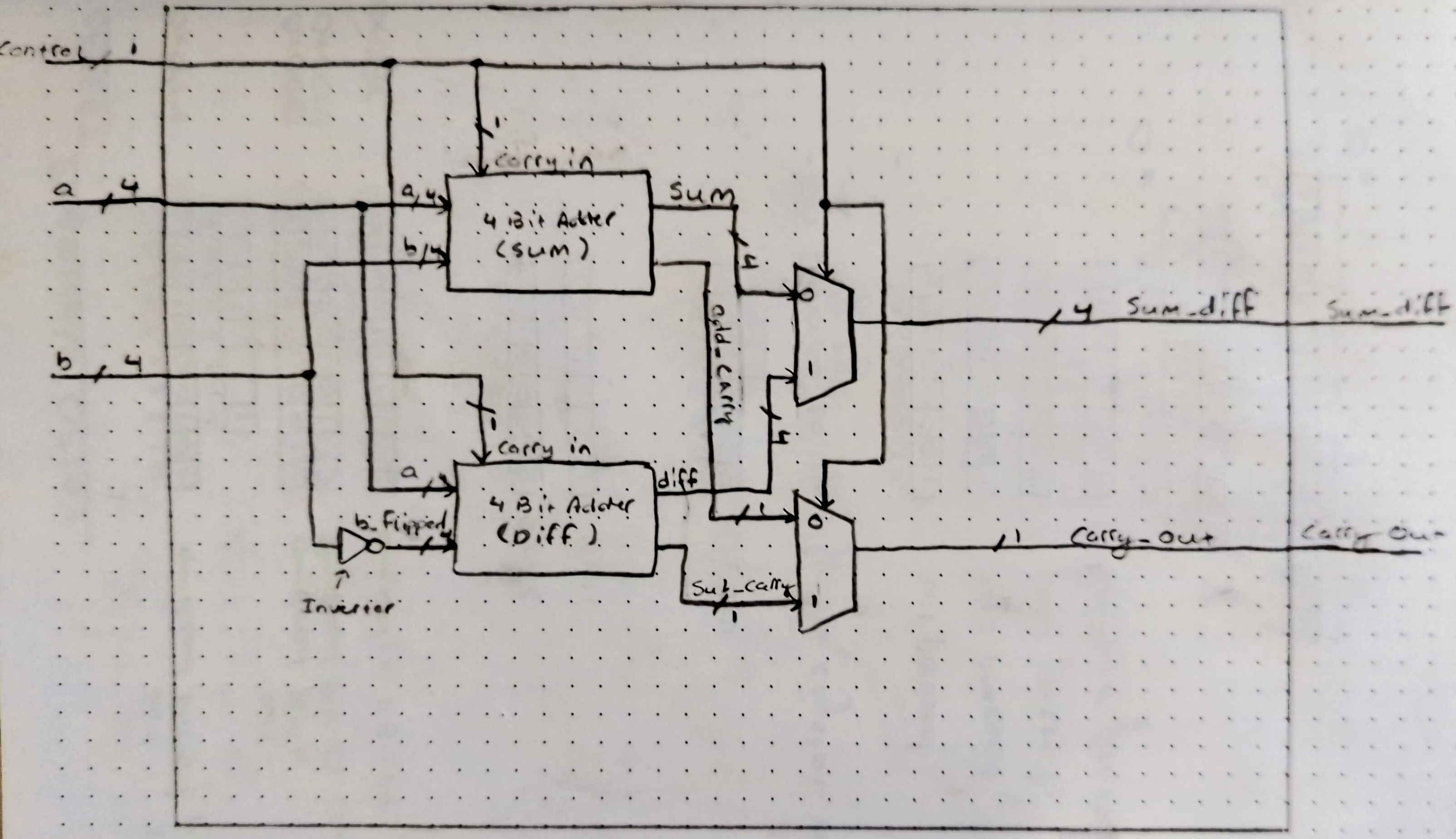
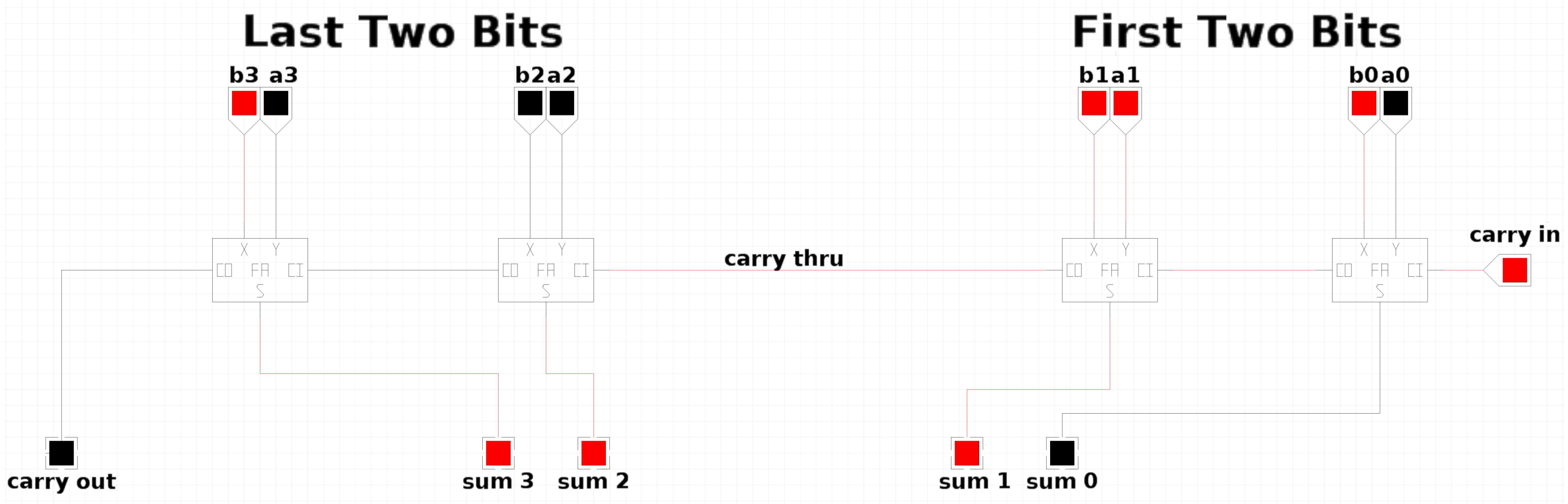


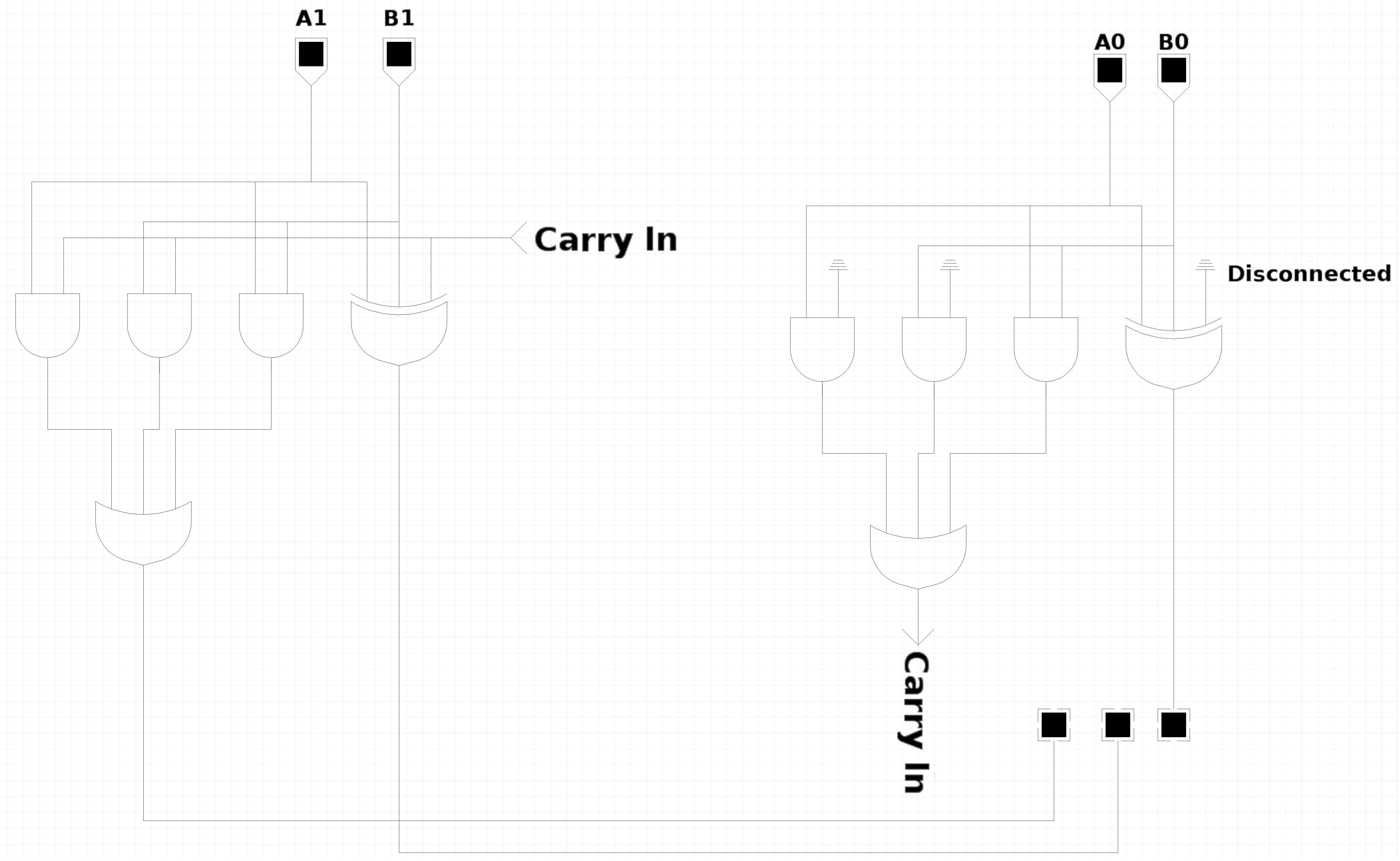
four bit adder subtracter

Four Bit Adder

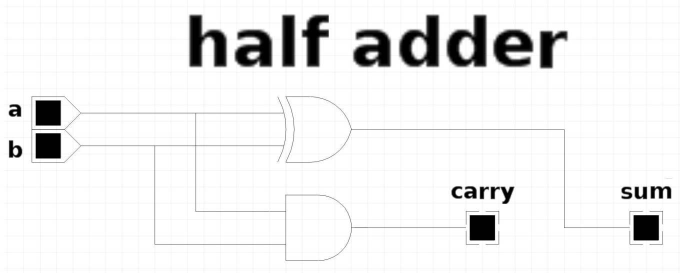
composed of two 2 bit adders

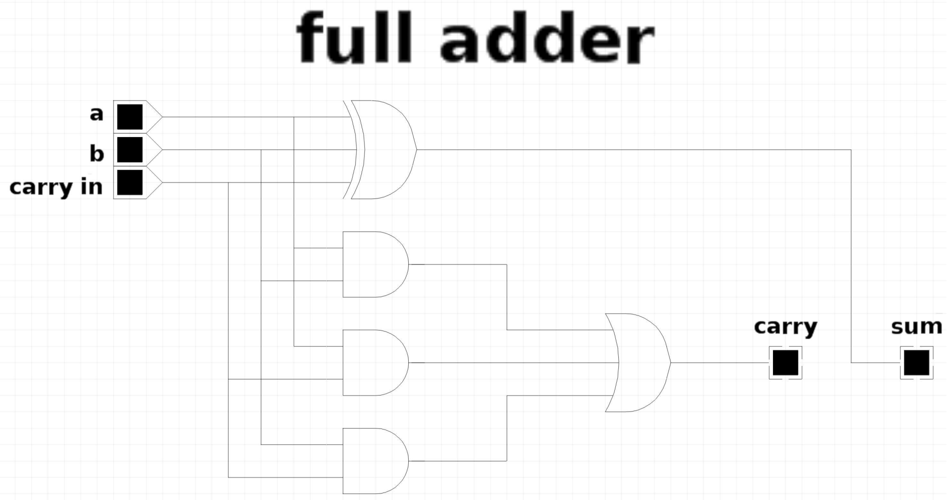


Two Bit Ripple Carry Adder



C:\Users\james\OneDrive\Documents\NES Clone Project\NES_Clone\Cedar Logic Files\ALU\half adder.cdl





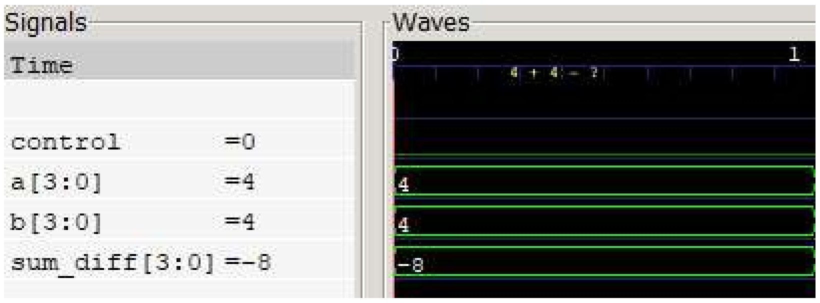
```vhdl
-- VHDL 8 bit processor Project
-- Four Bit Binary Adder Subtractor
-- James Hicks Sept 29 2023

library ieee;
use ieee.std_logic_1164.all;

entity four_bit_adder_subtractor is
    port (
            control           : in std_logic;
            a, b              : in std_logic_vector (3 downto 0);
            carry_out         : out std_logic;
            sum_diff          : out std_logic_vector (3 downto 0)
         );
end four_bit_adder_subtractor;

-- The control is essentially a carry in
-- With the added caviet that in addition to being added
-- to the other inputs, it also decides which output to take

architecture my_arch of four_bit_adder_subtractor is

    signal add_carry, sub_carry : std_logic;
    signal sum, diff : std_logic_vector (3 downto 0);
    signal b_flipped : std_logic_vector (3 downto 0);

begin

    b_flipped <= (not b(3)) & (not b(2)) & (not b(1)) & (not b(0));

    add : entity work.four_bit_adder(ripple_carry)
        port map (
                    carry_in => control,
                    a => a,
                    b => b,
                    carry_out => add_carry,
                    sum => sum
                 );
    subtract : entity work.four_bit_adder(ripple_carry)
        port map (
                    carry_in => control,
                    a => a,
                    b => b_flipped,
                    carry_out => sub_carry,
                    sum => diff
                 );

with control select sum_diff <=
    sum when '0',
    diff when '1',
    "XXXX" when others;

with control select carry_out <=
    add_carry when '0',
    sub_carry when '1',
    'X' when others;

--with control select sum_diff <=
--    sum when '0',
--    diff when '1',
--    "XXXX" when others;
--
--with control select carry_out <=
--    add_carry when '0',
--    '0' when '1',
--    'X' when others;

```

```vhdl
end my_arch;
```

```vhdl
-- VHDL 8 bit processor Project
-- Four Bit Binary Adder Subtractor Test Bench
-- James Hicks Sept 29 2023

-- The bit structure is as follows
-- (control bit) & (a input) & (b input)
-- Test bits out simply represents the sum or difference of inputs

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity four_bit_adder_subtractor_tb is
end four_bit_adder_subtractor_tb;

architecture tb_architecture of four_bit_adder_subtractor_tb is

-- 9 inputs (2 4-bit numbers + control) 5 outputs (a 4 bit number + carry)
    signal control     : std_logic;
    signal a,b         : std_logic_vector (3 downto 0);
    signal carry_out   : std_logic;
    signal sum_diff    : std_logic_vector (3 downto 0);

begin
    DUT : entity work.four_bit_adder_subtractor(my_arch)
        port map (
                        control => control,
                        a => a,
                        b => b,
                        carry_out => carry_out,
                        sum_diff => sum_diff
                  );
    process begin
        -- test 4 + 4 = 8 (out of range for signed 4 bit number)
        control <= '0';
        a <= "0100";
        b <= "0100";
        wait for 1 ns;

        -- test for -4 + -5 = -9 (out of range for signed 4 bit number)
        control <= '0';
        a <= "1100";
        b <= "1011";
        wait for 1 ns;

        -- test adding positive and negative extremes 7 + -8 = -1
        control <= '0';
        a <= "0111";
        b <= "1000";
        wait for 1 ns;

        -- test for subtraction overflow 3 - -6 = 9 (out of range)
        control <= '1';
        a <= "0101";
        b <= "1010";
        wait for 1 ns;

        -- test for subtraction underflow -7 - 7 (out of range)
        control <= '1';
        a <= "1111";
        b <= "0111";
        wait for 1 ns;

        -- test for 0 - 1 = -1
        control <= '1';
        a <= "0000";
        b <= "0001";
        wait for 1 ns;

        -- test for 1 - -1 = 0
        control <= '1';
        a <= "1111";
        b <= "1111";
        wait for 1 ns;

        assert false report "End of Test";
        wait;

    end process;

end tb_architecture;
```

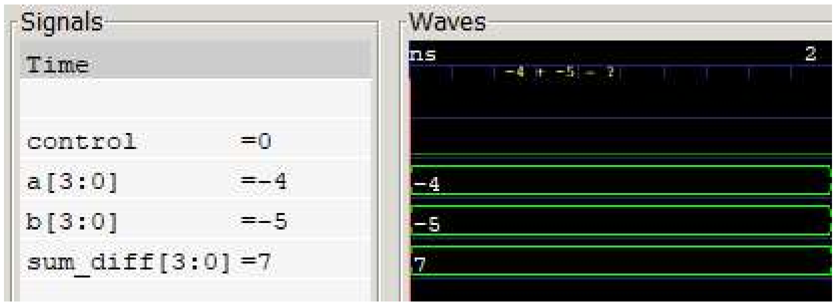
SIMULATION

Overflow Addition : 4 + 4 = ?



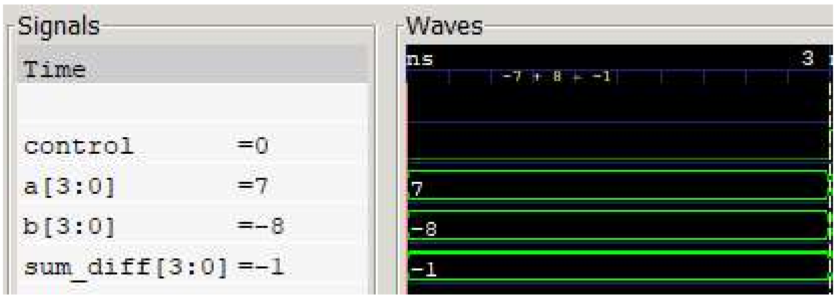
```
Time

control        =0
a[3:0]         =4
b[3:0]         =4
sum_diff[3:0]  =-8
```

Underflow addition: -4 + -5 = ?



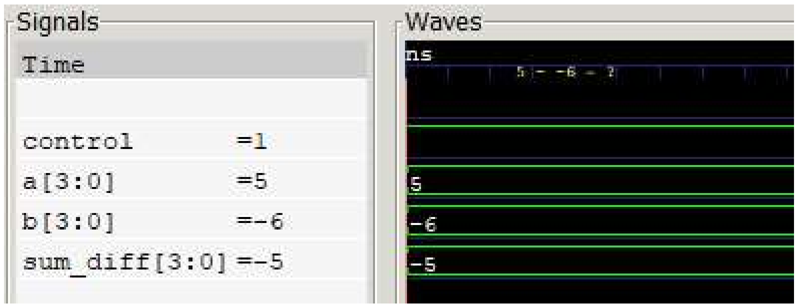
```
Time

control        =0
a[3:0]         =-4
b[3:0]         =-5
sum_diff[3:0]  =7
```

Adding extremes: 7 + -8 = -1



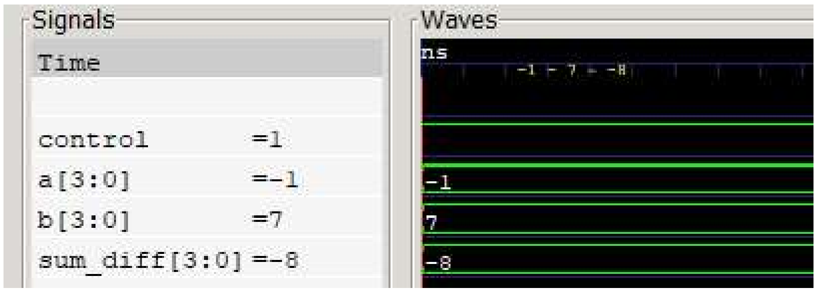
```
Time

control        =0
a[3:0]         =7
b[3:0]         =-8
sum_diff[3:0]  =-1
```

Overflow subtraction: 5 - -6 = ?



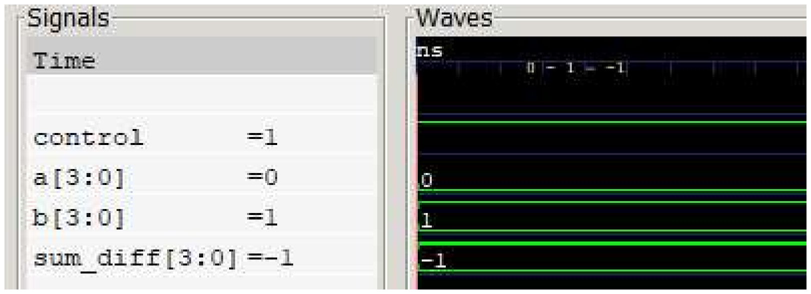
```
Time

control        =1
a[3:0]         =5
b[3:0]         =-6
sum_diff[3:0]  =-5
```

lowest possible outcome subtraction: -1 - 7 = -8



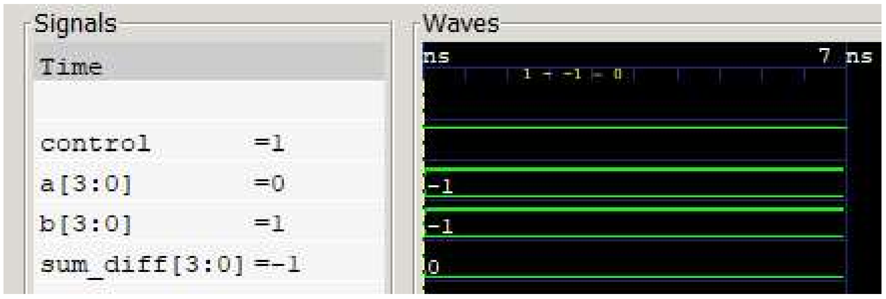
```
Signals                          Waves
Time                             ns
                                      -1 - 7 = -8
control        =1
a[3:0]         =-1               -1
b[3:0]         =7                7
sum_diff[3:0]  =-8               -8
```

0 - 1 = -1



```
Signals                          Waves
Time                             ns
                                      0 - 1 = -1
control        =1
a[3:0]         =0                0
b[3:0]         =1                1
sum_diff[3:0]  =-1               -1
```

-1 - -1 = 0



```
Signals                          Waves
Time                             ns                              7 ns
                                      1 - -1 = 0
control        =1
a[3:0]         =0                -1
b[3:0]         =1                -1
sum_diff[3:0]  =-1               0
```

```vhdl
-- VHDL 8 bit processor Project
-- Four Bit Adder
-- James Hicks Sept 23 2023

library ieee;
use ieee.std_logic_1164.all;

entity four_bit_adder is
    port (
            carry_in : in std_logic;
            a, b : in std_logic_vector (3 downto 0);
            carry_out : out std_logic;
            sum : out std_logic_vector (3 downto 0)
        );
end four_bit_adder;

architecture ripple_carry of four_bit_adder is

signal carry_through : std_logic;

begin
    first_two_bits : entity work.two_bit_adder(ripple_carry)
        port map (
                    carry_in => carry_in,
                    a => a (1 downto 0),
                    b => b (1 downto 0),
                    carry_out => carry_through,
                    sum => sum (1 downto 0)
                );
    last_two_bits : entity work.two_bit_adder(ripple_carry)
        port map (
                    carry_in => carry_through,
                    a => a (3 downto 2),
                    b => b (3 downto 2),
                    carry_out => carry_out,
                    sum => sum (3 downto 2)
                );

end ripple_carry;
```

```vhdl
-- VHDL 8 bit processor Project
-- Four Bit Adder Test Bench
-- James Hicks Sept 23 2023

-- The structure of test bits in is as follows
-- (carry in bit) & (a input) & (b input)
-- Test bits out simply represents the sum of the two input numbers

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity four_bit_adder_tb is
end four_bit_adder_tb;

architecture tb_architecture of four_bit_adder_tb is

-- 9 inputs (two 4 bit numbers + carry) 5 outputs (a 4 bit number + carry)
    signal test_bits_in : std_logic_vector (8 downto 0);
    signal test_bits_out : std_logic_vector (4 downto 0);

begin
    DUT : entity work.four_bit_adder(ripple_carry)
        port map (
                    carry_in => test_bits_in (8),

                    a (3) => test_bits_in (7),
                    a (2) => test_bits_in (6),
                    a (1) => test_bits_in (5),
                    a (0) => test_bits_in (4),

                    b (3) => test_bits_in (3),
                    b (2) => test_bits_in (2),
                    b (1) => test_bits_in (1),
                    b (0) => test_bits_in (0),

                    carry_out => test_bits_out (4),

                    sum (3) => test_bits_out (3),
                    sum (2) => test_bits_out (2),
                    sum (1) => test_bits_out (1),
                    sum (0) => test_bits_out (0)
                 );
    process begin
        for I in 0 to 255 loop
            test_bits_in <= std_logic_vector(to_unsigned(I,9));
            wait for 1 ns;
        end loop;

        assert false report "End of Test";
        wait;

    end process;

end tb_architecture;
```

```vhdl
-- VHDL 8 bit processor Project
-- 2 Bit Ripple Carry Adder
-- James Hicks Sept 22 2023

-- edited sept 23 to allow for carry in and carry out

library ieee;
use ieee.std_logic_1164.all;

entity two_bit_adder is
    port (
            carry_in : in std_logic;
            a, b : in std_logic_vector (1 downto 0);
            carry_out : out std_logic;
            sum : out std_logic_vector (1 downto 0)
         );
end two_bit_adder;

architecture ripple_carry of two_bit_adder is

    signal carry_through : std_logic;

begin
    lsb : entity work.full_adder(decomposed_arch)
        port map (
                    carry_in => carry_in,
                    i1 => a(0),
                    i0 => b(0),
                    carry_out => carry_through,
                    sum => sum(0)
                );
    msb : entity work.full_adder(decomposed_arch)
        port map (
                    carry_in => carry_through,
                    i1 => a(1),
                    i0 => b(1),
                    carry_out => carry_out,
                    sum => sum(1)
                 );

end ripple_carry;
```

```vhdl
-- VHDL 8 bit processor Project
-- Two Bit Adder Test Bench
-- James Hicks Sept 22 2023

library ieee;
use ieee.std_logic_1164.all;

entity two_bit_adder_tb is
end two_bit_adder_tb;

architecture tb_architecture of two_bit_adder_tb is

-- five inputs three outputs
    signal test_bits_in : std_logic_vector (4 downto 0);
    signal test_bits_out : std_logic_vector (2 downto 0);

begin
-- Two architectures::ripple_carry or carry_lookahead
    DUT : entity work.two_bit_adder(carry_lookahead)
        port map (
                    carry_in => test_bits_in (4),
                    a (1) => test_bits_in (3),
                    a (0) => test_bits_in (2),
                    b (1) => test_bits_in (1),
                    b (0) => test_bits_in (0),
                    carry_out => test_bits_out(2),
                    sum (1) => test_bits_out(1),
                    sum (0) => test_bits_out(0)
                );
    process begin
        -- test vector 1 --
        test_bits_in <= "00000";
        wait for 1 ns;

        -- test vector 2 --
        test_bits_in <= "00001";
        wait for 1 ns;

        -- test vector 3 --
        test_bits_in <= "00010";
        wait for 1 ns;

        -- test vector 4 --
        test_bits_in <= "00011";
        wait for 1 ns;

        -- test vector 5 --
        test_bits_in <= "00100";
        wait for 1 ns;

        -- test vector 6 --
        test_bits_in <= "00101";
        wait for 1 ns;

        -- test vector 7 --
        test_bits_in <= "00110";
        wait for 1 ns;

        -- test vector 8 --
        test_bits_in <= "00111";
        wait for 1 ns;

        -- test vector 9 --
        test_bits_in <= "01000";
        wait for 1 ns;

        -- test vector 10 --
        test_bits_in <= "01001";
        wait for 1 ns;

        -- test vector 11 --
        test_bits_in <= "01010";
        wait for 1 ns;

        -- test vector 12 --
        test_bits_in <= "01011";
        wait for 1 ns;

        -- test vector 13 --
        test_bits_in <= "01100";
        wait for 1 ns;

        -- test vector 14 --
        test_bits_in <= "01101";
        wait for 1 ns;

        -- test vector 15 --
        test_bits_in <= "01110";
        wait for 1 ns;

        -- test vector 16 --
        test_bits_in <= "01111";
        wait for 1 ns;

        assert false report "End of Test";
        wait;

    end process;

end tb_architecture;
```

```vhdl
-- VHDL 8 bit processor Project
-- Full Adder (not composed of half adders)
-- James Hicks Sept 21 2023

library ieee;
use ieee.std_logic_1164.all;

entity full_adder is
    port(carry_in, i1, i0 : in std_logic;
         carry_out, sum : out std_logic);
end full_adder;

architecture my_arch of full_adder is
    signal p2, p1, p0 : std_logic;
begin
    sum <= carry_in xor i1 xor i0;
    p2 <= carry_in and i1;
    p1 <= carry_in and i0;
    p0 <= i1 and i0;
    carry_out <= p2 or p1 or p0;
end my_arch;

-- decomposed architecture --
architecture decomposed_arch of full_adder is
    signal p0, p1, p2 : std_logic;
begin
    -- instantiate two half adders --
    half_adder_xin_yin : entity work.half_adder(my_arch)
        port map(i1=>i1, i0=>i0, sum=>p0, carry=>p1);
    half_adder_cin_sout : entity work.half_adder(my_arch)
        port map(i1=>carry_in, i0=>p0, sum=>sum, carry=>p2);
    carry_out <= p1 or p2;
end decomposed_arch;
```

```vhdl
-- VHDL 8 bit processor Project
-- Full Adder testbench
-- James Hicks Sept 22 2023

library ieee;
use ieee.std_logic_1164.all;

 -- a testbench has no ports
entity full_adder_tb is
end full_adder_tb;

architecture test of full_adder_tb is
    component full_adder
        port (
                carry_in, i1, i0 : in std_logic;
                carry_out, sum : out std_logic
             );
    end component;

    signal test_bits_in : std_logic_vector (2 downto 0);
    signal test_bits_out : std_logic_vector (1 downto 0);
begin
    DUT : full_adder port map (
                                carry_in => test_bits_in(2),
                                i1 => test_bits_in(1),
                                i0 => test_bits_in(0),
                                carry_out => test_bits_out(1),
                                sum => test_bits_out(0)
                              );
    process begin
        -- test vector 1 --
        test_bits_in <= "000";
        wait for 1 ns;

        -- test vector 2 --
        test_bits_in <= "001";
        wait for 1 ns;

        -- test vector 3 --
        test_bits_in <= "010";
        wait for 1 ns;

        -- test vector 4 --
        test_bits_in <= "011";
        wait for 1 ns;

        -- test vector 5 --
        test_bits_in <= "100";
        wait for 1 ns;

        -- test vector 6 --
        test_bits_in <= "101";
        wait for 1 ns;

        -- test vector 7 --
        test_bits_in <= "110";
        wait for 1 ns;

        -- test vector 8 --
        test_bits_in <= "111";
        wait for 1 ns;

        assert false report "End of Test";
        wait;

    end process;
end test;
```

```vhdl
-- VHDL 8 bit processor Project
-- Half Adder
-- James Hicks Sept 21 2023

library ieee;
use ieee.std_logic_1164.all;

entity half_adder is
port (i1, i0 : in std_logic;
      sum, carry : out std_logic);
end half_adder;

architecture my_arch of half_adder is
begin
    sum <= i1 xor i0;
    carry <= i1 and i0;
end my_arch;
```

```vhdl
-- VHDL 8 bit processor Project
-- Half Adder Test Bench
-- James Hicks Sept 22 2023

library ieee;
use ieee.std_logic_1164.all;

-- a testbench has no ports
entity half_adder_tb is
end half_adder_tb;

architecture tb_architecture of half_adder_tb is
    component half_adder
        port (
                i1, i0 : in std_logic;
                sum, carry : out std_logic
             );
    end component;

    signal test_bits_in : std_logic_vector (1 downto 0);
    signal test_bits_out : std_logic_vector (1 downto 0);
begin
    DUT : half_adder port map (
                                i1 => test_bits_in(1),
                                i0 => test_bits_in(0),
                                carry => test_bits_out(1),
                                sum => test_bits_out(0)
                              );
    process begin
        -- test vector 1 --
        test_bits_in <= "00";
        wait for 1 ns;

        -- test vector 2 --
        test_bits_in <= "01";
        wait for 1 ns;

        -- test vector 3 --
        test_bits_in <= "10";
        wait for 1 ns;

        -- test vector 4 --
        test_bits_in <= "11";
        wait for 1 ns;

        assert false report "End of Test";
        wait;

    end process;
end tb_architecture;
```

1. For the small circuits (e.g., full-adder), it is fine to have the simulation test all possible combinations. However, testing all combinations for the 4-bit binary adder/subtractor would mean testing $2^8 = 256$ combinations, which is probably too much for our purposes. Thus, it is reasonable to test only a few input vectors. I will let you choose them, but you should always check all the key possibilities (e.g., extremes, interesting scenarios, potential errors, etc.).
2. For the adder/subtractor simulation, make sure that you present the simulation as *signed decimals*.

D. Implementation

Download the 4-bit binary adder/subtractor to the board. Verify its operation. Have the instructor check your implementation.

James Hicks
Your Name

[illegible]
Instructor signature
Lab 2

Deliverables:

1. **(4 pts)** Copies of the circuit diagrams from part A. Note: your circuit should match the VHDL code, i.e., use similar input names, output names, and intermediate signal names.
2. **(4 pts)** A color-coded version of each one of your VHDL components (i.e., full-adder, 4-bit binary adder, and the 4-bit binary adder/subtractor.
3. **(4 pts)** A color-coded version of each of the appropriate test-benches.
4. **(4 pts)** Legible copies of your simulations. Make sure that you provide appropriate comments wherever needed, e.g., if something is not what you expected, or if the system fails, etc.
5. **(1 pt)** A copy of your constraints file.
6. **(3 pts)** A copy of the instructor signature.

All the deliverables should be in PDF format, preferably as a single file.

How to turn this in?
You should turn this in via Canvas.