERIOD;

            -- A - B; A > 0, B > 0

            Binvert <= '1';           -- A - B
            CarryIn <= '1';
            Operation <= "10";  -- 00: AND, 01: OR, 10: ADDITION, 11: Less (slt)

            a <= "00111111111100000110000001101010"; -- = 0x3ff0606a  = 1072717930
            b <= "00111111110000001010010011110010"; -- = 0x3fc0a4f2  = 1069589746
      -- expect:   "00000000001011111011101101111000" = 0x002fbb78     =    3128184

            WAIT FOR PERIOD;

            -- A - B; A > 0, B < 0, no overflow

            Binvert <= '1';           -- A - B
            CarryIn <= '1';
            Operation <= "10";  -- 00: AND, 01: OR, 10: ADDITION, 11: Less (slt)

            a <= "00111111111100000110000001101010"; -- = 0x3ff0606a  = 1072717930
            b <= "11111111111111101100011110000000"; -- = 0xfffec780  =     -80000
      -- expect:   "00111111111100011001100011101010" = 0x3ff198ea     = 1072797930

            WAIT FOR PERIOD;

            -- A - B; A > 0, B < 0, overflow

            Binvert <= '1';           -- A - B
            CarryIn <= '1';
            Operation <= "10";  -- 00: AND, 01: OR, 10: ADDITION, 11: Less (slt)

            a <= "01111111111100000110000001101010"; -- = 0x7ff0606a  = 2146459754
            b <= "11111111110000101111011100000000"; -- = 0xffc2f700  =   -4000000
      -- expect:  "10000000001011010110100101101010"; = 0x802d696a     ; 2150459754

            WAIT FOR PERIOD;

            -- A - B; A < 0, B < 0

                  -- <input code for this test case>

            -- A - B; A < 0, B > 0, no overflow

                  -- <input code for this test case>

            -- A - B; A < 0, B > 0, overflow

            Binvert <= '1';           -- A - B
            CarryIn <= '1';
            Operation <= "10";  -- 00: AND, 01: OR, 10: ADDITION, 11: Less (slt)

            a <= "11111111110000101111011100000000"; -- = 0xffc2f700  =    -4000000
            b <= "01111111111100000110000001101010"; -- = 0x7ff0606a  =     2146459754
      -- expect: "?01111111110100101001011010010110"; = 0x7fd29696     ; -2150459754
            WAIT FOR PERIOD;

      -- LESS (slt) --
            Operation <= "11";  -- 00: AND, 01: OR, 10: ADDITION, 11: Less (slt)

            -- a < b
            a      <= "00000000000000000000000000000101"; -- 5
            b      <= "00000000000000000000000000011100"; -- 28
            WAIT FOR PERIOD;

            -- a = b
            a      <= "00000000000000000000000000110110"; -- 54
            b      <= "00000000000000000000000000110110"; -- 54
            WAIT FOR PERIOD;

            -- a > b
            a      <= "00000000000000000000000000110110"; -- 54
            b      <= "00000000000000000000000000011100"; -- 28
            WAIT FOR PERIOD;

            -- a < b, a < 0, b < 0
            a      <= "11111111111111111111110111011111"; -- -545
            b      <= "11111111111111111111111011010100"; -- -300
            WAIT FOR PERIOD;

            -- a < b, a < 0, b > 0
            a      <= "11111111111111111111110111011111"; -- -545
            b  <= "00000000000000000000000100101100"; -- 300
            WAIT FOR PERIOD;

   END PROCESS;
end Behavioral;
```



Name	Value
a[31:0]	1072717930
b[31:0]	1539351794
binvert	0
carryin	0
operation[1:0]	00
result[31:0]	465576034
overflow	1
zero	0
period	50000 ps

Illustration 1: Waveform captured from simulation with the given test bench (WordALU_tb.vhd)