

Four bit adder subtractor

Four Bit Adder

composed of two 2 bit adders

Last Two Bits

b3 a3



b2 a2



carry thru

First Two Bits

b1 a1



b0 a0



carry in



carry out



sum 3



sum 2



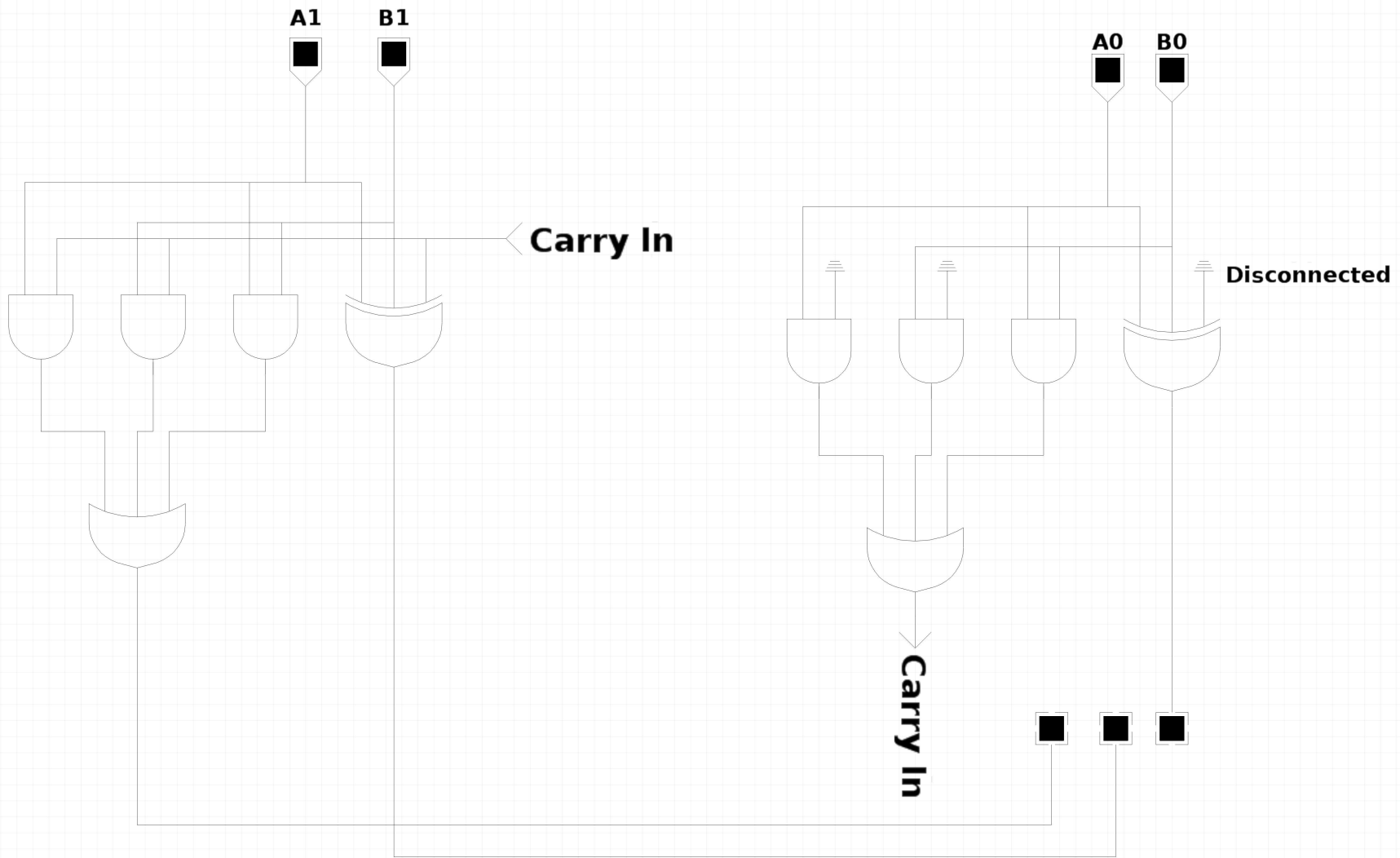
sum 1



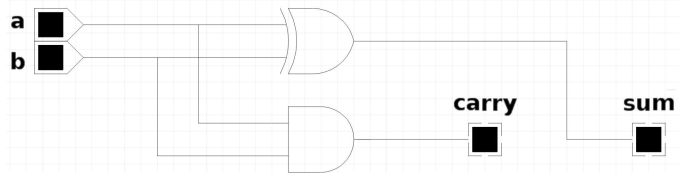
sum 0



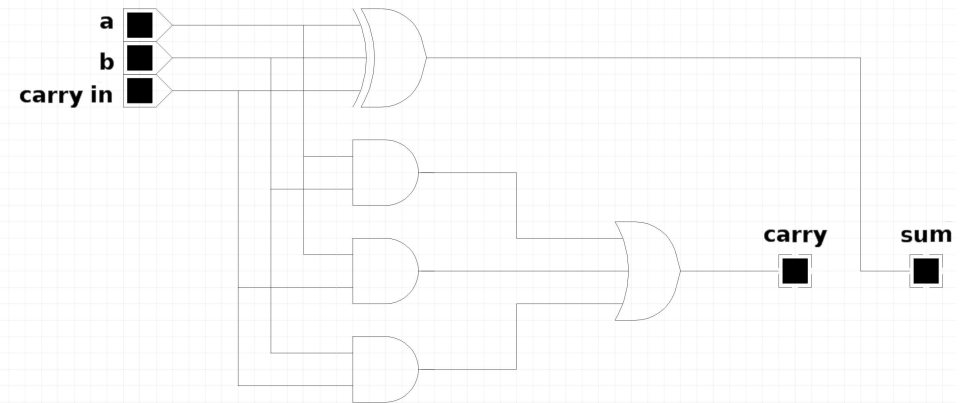
Two Bit Ripple Carry Adder



half adder



full adder



```

1  -- VHDL 8 bit processor Project
2  -- Four Bit Binary Adder Subtractor
3  -- James Hicks Sept 29 2023
4
5  library ieee;
6  use ieee.std_logic_1164.all;
7
8  entity four_bit_adder_subtractor is
9      port (
10         control : in std_logic;
11         a, b : in std_logic_vector (3 downto 0);
12         carry_out : out std_logic;
13         sum_diff : out std_logic_vector (3 downto 0)
14     );
15 end four_bit_adder_subtractor;
16
17 -- The control is essentially a carry in
18 -- With the added caveat that in addition to being added
19 -- to the other inputs, it also decides which output to take
20
21 architecture my_arch of four_bit_adder_subtractor is
22
23     signal add_carry, sub_carry : std_logic;
24     signal sum, diff : std_logic_vector (3 downto 0);
25     signal b_flipped : std_logic_vector (3 downto 0);
26
27 begin
28
29     b_flipped <= (not b(3)) & (not b(2)) & (not b(1)) & (not b(0));
30
31     add : entity work.four_bit_adder(ripple_carry)
32     port map (
33         carry_in => control,
34         a => a,
35         b => b,
36         carry_out => add_carry,
37         sum => sum
38     );
39     subtract : entity work.four_bit_adder(ripple_carry)
40     port map (
41         carry_in => control,
42         a => a,
43         b => b_flipped,
44         carry_out => sub_carry,
45         sum => diff
46     );
47
48     with control select sum_diff <=
49         sum when '0',
50         diff when '1',
51         "XXXX" when others;
52
53     with control select carry_out <=
54         add_carry when '0',
55         sub_carry when '1',
56         'X' when others;
57
58     --with control select sum_diff <=
59     --    sum when '0',
60     --    diff when '1',
61     --    "XXXX" when others;
62     --
63     --with control select carry_out <=
64     --    add_carry when '0',
65     --    '0' when '1',
66     --    'X' when others;
67

```

```
68 end my_arch;
```

```

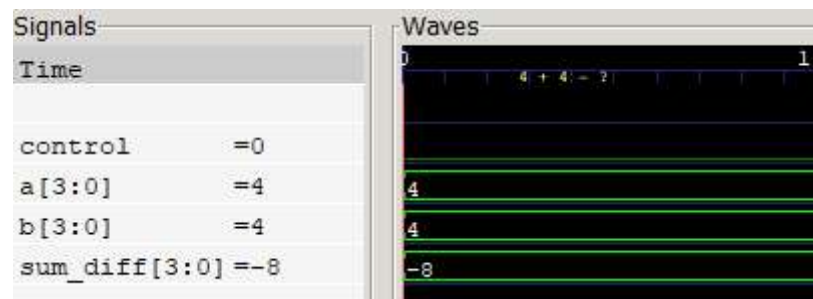
1  -- VHDL 8 bit processor Project
2  -- Four Bit Binary Adder Subtractor Test Bench
3  -- James Hicks Sept 29 2023
4
5  -- The bit structure is as follows
6  -- (control bit) & (a input) & (b input)
7  -- Test bits out simply represents the sum or difference of inputs
8
9  library ieee;
10 use ieee.std_logic_1164.all;
11 use ieee.numeric_std.all;
12
13 entity four_bit_adder_subtractor_tb is
14 end four_bit_adder_subtractor_tb;
15
16 architecture tb_architecture of four_bit_adder_subtractor_tb is
17
18     -- 9 inputs (2 4-bit numbers + control) 5 outputs (a 4 bit number + carry)
19     signal control : std_logic;
20     signal a, b : std_logic_vector(3 downto 0);
21     signal carry_out : std_logic;
22     signal sum_diff : std_logic_vector(3 downto 0);
23
24 begin
25     DUT : entity work.four_bit_adder_subtractor(my_arch)
26         port map (
27             control => control,
28             a => a,
29             b => b,
30             carry_out => carry_out,
31             sum_diff => sum_diff
32         );
33     process begin
34         -- test 4 + 4 = 8 (out of range for signed 4 bit number)
35         control <= '0';
36         a <= "0100";
37         b <= "0100";
38         wait for 1 ns;
39
40         -- test for -4 + -5 = -9 (out of range for signed 4 bit number)
41         control <= '0';
42         a <= "1100";
43         b <= "1011";
44         wait for 1 ns;
45
46         -- test adding positive and negative extremes 7 + -8 = -1
47         control <= '0';
48         a <= "0111";
49         b <= "1000";
50         wait for 1 ns;
51
52         -- test for subtraction overflow 3 - -6 = 9 (out of range)
53         control <= '1';
54         a <= "0101";
55         b <= "1010";
56         wait for 1 ns;
57
58         -- test for subtraction underflow -7 - 7 (out of range)
59         control <= '1';
60         a <= "1111";
61         b <= "0111";
62         wait for 1 ns;
63
64         -- test for 0 - 1 = -1
65         control <= '1';
66         a <= "0000";
67         b <= "0001";

```

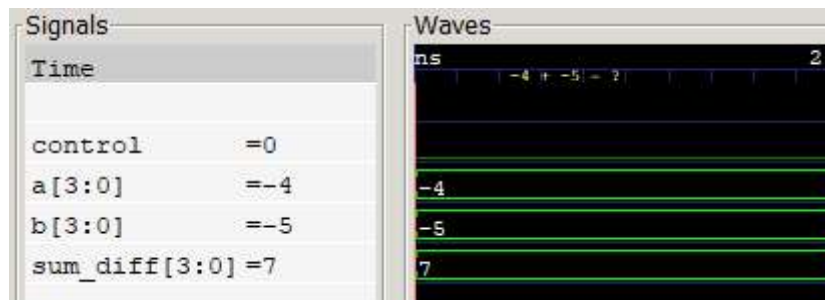
```
68         wait for 1 ns;
69
70         -- test for 1 -- 1 = 0
71         control <= '1';
72         a <= "1111";
73         b <= "1111";
74         wait for 1 ns;
75
76         assert false report "End of Test";
77         wait;
78
79     end process;
80
81 end tb_architecture;
```


SIMULATION

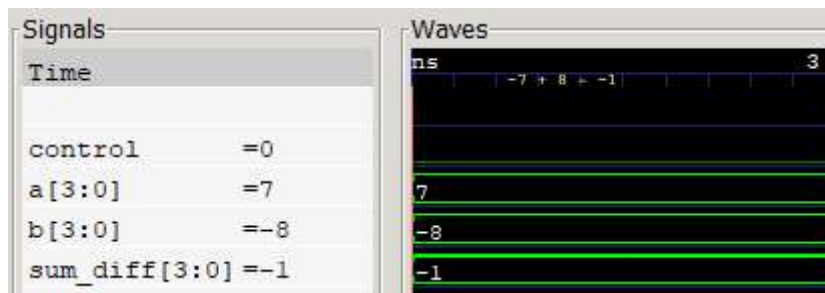
Overflow Addition : $4 + 4 = ?$



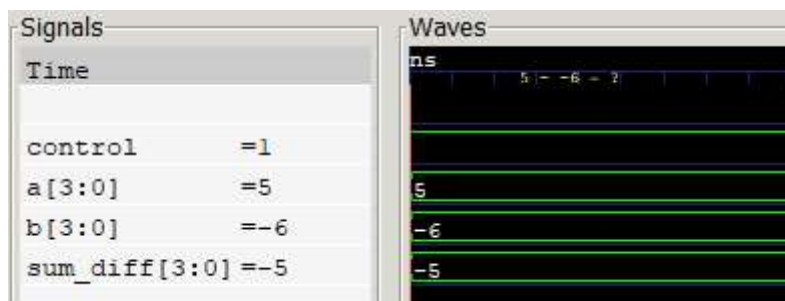
Underflow addition: $-4 + -5 = ?$



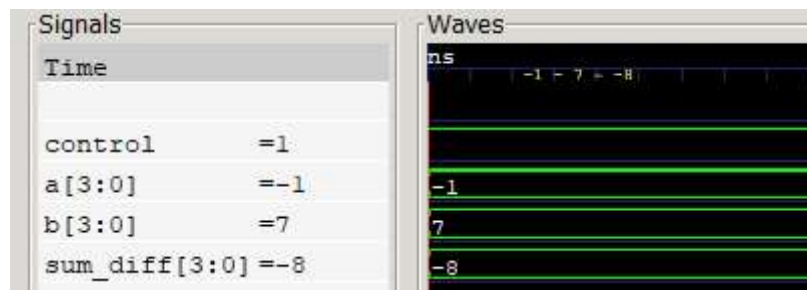
Adding extremes: $7 + -8 = -1$



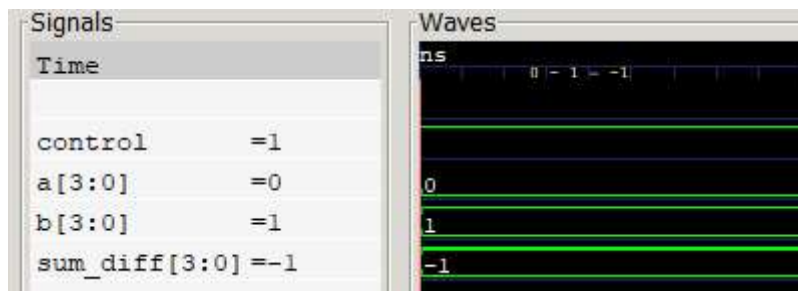
Overflow subtraction: $5 - -6 = ?$



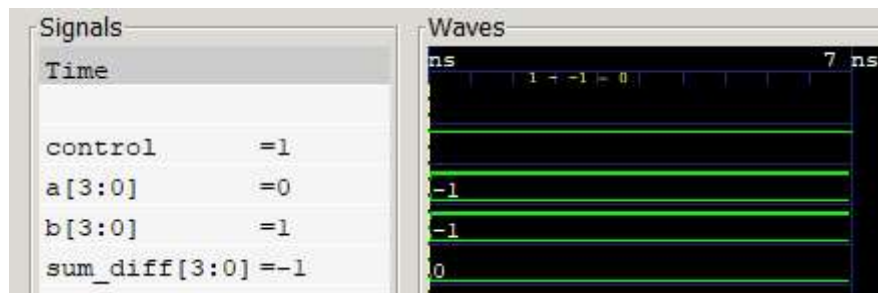
lowest possible outcome subtraction: $-1 - 7 = -8$



$0 - 1 = -1$



$-1 - -1 = 0$



```

1  -- VHDL 8 bit processor Project
2  -- Four Bit Adder
3  -- James Hicks Sept 23 2023
4
5  library ieee;
6  use ieee.std_logic_1164.all;
7
8  entity four_bit_adder is
9      port (
10         carry_in : in std_logic;
11         a, b : in std_logic_vector (3 downto 0);
12         carry_out : out std_logic;
13         sum : out std_logic_vector (3 downto 0)
14     );
15 end four_bit_adder;
16
17 architecture ripple_carry of four_bit_adder is
18
19     signal carry_through : std_logic;
20
21     begin
22         first_two_bits : entity work.two_bit_adder(ripple_carry)
23             port map (
24                 carry_in => carry_in,
25                 a => a (1 downto 0),
26                 b => b (1 downto 0),
27                 carry_out => carry_through,
28                 sum => sum (1 downto 0)
29             );
30         last_two_bits : entity work.two_bit_adder(ripple_carry)
31             port map (
32                 carry_in => carry_through,
33                 a => a (3 downto 2),
34                 b => b (3 downto 2),
35                 carry_out => carry_out,
36                 sum => sum (3 downto 2)
37             );
38
39     end ripple_carry;

```

```

1  -- VHDL 8 bit processor Project
2  -- Four Bit Adder Test Bench
3  -- James Hicks Sept 23 2023
4
5  -- The structure of test bits in is as follows
6  -- (carry in bit) & (a input) & (b input)
7  -- Test bits out simply represents the sum of the two input numbers
8
9  library ieee;
10 use ieee.std_logic_1164.all;
11 use ieee.numeric_std.all;
12
13 entity four_bit_adder_tb is
14 end four_bit_adder_tb;
15
16 architecture tb_architecture of four_bit_adder_tb is
17
18     -- 9 inputs (two 4 bit numbers + carry) 5 outputs (a 4 bit number + carry)
19     signal test_bits_in : std_logic_vector (8 downto 0);
20     signal test_bits_out : std_logic_vector (4 downto 0);
21
22 begin
23     DUT : entity work.four_bit_adder(ripple_carry)
24         port map (
25             carry_in => test_bits_in (8),
26
27             a (3) => test_bits_in (7),
28             a (2) => test_bits_in (6),
29             a (1) => test_bits_in (5),
30             a (0) => test_bits_in (4),
31
32             b (3) => test_bits_in (3),
33             b (2) => test_bits_in (2),
34             b (1) => test_bits_in (1),
35             b (0) => test_bits_in (0),
36
37             carry_out => test_bits_out (4),
38
39             sum (3) => test_bits_out (3),
40             sum (2) => test_bits_out (2),
41             sum (1) => test_bits_out (1),
42             sum (0) => test_bits_out (0)
43         );
44     process begin
45         for I in 0 to 255 loop
46             test_bits_in <= std_logic_vector(to_unsigned(I,9));
47             wait for 1 ns;
48         end loop;
49
50         assert false report "End of Test";
51         wait;
52     end process;
53
54 end tb_architecture;
55

```

```

1  -- VHDL 8 bit processor Project
2  -- 2 Bit Ripple Carry Adder
3  -- James Hicks Sept 22 2023
4
5  -- edited sept 23 to allow for carry in and carry out
6
7  library ieee;
8  use ieee.std_logic_1164.all;
9
10 entity two_bit_adder is
11     port (
12         carry_in : in std_logic;
13         a, b : in std_logic_vector (1 downto 0);
14         carry_out : out std_logic;
15         sum : out std_logic_vector (1 downto 0)
16     );
17 end two_bit_adder;
18
19 architecture ripple_carry of two_bit_adder is
20
21     signal carry_through : std_logic;
22
23 begin
24     lsb : entity work.full_adder(decomposed_arch)
25     port map (
26         carry_in => carry_in,
27         i1 => a(0),
28         i0 => b(0),
29         carry_out => carry_through,
30         sum => sum(0)
31     );
32     msb : entity work.full_adder(decomposed_arch)
33     port map (
34         carry_in => carry_through,
35         i1 => a(1),
36         i0 => b(1),
37         carry_out => carry_out,
38         sum => sum(1)
39     );
40
41 end ripple_carry;

```

```

1  -- VHDL 8 bit processor Project
2  -- Two Bit Adder Test Bench
3  -- James Hicks Sept 22 2023
4
5  library ieee;
6  use ieee.std_logic_1164.all;
7
8  entity two_bit_adder_tb is
9  end two_bit_adder_tb;
10
11 architecture tb_architecture of two_bit_adder_tb is
12
13     -- five inputs three outputs
14     signal test_bits_in : std_logic_vector (4 downto 0);
15     signal test_bits_out : std_logic_vector (2 downto 0);
16
17 begin
18     -- Two architectures::ripple_carry or carry_lookahead
19     DUT : entity work.two_bit_adder (carry_lookahead)
20         port map (
21             carry_in => test_bits_in (4),
22             a (1) => test_bits_in (3),
23             a (0) => test_bits_in (2),
24             b (1) => test_bits_in (1),
25             b (0) => test_bits_in (0),
26             carry_out => test_bits_out (2),
27             sum (1) => test_bits_out (1),
28             sum (0) => test_bits_out (0)
29         );
30     process begin
31         -- test vector 1 --
32         test_bits_in <= "00000";
33         wait for 1 ns;
34
35         -- test vector 2 --
36         test_bits_in <= "00001";
37         wait for 1 ns;
38
39         -- test vector 3 --
40         test_bits_in <= "00010";
41         wait for 1 ns;
42
43         -- test vector 4 --
44         test_bits_in <= "00011";
45         wait for 1 ns;
46
47         -- test vector 5 --
48         test_bits_in <= "00100";
49         wait for 1 ns;
50
51         -- test vector 6 --
52         test_bits_in <= "00101";
53         wait for 1 ns;
54
55         -- test vector 7 --
56         test_bits_in <= "00110";
57         wait for 1 ns;
58
59         -- test vector 8 --
60         test_bits_in <= "00111";
61         wait for 1 ns;
62
63         -- test vector 9 --
64         test_bits_in <= "01000";
65         wait for 1 ns;
66
67         -- test vector 10 --
68         test_bits_in <= "01001";
69         wait for 1 ns;
70
71         -- test vector 11 --
72         test_bits_in <= "01010";

```

```
73         wait_for(1 ns;
74
75         -- test vector 12 --
76         test_bits_in <= "01011";
77         wait_for(1 ns;
78
79         -- test vector 13 --
80         test_bits_in <= "01100";
81         wait_for(1 ns;
82
83         -- test vector 14 --
84         test_bits_in <= "01101";
85         wait_for(1 ns;
86
87         -- test vector 15 --
88         test_bits_in <= "01110";
89         wait_for(1 ns;
90
91         -- test vector 16 --
92         test_bits_in <= "01111";
93         wait_for(1 ns;
94
95         assert false report "End of Test";
96         wait;
97
98     end process;
99
100 end tb_architecture;
```

```

1  -- VHDL 8 bit processor Project
2  -- Full Adder (not composed of half adders)
3  -- James Hicks Sept 21 2023
4
5  library ieee;
6  use ieee.std_logic_1164.all;
7
8  entity full_adder is
9      port (carry_in, i1, i0 : in std_logic;
10           carry_out, sum : out std_logic);
11  end full_adder;
12
13  architecture my_arch of full_adder is
14      signal p2, p1, p0 : std_logic;
15  begin
16      sum <= carry_in xor i1 xor i0;
17      p2 <= carry_in and i1;
18      p1 <= carry_in and i0;
19      p0 <= i1 and i0;
20      carry_out <= p2 or p1 or p0;
21  end my_arch;
22
23  -- decomposed architecture --
24  architecture decomposed_arch of full_adder is
25      signal p0, p1, p2 : std_logic;
26  begin
27      -- instantiate two half adders --
28      half_adder_xin_yin : entity work.half_adder (my_arch)
29          port map (i1=>i1, i0=>i0, sum=>p0, carry=>p1);
30      half_adder_cin_sout : entity work.half_adder (my_arch)
31          port map (i1=>carry_in, i0=>p0, sum=>sum, carry=>p2);
32      carry_out <= p1 or p2;
33  end decomposed_arch;

```



```

1  -- VHDL 8 bit processor Project
2  -- Full Adder testbench
3  -- James Hicks Sept 22 2023
4
5  library ieee;
6  use ieee.std_logic_1164.all;
7
8  -- a testbench has no ports
9  entity full_adder_tb is
10 end full_adder_tb;
11
12 architecture test of full_adder_tb is
13     component full_adder
14     port (
15         carry_in, i1, i0 : in std_logic;
16         carry_out, sum : out std_logic
17     );
18     end component;
19
20     signal test_bits_in : std_logic_vector (2 downto 0);
21     signal test_bits_out : std_logic_vector (1 downto 0);
22 begin
23     DUT : full_adder port map (
24         carry_in => test_bits_in(2),
25         i1 => test_bits_in(1),
26         i0 => test_bits_in(0),
27         carry_out => test_bits_out(1),
28         sum => test_bits_out(0)
29     );
30     process begin
31         -- test vector 1 --
32         test_bits_in <= "000";
33         wait for 1 ns;
34
35         -- test vector 2 --
36         test_bits_in <= "001";
37         wait for 1 ns;
38
39         -- test vector 3 --
40         test_bits_in <= "010";
41         wait for 1 ns;
42
43         -- test vector 4 --
44         test_bits_in <= "011";
45         wait for 1 ns;
46
47         -- test vector 5 --
48         test_bits_in <= "100";
49         wait for 1 ns;
50
51         -- test vector 6 --
52         test_bits_in <= "101";
53         wait for 1 ns;
54
55         -- test vector 7 --
56         test_bits_in <= "110";
57         wait for 1 ns;
58
59         -- test vector 8 --
60         test_bits_in <= "111";
61         wait for 1 ns;
62
63         assert false report "End of Test";
64         wait;
65
66     end process;
67 end test;

```

```
1  -- VHDL 8 bit processor Project
2  -- Half Adder
3  -- James Hicks Sept 21 2023
4
5  library ieee;
6  use ieee.std_logic_1164.all;
7
8  entity half_adder is
9  port (i1, i0 : in std_logic;
10       sum, carry : out std_logic);
11  end half_adder;
12
13  architecture my_arch of half_adder is
14  begin
15     sum <= i1 xor i0;
16     carry <= i1 and i0;
17  end my_arch;
```

```

1  -- VHDL 8 bit processor Project
2  -- Half Adder Test Bench
3  -- James Hicks Sept 22 2023
4
5  library ieee;
6  use ieee.std_logic_1164.all;
7
8  -- a testbench has no ports
9  entity half_adder_tb is
10 end half_adder_tb;
11
12 architecture tb_architecture of half_adder_tb is
13     component half_adder
14     port (
15         i1, i0 : in std_logic;
16         sum, carry : out std_logic
17     );
18     end component;
19
20     signal test_bits_in : std_logic_vector (1 downto 0);
21     signal test_bits_out : std_logic_vector (1 downto 0);
22 begin
23     DUT : half_adder port map (
24         i1 => test_bits_in(1),
25         i0 => test_bits_in(0),
26         carry => test_bits_out(1),
27         sum => test_bits_out(0)
28     );
29     process begin
30         -- test vector 1 --
31         test_bits_in <= "00";
32         wait for 1 ns;
33
34         -- test vector 2 --
35         test_bits_in <= "01";
36         wait for 1 ns;
37
38         -- test vector 3 --
39         test_bits_in <= "10";
40         wait for 1 ns;
41
42         -- test vector 4 --
43         test_bits_in <= "11";
44         wait for 1 ns;
45
46         assert false report "End of Test";
47         wait;
48
49     end process;
50 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

Instructor signature

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.