Computer Logic – Practical 2

Objective:

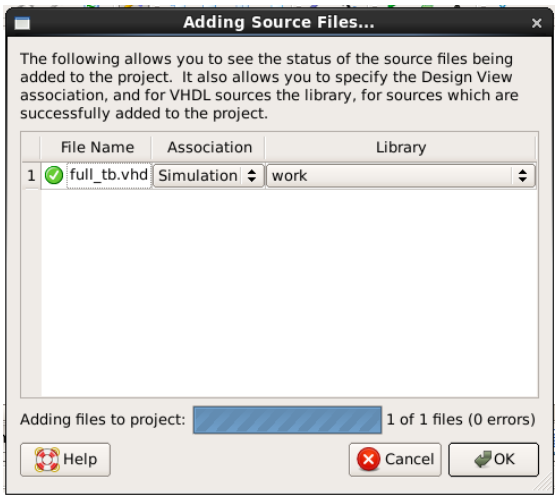
To build, test and compare a carry-ripple adder and a carry-lookahead adder.

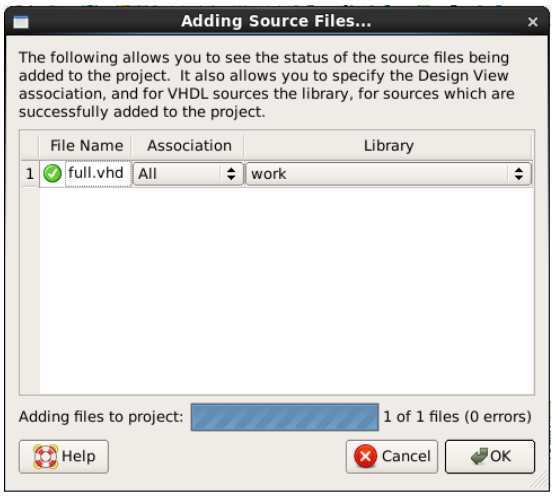
Tasks:

1) A new project called "*Project2*" was created using the same procedure as used in "*Project1*".

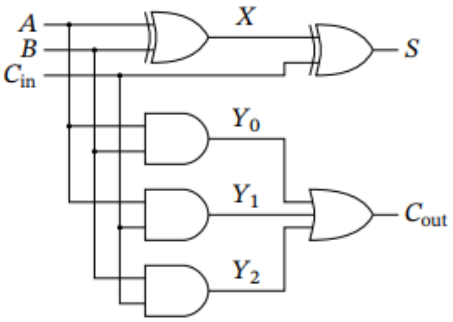
2) The following files where then copied to the project directory:
   - *full.vhd*
   - *full_tb.vhd*
   - *ripple.vhd*
   - *ripple_tb.vhd*
   - *full_lookahead.vhd*
   - *full_lookahead_tb.vhd*
   - *lookahead.vhd*
   - *lookahead_tb.vhd*

3) The 8 files above were then added to the project with the menu item *Project: Add Source…* The association was set to *Simulation* for files with "_tb" in their filename. Those without "_tb" in the filename were associated to *All* as shown:



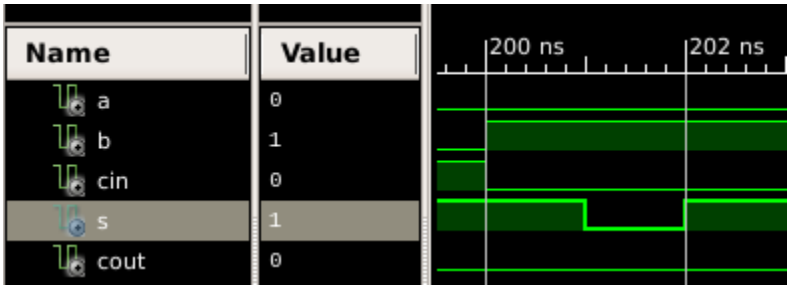


4) The file *full.vhd* defines the single-bit full adder shown below. Statements for X and S were already written in this file and both contained a delay clause. Statements for Y_0 , Y_1 , Y_2 and C_{out} were then added to complete the full-adder. Similar delay clauses to those of X and S were implemented in the VHDL code.

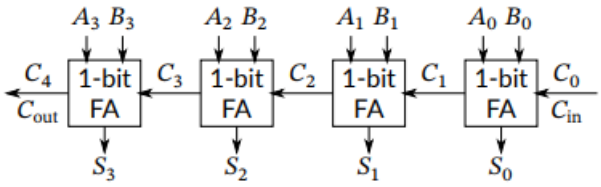



```
begin  -- architecture behavior

   X <= A xor B   after delay;
   S <= Cin xor X after delay;
   Y0 <= A and B after delay;
   Y1 <= A and Cin after delay;
   Y2 <= B and Cin after delay;
   Cout <= Y0 or Y1 or Y2 after delay;

end behavior;
```

5) The *full_tb.vhd* file defining a testbench for the full adder was simulated using behavioral simulation and the output result was analysed. The emulated gate delay was known to be 1 ns for each gate whilst the propagation delays for the outputs S and C_{out} were found to be 2 ns as shown in one instance between 200 and 202 ns.



6) In this step, the entity *ripple.vhd* defining the four-bit ripple-carry adder in the diagram was looked into. This particular task required observing the VHDL code only, especially the part where the four full adders were instantiated.




```
gen_full : for i in 0 to 3 generate
    full_i : full
        port map (
            A    => A(i),
            B    => B(i),
            Cin  => C(i),
            S    => S(i),
            Cout => C(i + 1));
end generate gen_full;
```

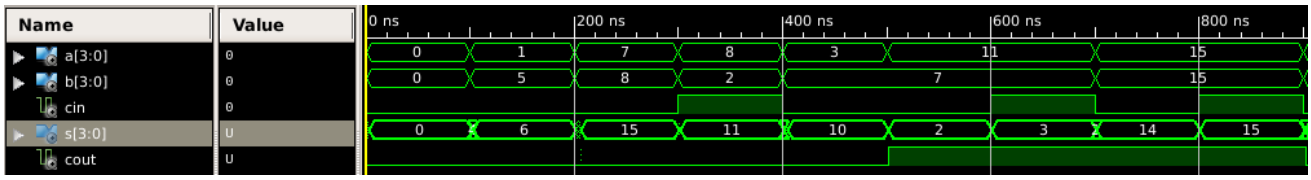
7) The test bench *ripple_tb.vhd* was used to simulate inputs for the testing of the carry-ripple adder. This time, these inputs were not tested exhaustively. The following additions were performed to try out the test bench with the first addition having all inputs as 0s and the final one having all inputs as 1s:

```
A   <= "0000";
B   <= "0000";
Cin <= '0';
wait for 100 ns;

A   <= "1000";
B   <= "0010";
Cin <= '1';
wait for 100 ns;

A   <= "1111";
B   <= "1111";
Cin <= '0';
wait for 100 ns;
```

```
A   <= "0001";
B   <= "0101";
Cin <= '0';
wait for 100 ns;

A   <= "0011";
B   <= "0111";
Cin <= '0';
wait for 100 ns;
```

```
A   <= "0111";
B   <= "1000";
Cin <= '0';
wait for 100 ns;

A   <= "1011";
B   <= "0111";
Cin <= '0';
wait for 100 ns;

A   <= "1111";
B   <= "1111";
Cin <= '1';
wait for 100 ns;
```

All these input statements were chosen carefully to try out as many different cases as possible. The sum and carry outputs were then checked to see if they were as expected.



8) In this step, the propagation delay of C_{out} for the ripple-carry adder was measured as follows:
   a) An addition operation with operands A = 0000_2 , B = 0000_2 , and C_{in} = 0 was needed to be added so that C_0, C_1, C_2, C_3 and C_4 all equal 0. However this was already part of the code as it was one of the operands being tested in step 7.

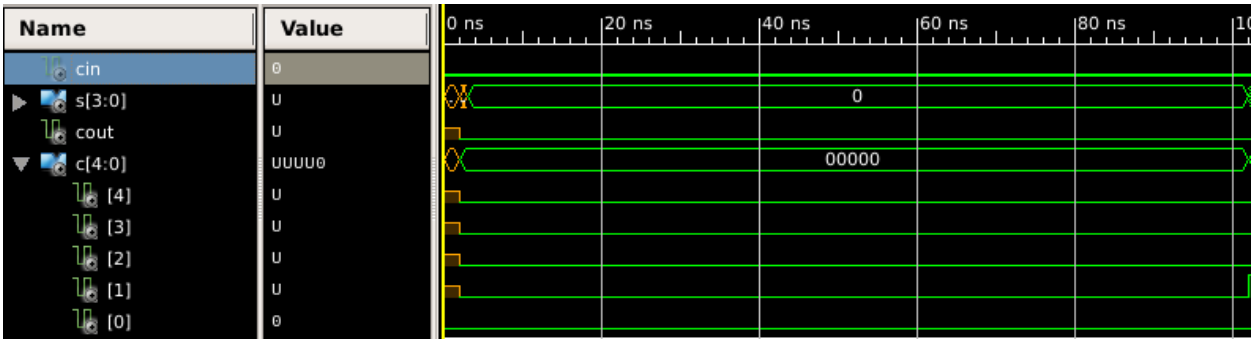
```
A   <= "0000";
B   <= "0000";
Cin <= '0';
wait for 100 ns;
```

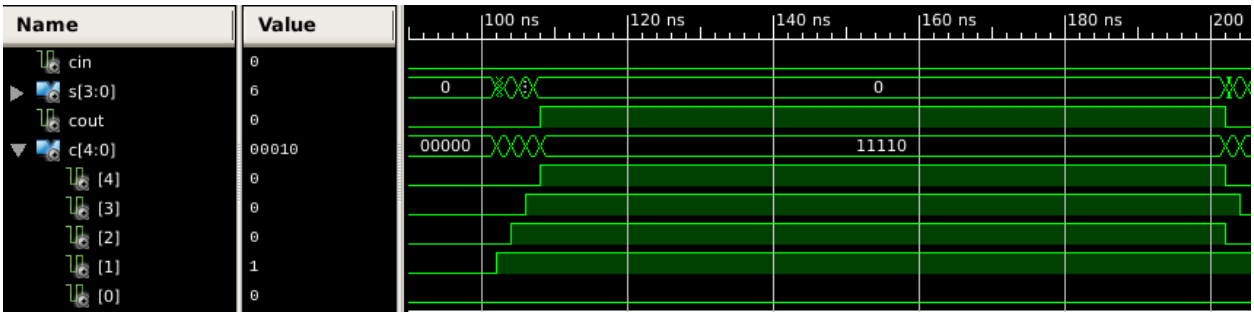
   b) Another addition operation with operands A = 0001_2 , B= 1111_2 , and C_{in} = 0 was needed to be added so that the carry propagates through all the internal carries. This operand was not part of the previous operands being tested in step 7 so it was added at this stage.

```
A   <= "0001";
B   <= "1111";
Cin <= '0';
wait for 100 ns;
```

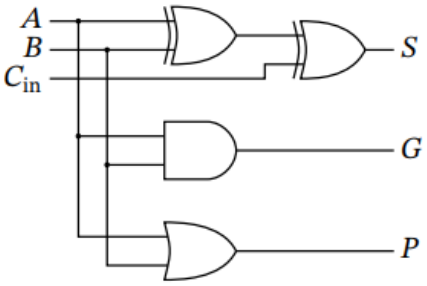
c) The test bench was simulated once again to observe the internal propagation delay of the ripple. This was done by dragging the object *C [4:0]* from the *instances pane: ripple_tb: uut* to the waveform pane. The added waveforms were then made visible once the simulation was reloaded again.

The screenshots below show the different cases. The first one represents case (a) in which all inputs were 0 and there was no need for the carry to propagate at all. The second screenshot represents case (b) in which the carry was required to propagate through all internal carries. This was done with a propagation delay of 2 ns per propagation.



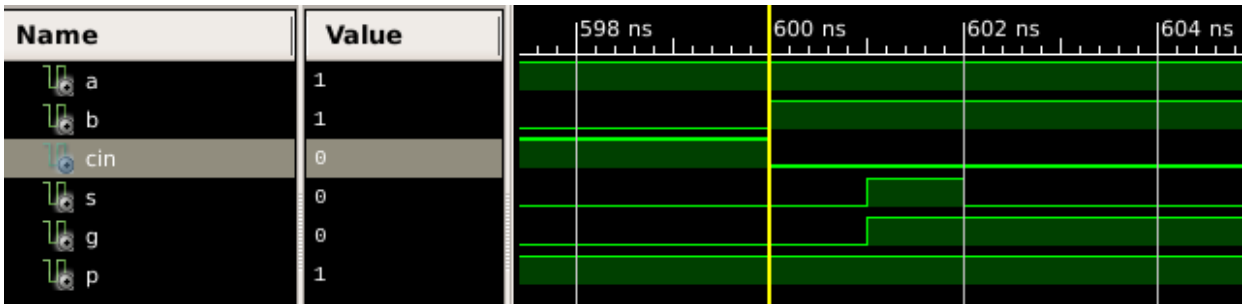


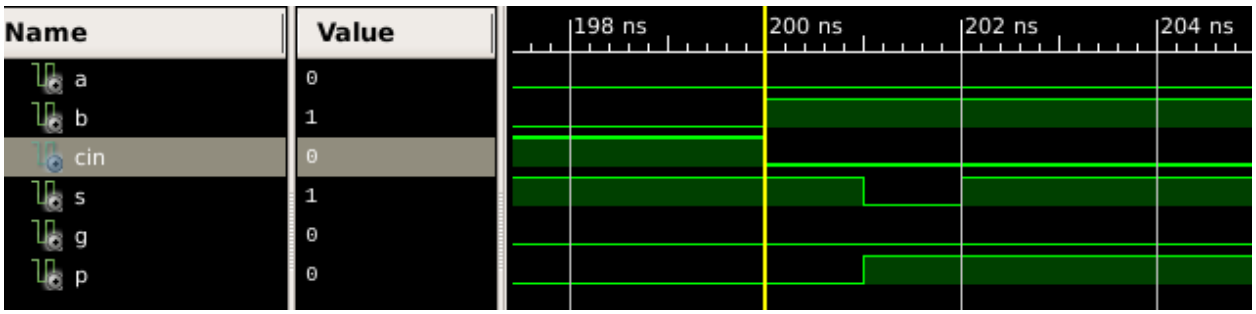
9) The file *full_lookahead.vhd* contains a single-bit full-adder for a carry-lookahead adder shown. Statements for G and P were then added to complete the single-bit full-adder.



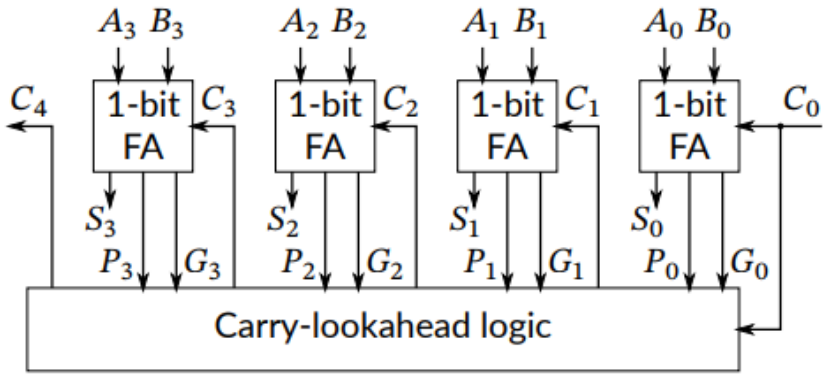

```
X <= A xor B   after delay;
S <= Cin xor X after delay;
G <= A and B after delay;
P <= A or B after delay;
```

10) The file *full_lookahead_tb.vhd* contains a test bench for the single-bit carry-lookahead adder. This file was simulated and the propagation delays for G and P were measured. These were both found to be 1 ns. The screenshots below show proof for this with the first one showing G and the second, P.



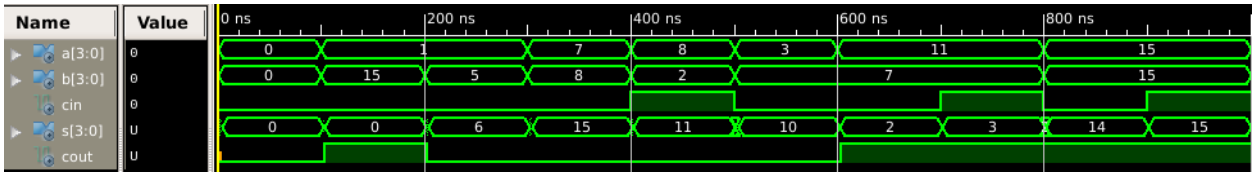


11) The file *lookahead.vhd* containing an implementation of the carry-lookahead adder shown was modified by including statements for C_2, C_3 and C_4.

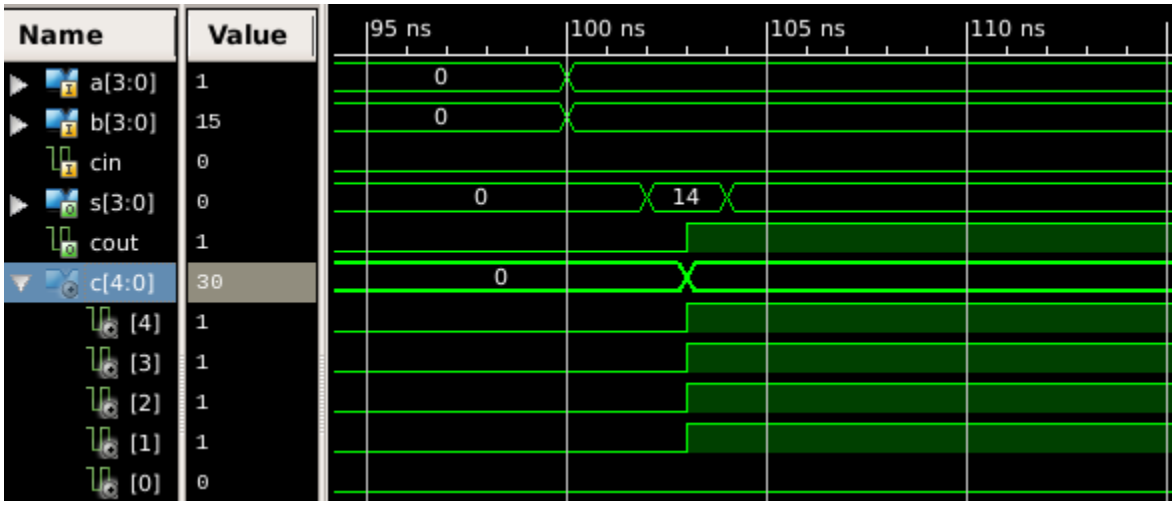


```
C(1) <= G(0) or (C(0) and P(0)) after 2 * delay;
C(2) <= G(1) or (G(0) and P(1)) or (C(0) and P(0) and P(1)) after 2 * delay;
C(3) <= G(2) or (G(1) and P(2)) or (G(0) and P(1) and P(2))
             or (C(0) and P(0) and P(1) and P(2)) after 2 * delay;
C(4) <= G(3) or (G(2) and P(3)) or (G(1) and P(2) and P(3))
             or (G(0) and P(1) and P(2) and P(3))
             or (C(0) and P(0) and P(1) and P(2) and P(3)) after 2 * delay;
```

12) The implementation of the test bench *lookahead_tb.vhd* was completed by copying the complete stimulus process from *ripple_tb.vhd*. Then, each output sum was checked for correctness sake.



13) The maximum propagation carry delay for the lookahead adder was found to be 3 ns as shown in the screenshot. This is because the signal would have to pass through 3 gates having a propagation delay of 1 ns delay per gate. Comparing this to step 8, we see that all the C_{outs} in this case were computed at the same time since one C_{out} didn't lead on to the other. On the other hand, in step 8 the propagation delay was of 2 ns per C_{out} since one input would lead onto the other.



Appendix (Code):

full.vhd:

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity full is
6
7      port (
8          A    : in  std_logic;
9          B    : in  std_logic;
10         Cin  : in  std_logic;
11         S    : out std_logic;
12         Cout : out std_logic);
13
14 end entity full;
15
16 architecture behavior of full is
17
18     constant delay : time := 1 ns;
19
20     signal X, Y0, Y1, Y2 : std_logic;
21
22 begin  -- architecture behavior
23
24     X <= A xor B   after delay;
25     S <= Cin xor X after delay;
26     Y0 <= A and   B after delay;
27     Y1 <= A and Cin after delay;
28     Y2 <= B and Cin after delay;
29     Cout <= Y0 or Y1 or Y2 after delay;
30
31 end behavior;
```

full_tb.vhd:

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity full_tb is
6  end entity full_tb;
7
8  architecture behavior of full_tb is
9
10     component full is
11         port (
12             A    : in  std_logic;
13             B    : in  std_logic;
14             Cin  : in  std_logic;
15             S    : out std_logic;
16             Cout : out std_logic);
17     end component full;
18
19     -- inputs
20     signal A   : std_logic := '0';
21     signal B   : std_logic := '0';
22     signal Cin : std_logic := '0';
23
24     -- outputs
25     signal S    : std_logic;
26     signal Cout : std_logic;
27
28 begin  -- architecture behavior
29
30     -- unit under test
31     uut : full
32         port map (
33             A    -> A,
34             B    -> B,
35             Cin  -> Cin,
36             S    -> S,
37             Cout -> Cout);
38
39     -- stimulus process
40     stim_proc : process is
41
42         variable u : unsigned(2 downto 0);
43
44     begin  -- process stim_proc
45
46         -- include 8 in loop so that the 3 inputs go back to 0
47         for i in 0 to 8 loop
48             -- convert i to a 3-bit vector
49             u := to_unsigned(i, 3);
50
51             A   <= u(2);
52             B   <= u(1);
53             Cin <= u(0);
54
55             wait for 100 ns;
56         end loop;
57
58     end process stim_proc;
59
60 end architecture behavior;
```

ripple.vhd:

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity ripple is
6
7      port (
8          A    : in  std_logic_vector(3 downto 0);
9          B    : in  std_logic_vector(3 downto 0);
10         Cin  : in  std_logic;
11         S    : out std_logic_vector(3 downto 0);
12         Cout : out std_logic);
13
14 end entity ripple;
15
16 architecture behavior of ripple is
17
18     component full is
19         port (
20             A    : in  std_logic;
21             B    : in  std_logic;
22             Cin  : in  std_logic;
23             S    : out std_logic;
24             Cout : out std_logic);
25     end component full;
26
27     -- This signal is for internal connections.
28     signal C : std_logic_vector(4 downto 0);
29
30 begin  -- architecture behavior
31
32     -- Connect four components to the internal connections.
33     -- Notice that Cin is an input and Cout is an output,
34     -- so that C(0) to C(3) are connected to input pins,
35     -- and C(1) to C(4) are connected to output pins.
36     gen_full : for i in 0 to 3 generate
37         full_i : full
38             port map (
39                 A    => A(i),
40                 B    => B(i),
41                 Cin  => C(i),
42                 S    => S(i),
43                 Cout => C(i + 1));
44     end generate gen_full;
45
46     -- Connect the Cin and Cout ports.
47     -- Notice that C(0) is connected to an input pin in gen_full above,
48     -- so we need to drive it here.
49     -- On the other hand, C(4) is connected to an output pin above,
50     -- so it is already driven, but we can use it to drive Cout.
51     C(0) <= Cin;
52     Cout <= C(4);
53
54 end architecture behavior;
```

***ripple_tb.vhd**:*

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity ripple_tb is
6  end entity ripple_tb;
7
8  architecture behavior of ripple_tb is
9
10     component ripple is
11         port (
12             A    : in  std_logic_vector(3 downto 0);
13             B    : in  std_logic_vector(3 downto 0);
14             Cin  : in  std_logic;
15             S    : out std_logic_vector(3 downto 0);
16             Cout : out std_logic);
17     end component ripple;
18
19     -- inputs
20     signal A   : std_logic_vector(3 downto 0) := "0000";
21     signal B   : std_logic_vector(3 downto 0) := "0000";
22     signal Cin : std_logic                    := '0';
23
24     -- outputs
25     signal S    : std_logic_vector(3 downto 0);
26     signal Cout : std_logic;
27
28 begin  -- architecture behavior
29
30     -- unit under test
31     uut : ripple
```

```
32          port map (
33              A    => A,
34              B    => B,
35              Cin  => Cin,
36              S    => S,
37              Cout => Cout);
38
39      -- stimulus process
40      stim_proc : process is
41      begin  -- process stim_proc
42
43          A   <= "0000";
44          B   <= "0000";
45          Cin <= '0';
46          wait for 100 ns;
47
48          A   <= "0001";
49          B   <= "1111";
50          Cin <= '0';
51          wait for 100 ns;
52
53          A   <= "0001";
54          B   <= "0101";
55          Cin <= '0';
56          wait for 100 ns;
57
58          A   <= "0111";
59          B   <= "1000";
60          Cin <= '0';
61          wait for 100 ns;
62
```

```
63          A   <= "1000";
64          B   <= "0010";
65          Cin <= '1';
66          wait for 100 ns;
67
68          A   <= "0011";
69          B   <= "0111";
70          Cin <= '0';
71          wait for 100 ns;
72
73          A   <= "1011";
74          B   <= "0111";
75          Cin <= '0';
76          wait for 100 ns;
77
78          A   <= "1011";
79          B   <= "0111";
80          Cin <= '1';
81          wait for 100 ns;
82
83          A   <= "1111";
84          B   <= "1111";
85          Cin <= '0';
86          wait for 100 ns;
87
88          A   <= "1111";
89          B   <= "1111";
90          Cin <= '1';
91          wait for 100 ns;
92
93      end process stim_proc;
94  end architecture behavior;
```

full_lookahead.vhd:

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

entity full_lookahead_tb is
end entity full_lookahead_tb;

architecture behavior of full_lookahead_tb is

    component full_lookahead is
        port (
            A   : in  std_logic;
            B   : in  std_logic;
            Cin : in  std_logic;
            S   : out std_logic;
            G   : out std_logic;
            P   : out std_logic);
    end component full_lookahead;

    -- inputs
    signal A   : std_logic := '0';
    signal B   : std_logic := '0';
    signal Cin : std_logic := '0';

    -- outputs
    signal S : std_logic;
    signal G : std_logic;
    signal P : std_logic;

begin  -- architecture behavior


    -- unit under test
    uut : full_lookahead
        port map (
            A   => A,
            B   => B,
            Cin => Cin,
            S   => S,
            G   => G,
            P   => P);

    -- stimulus process
    stim_proc : process is

        variable u : unsigned(2 downto 0);

    begin  -- process stim_proc
        -- include 8 in loop so that the 3 inputs go back to 0
        for i in 0 to 8 loop
            -- convert i to a 3-bit vector
            u := to_unsigned(i, 3);

            A   <= u(2);
            B   <= u(1);
            Cin <= u(0);

            wait for 100 ns;
        end loop;

    end process stim_proc;

end architecture behavior;
```

full_lookahead_tb.vhd:

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity full_lookahead_tb is
6  end entity full_lookahead_tb;
7
8  architecture behavior of full_lookahead_tb is
9
10     component full_lookahead is
11         port (
12             A   : in  std_logic;
13             B   : in  std_logic;
14             Cin : in  std_logic;
15             S   : out std_logic;
16             G   : out std_logic;
17             P   : out std_logic);
18     end component full_lookahead;
19
20     -- inputs
21     signal A   : std_logic := '0';
22     signal B   : std_logic := '0';
23     signal Cin : std_logic := '0';
24
25     -- outputs
26     signal S : std_logic;
27     signal G : std_logic;
28     signal P : std_logic;
29
30 begin  -- architecture behavior
31
32     -- unit under test
33     uut : full_lookahead
34         port map (
35             A   => A,
36             B   => B,
37             Cin => Cin,
38             S   => S,
39             G   => G,
40             P   => P);
41
42     -- stimulus process
43     stim_proc : process is
44
45         variable u : unsigned(2 downto 0);
46
47     begin  -- process stim_proc
48
49         -- include 8 in loop so that the 3 inputs go back to 0
50         for i in 0 to 8 loop
51             -- convert i to a 3-bit vector
52             u := to_unsigned(i, 3);
53
54             A   <= u(2);
55             B   <= u(1);
56             Cin <= u(0);
57
58             wait for 100 ns;
59         end loop;
60
61     end process stim_proc;
62 end architecture behavior;
```

lookahead.vhd:

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

entity lookahead 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 entity lookahead;

architecture behavior of lookahead is

    constant delay : time := 1 ns;

    component full_lookahead is
        port (
            A   : in  std_logic;
            B   : in  std_logic;
            Cin : in  std_logic;
            S   : out std_logic;
            G   : out std_logic;
            P   : out std_logic);
    end component full_lookahead;

    -- These signals are for internal connections.
    signal C : std_logic_vector(4 downto 0);
    signal G : std_logic_vector(3 downto 0);
    signal P : std_logic_vector(3 downto 0);

begin  -- architecture behavior

    -- Connect four components to the internal connections.
    gen_full_lookahead : for i in 0 to 3 generate
        full_lookahead_i : full_lookahead
            port map (
                A   => A(i),
                B   => B(i),
                Cin => C(i),
                S   => S(i),
                G   => G(i),
                P   => P(i));
    end generate gen_full_lookahead;

    -- Connect the Cin and Cout ports.
    C(0) <= Cin;
    Cout <= C(4);

    -- Carry-lookahead logic
    C(1) <= G(0) or (C(0) and P(0)) after 2 * delay;
    C(2) <= G(1) or (G(0) and P(1)) or (C(0) and P(0) and P(1)) after 2 * delay;
    C(3) <= G(2) or (G(1) and P(2)) or (G(0) and P(1) and P(2))
                 or (C(0) and P(0) and P(1) and P(2)) after 2 * delay;
    C(4) <= G(3) or (G(2) and P(3)) or (G(1) and P(2) and P(3))
                 or (G(0) and P(1) and P(2) and P(3))
                 or (C(0) and P(0) and P(1) and P(2) and P(3)) after 2 * delay;

end architecture behavior;
```

lookahead_tb.vhd:

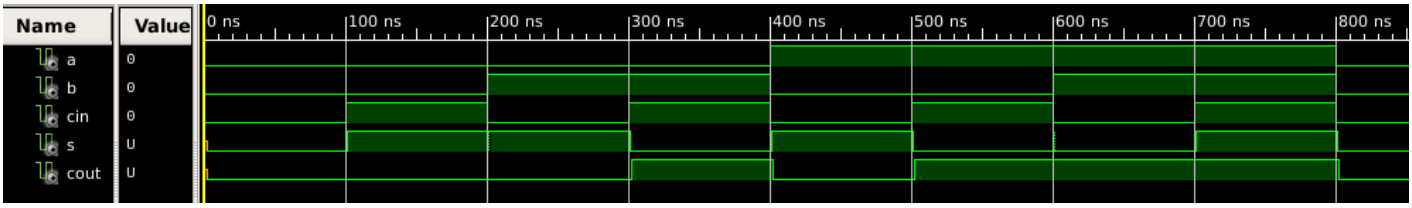
```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity lookahead_tb is
6  end entity lookahead_tb;
7
8  architecture behavior of lookahead_tb is
9
10     component lookahead is
11         port (
12             A    : in  std_logic_vector(3 downto 0);
13             B    : in  std_logic_vector(3 downto 0);
14             Cin  : in  std_logic;
15             S    : out std_logic_vector(3 downto 0);
16             Cout : out std_logic);
17     end component lookahead;
18
19     -- inputs
20     signal A   : std_logic_vector(3 downto 0) := "0000";
21     signal B   : std_logic_vector(3 downto 0) := "0000";
22     signal Cin : std_logic                    := '0';
23
24     -- outputs
25     signal S    : std_logic_vector(3 downto 0);
26     signal Cout : std_logic;
27
28 begin  -- architecture behavior
29
30     -- unit under test
31     uut : lookahead
```

```
32          port map (
33              A    => A,
34              B    => B,
35              Cin  => Cin,
36              S    => S,
37              Cout => Cout);
38
39      -- stimulus process
40      stim_proc : process is
41      begin  -- process stim_proc
42
43          A   <= "0000";
44          B   <= "0000";
45          Cin <= '0';
46          wait for 100 ns;
47
48          A   <= "0001";
49          B   <= "1111";
50          Cin <= '0';
51          wait for 100 ns;
52
53          A   <= "0001";
54          B   <= "0101";
55          Cin <= '0';
56          wait for 100 ns;
57
58          A   <= "0111";
59          B   <= "1000";
60          Cin <= '0';
61          wait for 100 ns;
62
```

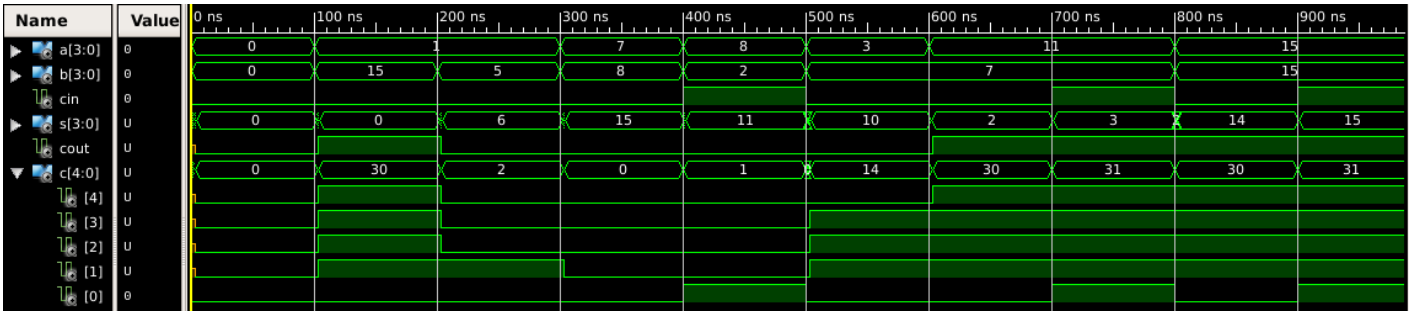
```
63          A   <= "1000";
64          B   <= "0010";
65          Cin <= '1';
66          wait for 100 ns;
67
68          A   <= "0011";
69          B   <= "0111";
70          Cin <= '0';
71          wait for 100 ns;
72
73          A   <= "1011";
74          B   <= "0111";
75          Cin <= '0';
76          wait for 100 ns;
77
78          A   <= "1011";
79          B   <= "0111";
80          Cin <= '1';
81          wait for 100 ns;
82
83          A   <= "1111";
84          B   <= "1111";
85          Cin <= '0';
86          wait for 100 ns;
87
88          A   <= "1111";
89          B   <= "1111";
90          Cin <= '1';
91          wait for 100 ns;
92
93      end process stim_proc;
94  end architecture behavior;
```

Simulated Behavioral Models:

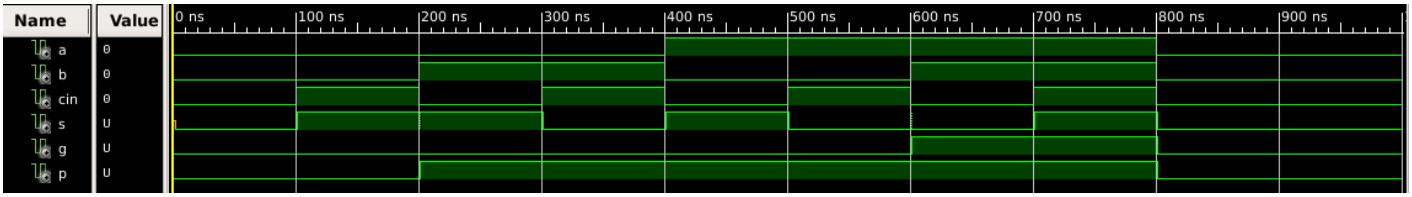
full_tb.vhd:



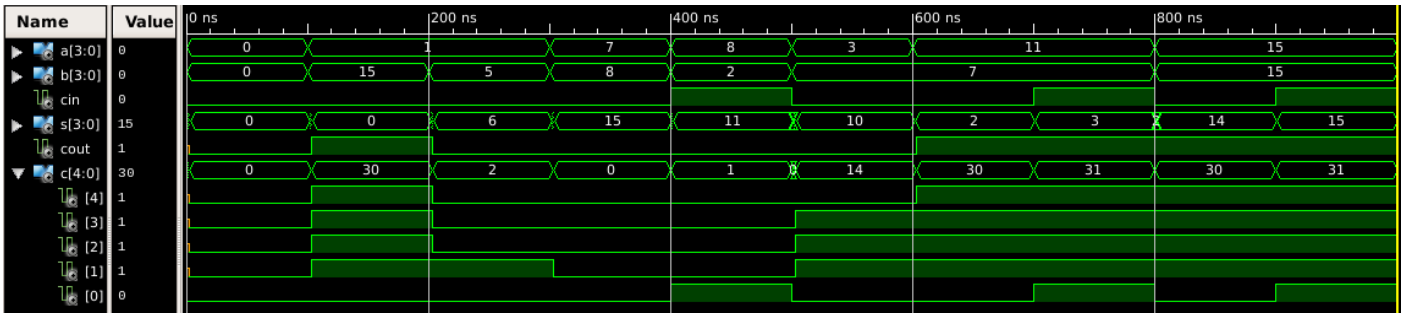
ripple_tb.vhd:



full_lookahead_tb.vhd:



lookahead_tb.vhd:



Conclusion:

A carry-ripple adder and a carry-lookahead adder were successfully built, tested and compared.