ECE 585 – Computer Organization and Design

Project 1 – 32-bit Full Adder

I acknowledge all works including figures, codes and writings belong to me and/or persons who are referenced. I understand if any similarity in the code, comments, customized program behavior, report writings and/or figures are found, both the helper (original work) and the requestor (duplicated/modified work) will be called for academic disciplinary action.

Anand Baraguru Venkateshmurthy

(10/28/2019)

Abstract

The main objective of the study is to design and implement Ripple Carry Adder (RCA) and Carry Look Ahead Adder (CLA). In this study I have used Xilinx ISE (WebPack) VHDL simulator for implementing 1-bit, 4-bit, 32-bit RCA and 4-bit, 16-bit and 32-bit CLA.

In this study, I have tested the operation of the above said RCA and CLA by inputting different values and compared the simulation results between them. This study also records simulation results by capturing the screen of simulation output with inputs used and the corresponding output data.

Introduction

The purpose of the project is to get familiarize with the VHDL programming and simulation environment by designing and implementing a 32-bit adder.

First we get started with 1-bit Full Adder and then based on it we design and implement 4-bit and 32-bit Ripple carry Adder (RCA) and 4-bit, 16-bit and 32-bit Carry Look Ahead Adder (CLA). 1-bit Full Adder (FA) consists of 3 inputs (a, b and Carryin) and 2 outputs (Sum and Carryout). 4-bit RCA is implemented by cascading four 1-bit Full Adders. In RCA, each FA's sum is determined only after carry generated by the previous stage FA is produced. This results in delay. To speed up addition, carry into the higher order bits should be determined sooner so that the carry bits will be ready to calculate the sum faster. For this CLA is used to compute carries in advance.

Background

i) Description of full-adder

1-bit Full Adder consists of 3 inputs (a, b and Carryin) and 2 outputs (Sum and Carryout) as shown in figure 1 and it's truth table is shown in figure 2.

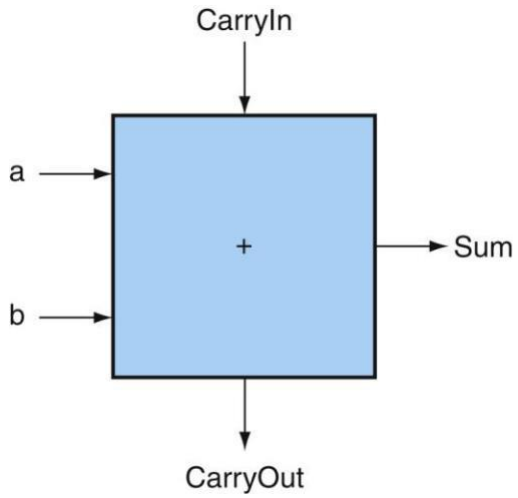

Figure 1. 1-bit Full Adder

Inputs			Outputs		
a	b	CarryIn	CarryOut	Sum	Comments
0	0	0	0	0	$0 + 0 + 0 = 00_{two}$
0	0	1	0	1	$0 + 0 + 1 = 01_{two}$
0	1	0	0	1	$0 + 1 + 0 = 01_{two}$
0	1	1	1	0	$0 + 1 + 1 = 10_{two}$
1	0	0	0	1	$1 + 0 + 0 = 01_{two}$
1	0	1	1	0	$1 + 0 + 1 = 10_{two}$
1	1	0	1	0	$1 + 1 + 0 = 10_{two}$
1	1	1	1	1	$1 + 1 + 1 = 11_{two}$

Figure 2. Truth Table of 1-bit Full Adder

A Ripple Carry Adder (RCA) is a combinational logic circuit which performs addition of two n-bit binary numbers by cascading n-Full Adders. Figure 3 shows 4-bit RCA. But the drawback of RCA is it's delay. Because each Full Adders's sum is determined after carry generated by the previous stage Full Adder is produced.

To speed up the addition, carry into the higher order bits should be determined sooner so that carry bits required to find sum will be available sooner. For this,

Carry Lookahead Adder (CLA) is used. CLA computes carry in advance. The carry logic equation is

$$c_{i+1}=a_ib_i+(a_i+b_i)c_i$$

where,

a_ib_i is called generate signal

a_i+b_i is called propagate signal

Thus the equation can be re-written as

$$c_{i+1}=g_i+p_ic_i$$

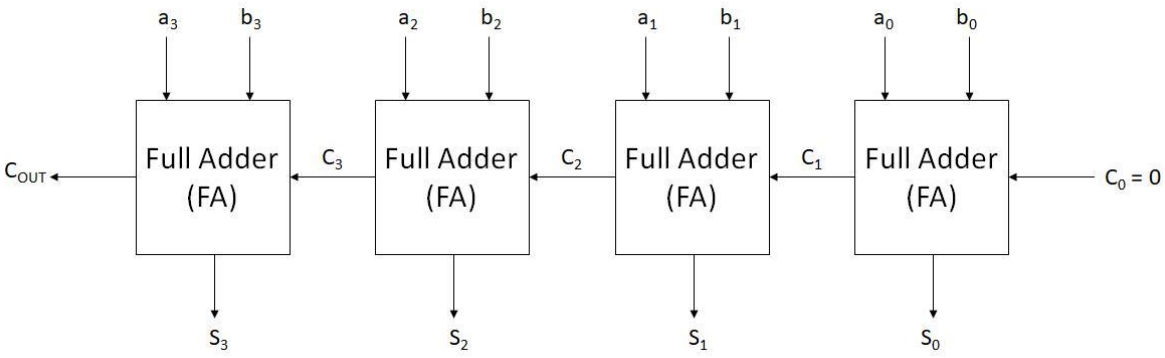

Figure 3. 4-bit Ripple Carry Adder using four 1-bit Full Adder

For a 4-bit CLA, carries are calculated as:

For a 4-bit CLA, carries are calculated as the following:

$c1=g0+p0c0$

$c2=g1+p1c1=g1+p1(g0+p0c0)$

$=g1+p1g0+p1p0c0$

$c3=g2+p2c2=g2+p2(g1+p1g0+p1p0c0)$

$=g2+p2g1+p2p1g0+p2p1p0c0$

$c4=g3+p3c3=g3+p3(g2+p2g1+p2p1g0+p2p1p0c0)$

$=g3+p3g2+p3p2g1+p3p2p1g0+p3p2p1p0c0$

Figure 4 shows 4-bit CLA

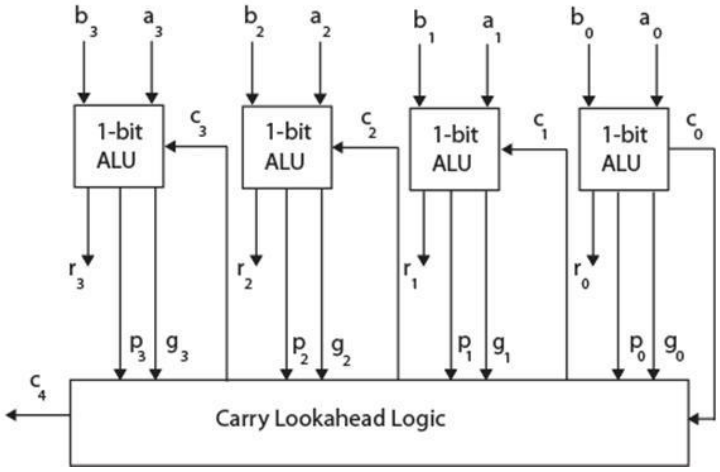

Figure 4. 4-bit CLA

By cascading 4-bit CLAs, 16-bit CLA is implemented which is shown in figure 5.

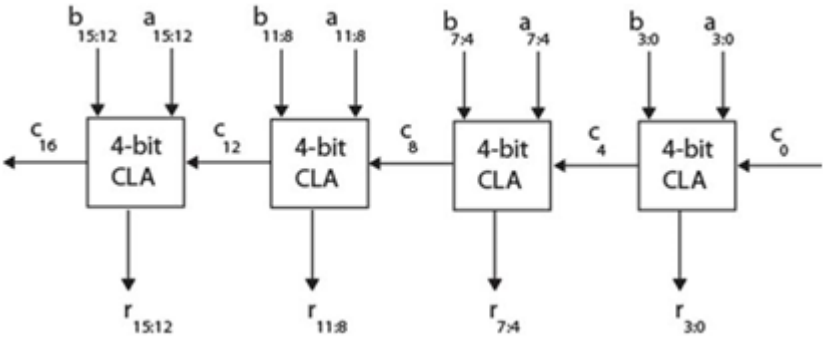

Figure 5. 16-bit CLA

ii) Description of Xilinx ISE simulator and environment.

Xilinx ISE simulator is a Hardware Description Language simulator that allows to perform functional and timing simulations for VHDL, Verilog and mixed language designs. I have used ISE 14.7 (WebPack) throughout this project.

Hardware requirements:

CPU speed: 2.2 GHz minimum or higher

Processor: Intel Pentium 4, Intel Core Duo, or Xeon Processors.

Memory/RAM: 2 GB or higher.

Operating System:

Windows: Windows 7, 8.1, 10

Simulation Results and discussion

Part 1

1. 1-bit RCA simulation result

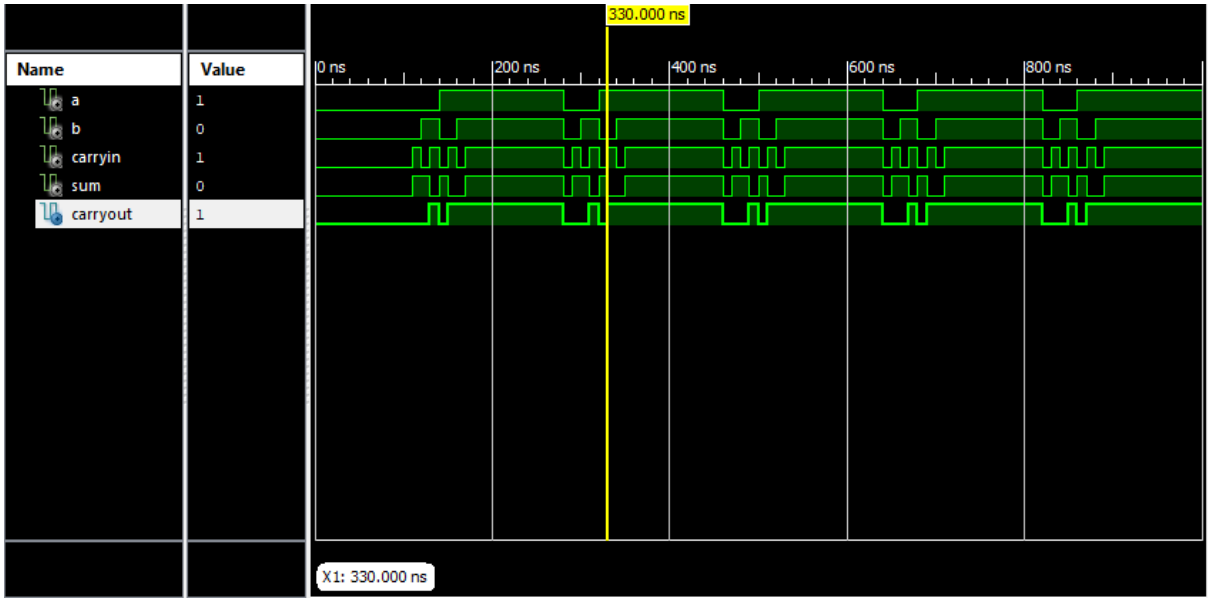

Figure 6.

Inputs: a=1, b=0, Carryin=1

Outputs: Sum=0, Carryout=1

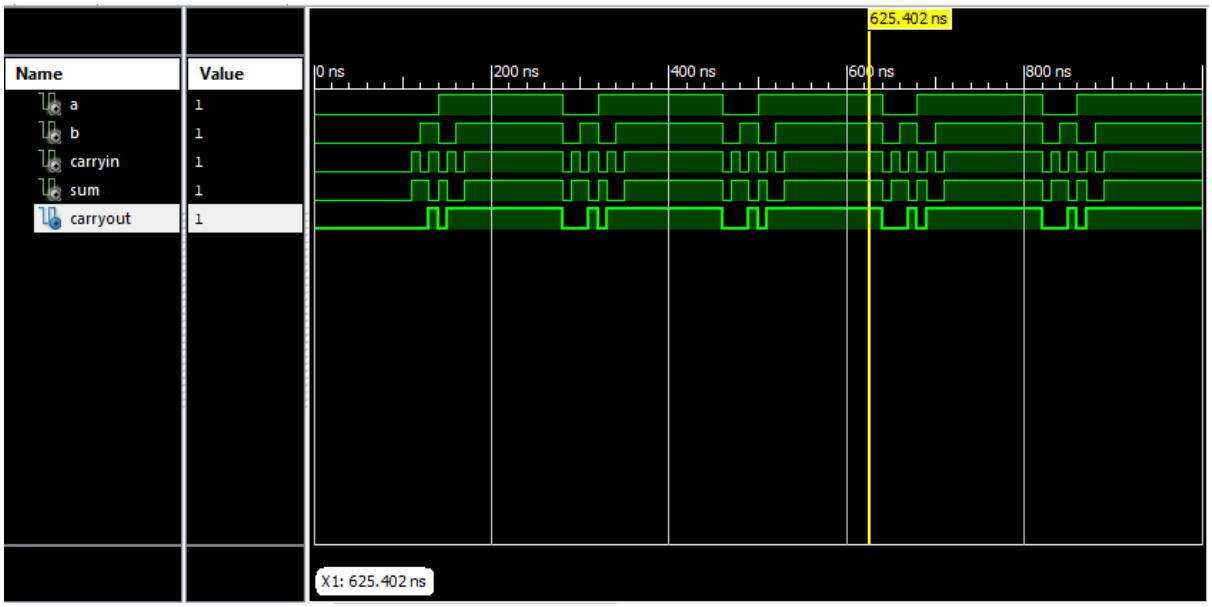

Figure 7.

Inputs: a=1, b=1, Carryin=1

Outputs: Sum=1, Carryout=1

2. 4-bit RCA simulation result

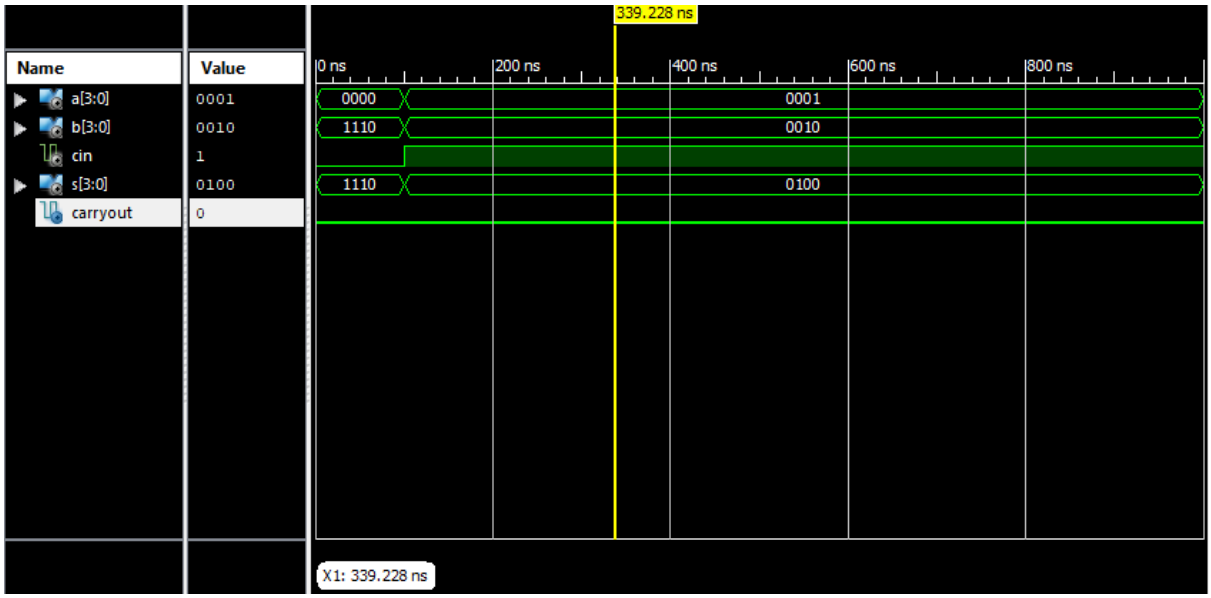

Figure 8.

Inputs: a=0001, b=0010, cin=1

Outputs: S=0100, carryout=0

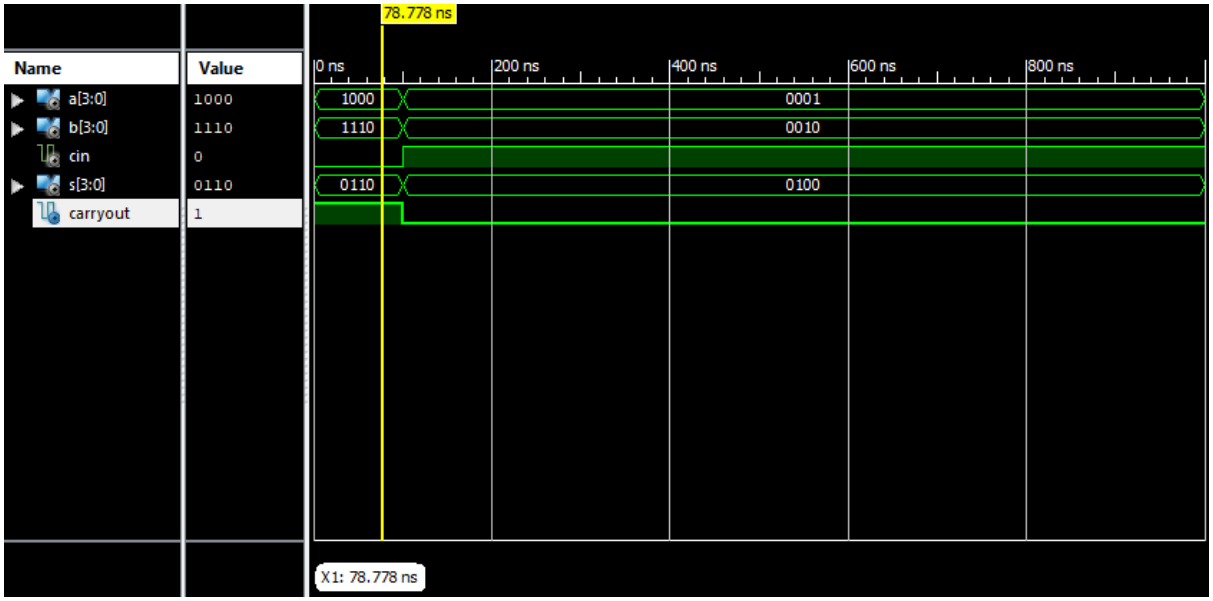

Figure 9.

Inputs: a=1000, b=1110, cin=0

Outputs: s=0110, carryout=1

3. 32-bit RCA simulation result

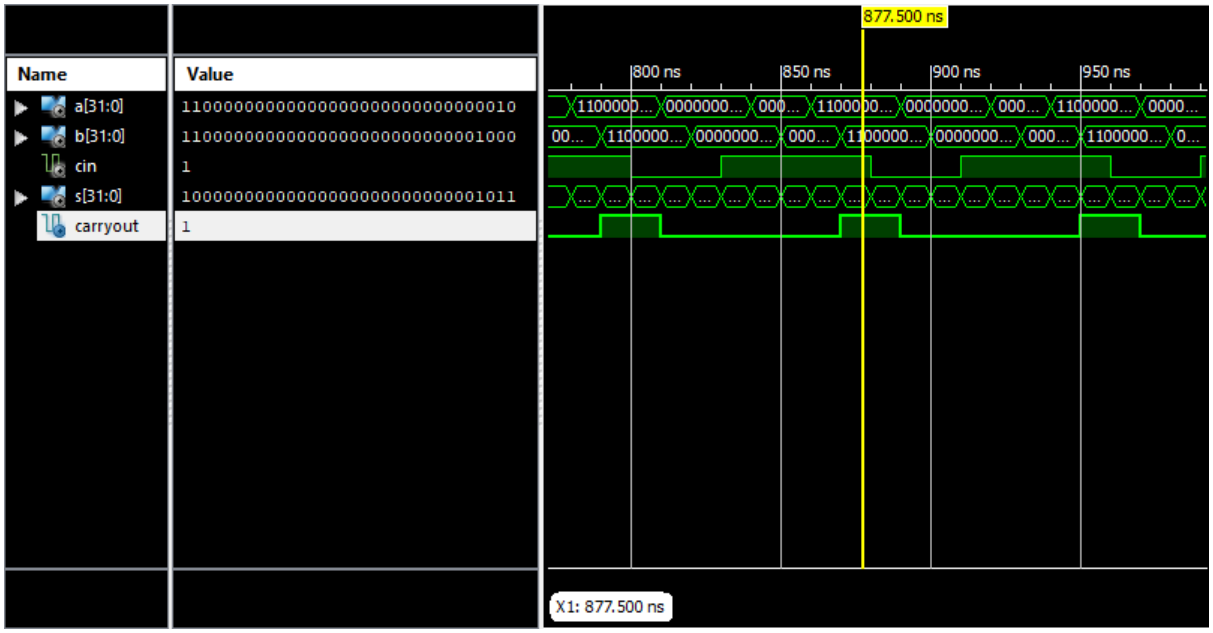

Figure 10.

Inputs: A=1100 0000 0000 0000 0000 0000 0000 0010

B=1100 0000 0000 0000 0000 0000 0000 1000

Cin=1

Outputs: S=1000 0000 0000 0000 0000 0000 0000 1011

Carryout=1

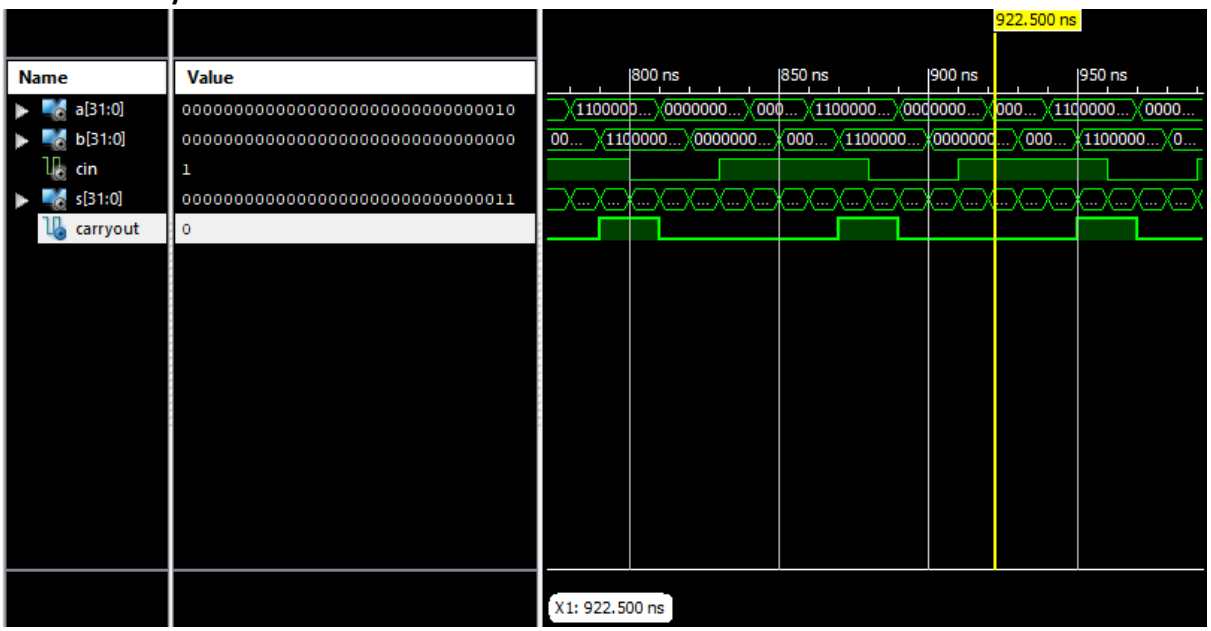

Figure 11.

Inputs: A=0000 0000 0000 0000 0000 0000 0000 0010

B=0000 0000 0000 0000 0000 0000 0000 0000

Cin=1

Outputs: sum=0000 0000 0000 0000 0000 0000 0000 0011

Carryout=0

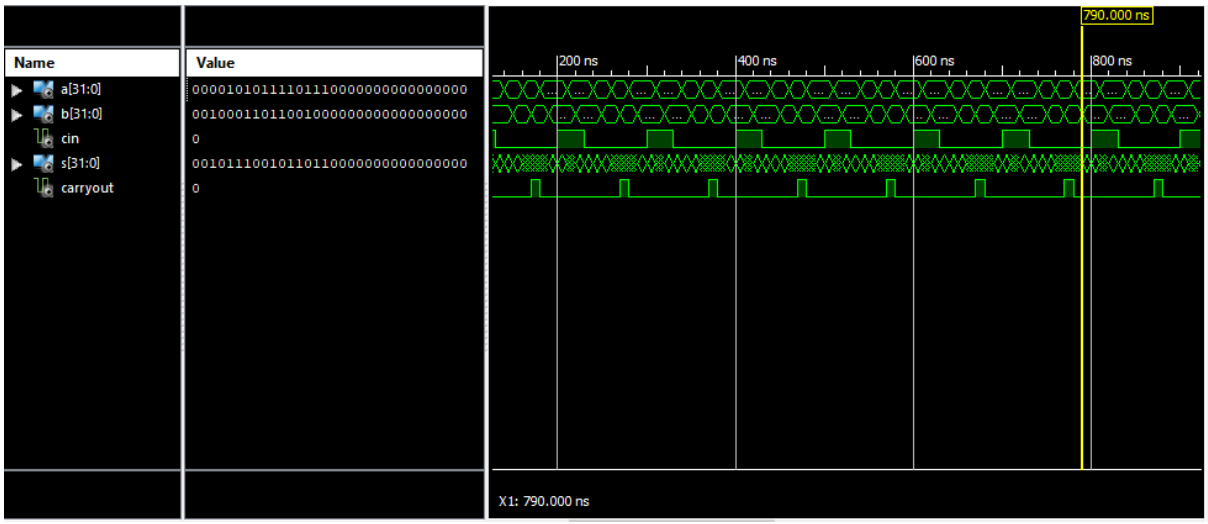

Figure 12.

Inputs: First four: a=2042 -> in hexadecimal = 0AF7 -> in binary= 0000 1010 1111 0111 0000 0000 0000 0000

Last four: b=9060 -> in hexadecimal = 2364 -> in binary= 0010 0011 0110 0100 0000 0000 0000 0000

Cin = 0;

Outputs: Sum=0010 1110 0101 1011 0000 0000 0000 0000

Carryout=0

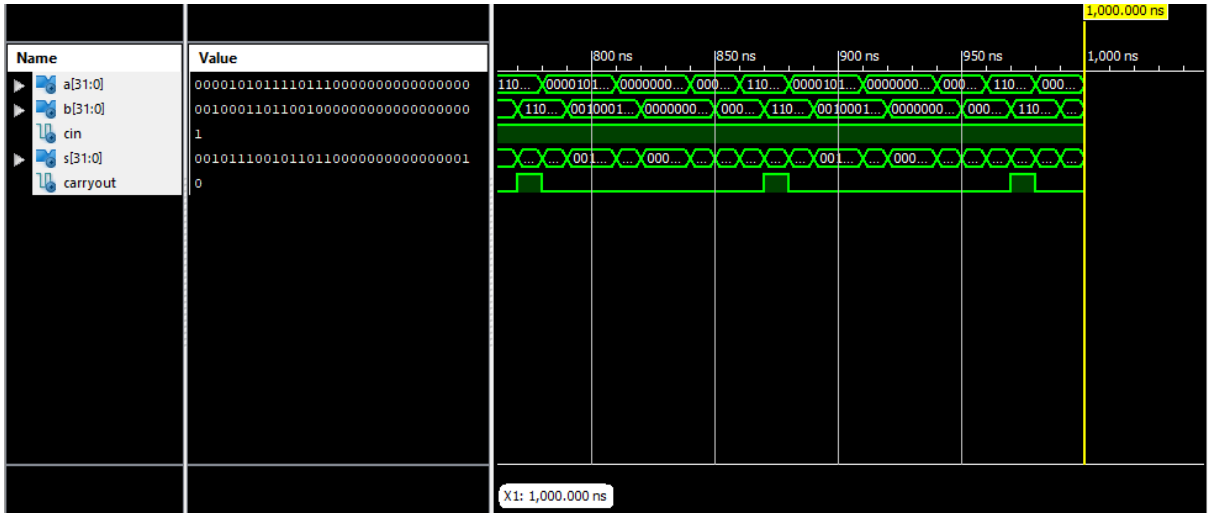

Figure 13.

Inputs: First four: a=2042 -> in hexadecimal = 0AF7 -> in binary= 0000 1010 1111 0111 0000 0000 0000 0000

Last four: b=9060 -> in hexadecimal = 2364 -> in binary= 0010 0011 0110 0100 0000 0000 0000 0000

Cin = 1;

Outputs: Sum=0010 1110 0101 1011 0000 0000 0000 0001

Carryout=1

Simulation result comparison of 1-bit, 4-bit and 32-bit RCA

On comparing the simulation results of 1-bit, 4-bit and 32-bit RCA, delay of 1-bit RCA is less than delay of 4-bit RCA and delay of 4-bit RCA is less than delay of 32-bit RCA. So, 1-bit RCA is faster than 4-bit RCA and 4-bit RCA is faster than 32-bit RCA. Area needed for 1-bit RCA is less than 4-bit RCA and that of 4-bit RCA is less than 32-bit RCA.

Part 2

1. 4-bit CLA simulation result.

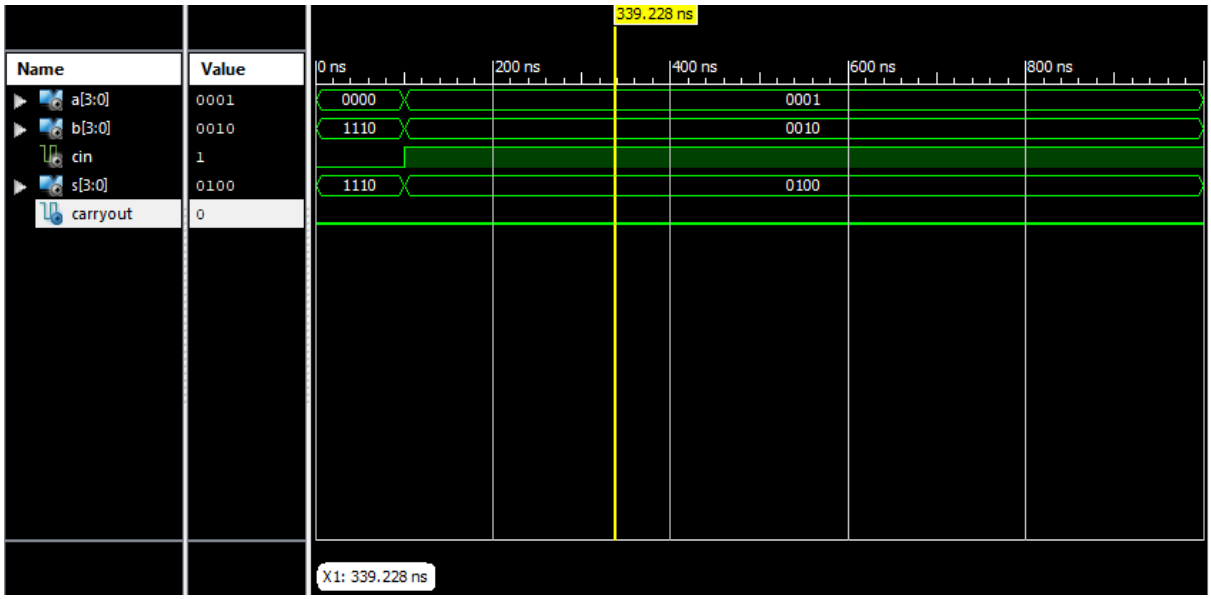

Figure 14

Inputs: a=0001, b=0010, cin=1

Outputs: S=0100, carryout=0

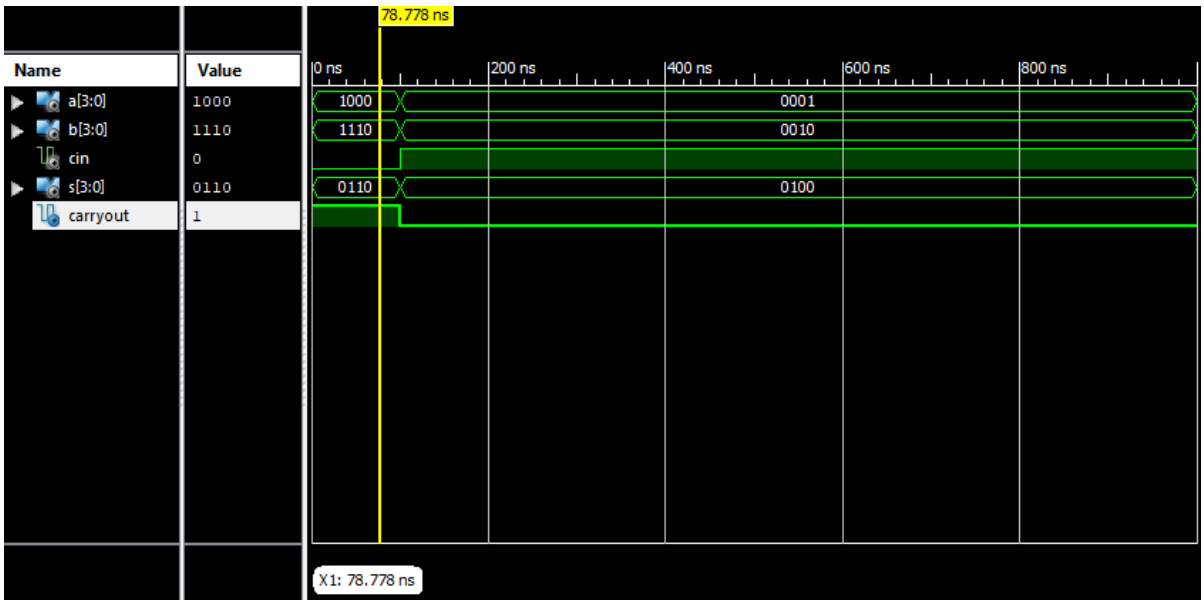

Figure 15

Inputs: a=1000, b=1110, cin=0

Outputs: s=0110, carryout=1

2. 16-bit CLA simulation result.

Figure 16

Inputs: a= 1100 0000 0000 0110

b= 1100 0000 0000 1011

carry_in=0

Outputs: sum=1000 0000 0001 0001

Carry_out=1

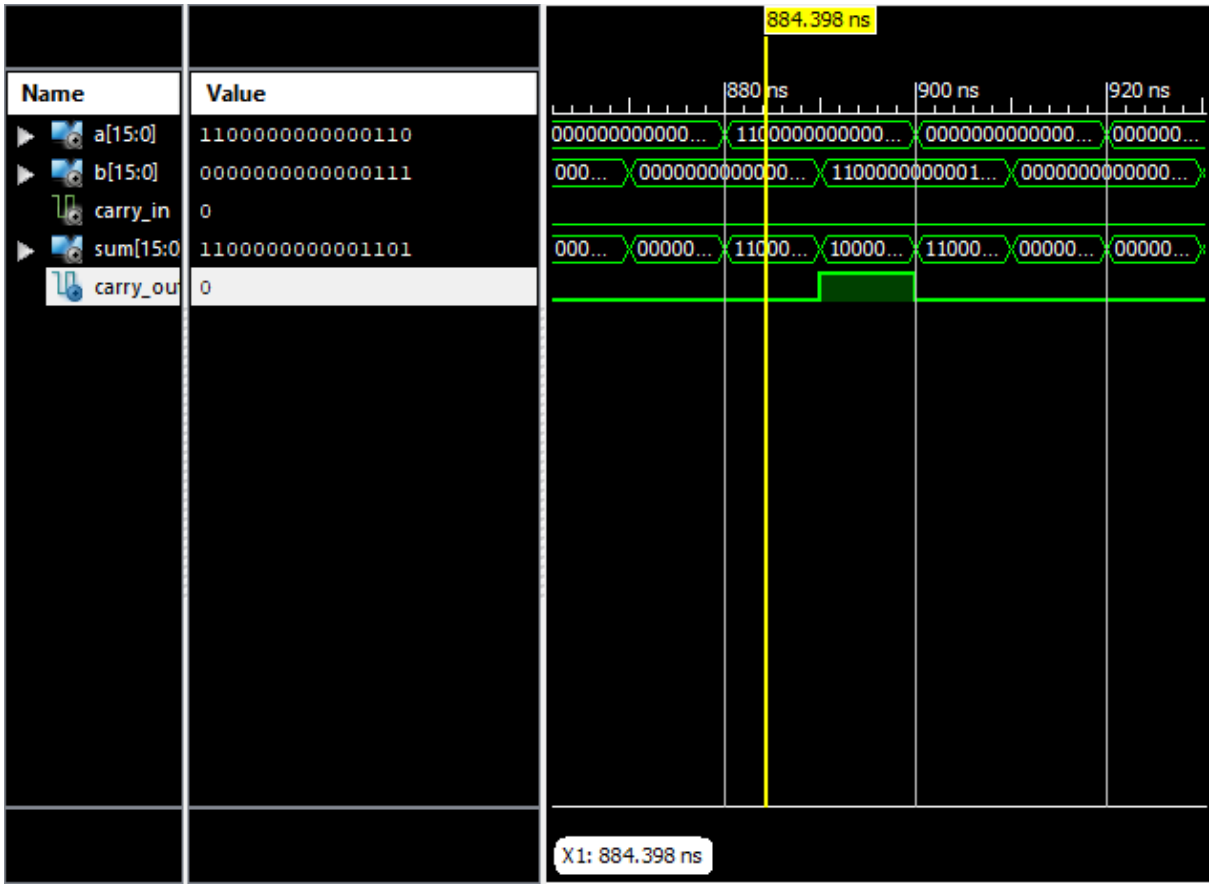

Figure 17

Inputs: a=1100 0000 0000 0110

b= 0000 0000 0000 0111

carry_in=0

outputs: sum=1100 0000 0000 1101

carry_out=0

3. 32-bit CLA simulation result.

Figure 18

Inputs: a=1100 0000 0000 0000 0000 0000 0000 0010

b=0000 0000 0000 0000 0000 0000 0000 1000

carry_in=1

Outputs: sum=1100 0000 0000 0000 0000 0000 0000 1011

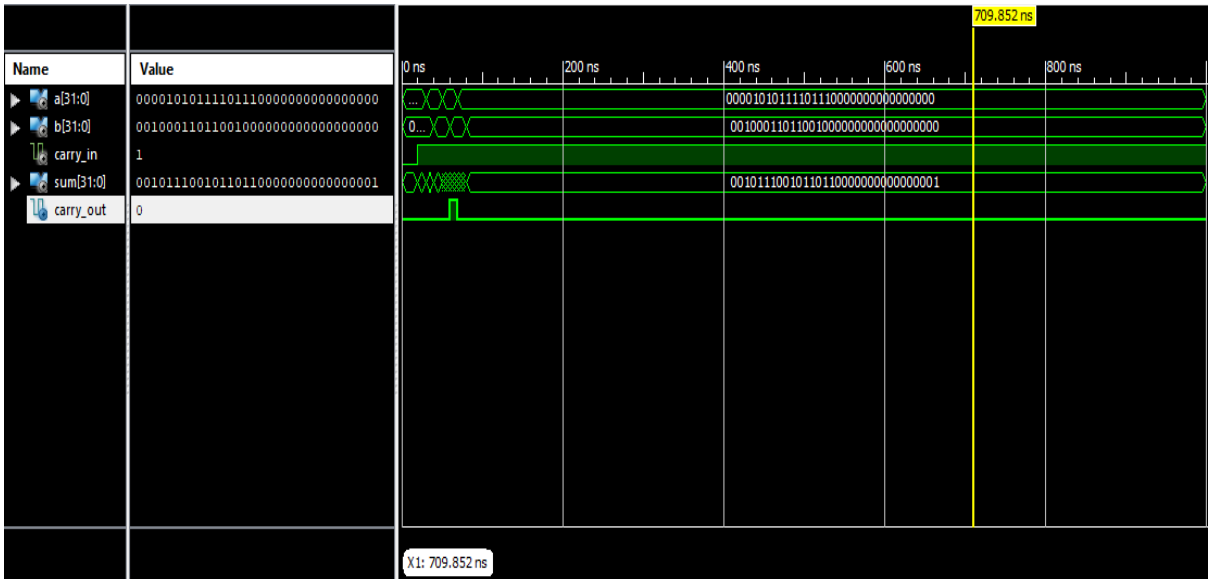

Figure 19

Inputs: First four: a=2042 -> in hexadecimal = 0AF7 -> in binary= 0000 1010 1111 0111 0000 0000 0000 0000

Last four: b=9060 -> in hexadecimal = 2364 -> in binary= 0010 0011 0110 0100 0000 0000 0000 0000

Carry_in = 1;

Outputs: Sum=0010 1110 0101 1011 0000 0000 0000 0001

Carryout=0

Simulation result comparison of 4-bit, 16-bit and 32-bit CLA

On comparing the simulation results of 4-bit, 16-bit and 32-bit CLA, delay of 4-bit CLA is less than delay of 16-bit CLA and delay of 16-bit CLA is less than delay of 32-bit CLA. So, 4-bit CLA is faster than 16-bit CLA and 16-bit CLA is faster than 32-bit CLA. Area needed for 4-bit CLA is lesser than 16-bit CLA and that of 16-bit CLA is less than 32-bit CLA.

Simulation result comparison of 32-bit RCA and 32-bit CLA

The 32-bit RCA is compact in design but takes longer computation time. 32-bit CLA need larger area compared to 32-bit RCA but it is faster than CLA. So time critical applications use CLA.

CONCLUSION

In this project, I got familiar with the VHDL programming, as well as the simulation environment. I learnt the designing, implementation and operation of 32-bit Full Adders using VHDL programming. I understood the operation of 1-bit full adder and how cascading is used to design 4-bit, 16-bit and 32-bit Full Adders.

1-bit Full adder has only 3 inputs. To add n-bits Ripple Carry Adder (RCA) is used. But the drawback in RCA is the sum can be determined only after the carry generated by previous stage adder is produced. This drawback is overcomed by using Carry Lookahead Adder (CLA) in which carries are computed in advance and available sooner to determine the sum faster.

REFERENCES:

[1] https://www.xilinx.com/support/documentation/sw_manuals/xilinx11/ug682.pdf

Reference [1] helped me in installing, setting up Xilinx ISE and how to obtain license.

[2]https://www.xilinx.com/support/documentation/sw_manuals/xilinx14_1/ise_tutorial_ug695.pdf

[3] http://www.eng.ucy.ac.cy/theocharides/Courses/ECE408/qst.pdf

Reference [2] and [3] helped me in creating a new project, understand different vhdl modelling styles with examples and design implementation, writing VHDL codes, test bench codes and how to simulate them.

APPENDIX

Based on the given information, design a 1-bit RCA using Behavioral Model.

VHDL code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;      -- Using STD_LOGIC_1164 package

entity onebitrca is
port (a : in STD_LOGIC;                -- a is used as input port
       b : in STD_LOGIC;               -- b is used as input port
       Carryin : in STD_LOGIC;         -- Carryin is used as input port
       Sum : out STD_LOGIC;            -- Sum is used as output port
       Carryout : out STD_LOGIC);      -- Carryout is used as output port
end onebitrca;

architecture Behavioral of onebitrca is
begin
Sum <= a xor b xor Carryin;                                        -- Sum Equation
Carryout <= (a and b) or (Carryin and b) or (Carryin and a);      -- Carryout equation
end Behavioral;
```

Test Bench:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;                -- Using STD_LOGIC_1164 package

ENTITY onebitrca_tb IS
END onebitrca_tb;

ARCHITECTURE behavior OF onebirrca_tb IS

 COMPONENT onebitrca                    -- Component declaration
 PORT(
       a : IN  std_logic;               -- a is declared as input port
       b : IN  std_logic;               -- b is declared as input port
       Carryin : IN  std_logic;         -- Carryin is declared as input port
       Sum : OUT  std_logic;            -- Sum is declared as output port
       Carryout : OUT  std_logic        -- Carryput is declared as output port
       );
   END COMPONENT

            -- Signal declaration

  signal a : std_logic := '0';
  signal b : std_logic := '0';
  signal Carryin : std_logic := '0';
  signal Sum : std_logic;
  signal Carryout : std_logic;
```

```
BEGIN
 -- Instantiate the Unit Under Test (UUT)
  uut: onebitrca PORT MAP (
        a => a,
        b => b,
        Carryin => Carryin,
        Sum => Sum,
        Carryout => Carryout
       );
   --Process name :   stim_proc
   -- This process tests the implementation by possible input values
  stim_proc: process
  begin
    wait for 100 ns;
     a <= '0';
     b <= '0';
     Carryin <= '0';
     wait for 10 ns;

     a <= '0';
     b <= '0';
    Carryin <= '1';
    wait for 10 ns;

    a <= '0';
    b <= '1';
    Carryin <= '0';
    wait for 10 ns;

    a <= '0';
    b <= '1';
    Carryin <= '1';
    wait for 10 ns;

    a <= '1';
    b <= '0';
    Carryin <= '0';
    wait for 10 ns;

    a <= '1';
    b <= '0';
    Carryin <= '1';
    wait for 10 ns;

    a <= '1';
    b <= '1';
    Carryin <= '0';
```

```
    wait for 10 ns;

    a <= '1';
    b <= '1';
    Carryin <= '1';
    wait for 10 ns;

    end process;
    END;
```

Design a 4-bit RCA using Structural Model based on your 1-bit RCA.

VHDL code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;                    -- Using STD_LOGIC_LIBRARY_1164 package
          entity four_bit_rca is
port( A:IN STD_LOGIC_VECTOR(3 downto 0);   -- Declaring A as four bit input port
       B:IN STD_LOGIC_VECTOR(3 downto 0);   -- Declaring B as four bit input port
       Cin:IN STD_LOGIC;                    -- Declaring Cin as input port
       S:out STD_LOGIC_VECTOR(3 downto 0); -- Declaring S as four bit output port
       Carryout:out std_logic               -- Declaring Carryout as output port
       );
end fourbitrca;
          architecture Structural of four_bit_rca is        -- architecture declaration
 --Component declaration
component fa is
port(a,b,c:in std_logic;                    -- Declaring a, b and c as inputs
      Cout:out std_logic;                   --  Declaring Cout as output
      s:out std_logic                       --  Declaring s as output
);
end component;
 signal carry : STD_LOGIC_VECTOR (3 downto 0);
 begin
 --Port mapping
fa0:fa port map(a=>A(0), b=>B(0), c=>Cin, Cout=>carry(0), s=>S(0));
fa1:fa port map(a=>A(1), b=>B(1), c=>carry(0), Cout=>carry(1), s=>S(1));
fa2:fa port map(a=>A(2), b=>B(2), c=>carry(1), Cout=>carry(2), s=>S(2));
fa3:fa port map(a=>A(3), b=>B(3), c=>carry(2), Cout=>Carryout, s=>S(3));
end Structural;
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;            -- Using STD_LOGIC_LIBRARY_1164 package
entity fa is
port( a,b,c:in std_logic;                   --  Declaring a, b and c as inputs
     Cout:out std_logic;                    --  Declaring Cout as output
               s:out std_logic              --  Declaring s as output
            );
end fa;
```

```
architecture fa_architecture of fa is          -- architecture declaration
begin
s<=a xor b xor c;                              -- Sum result expression
Cout<=(a and b) or (c and a) or (c and b);   -- Cout expression
end fa_architecture;
```

Test Bench:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;                  -- Using STD_LOGIC_LIBRARY_1164 package
ENTITY four_bit_rca_tb IS                     -- fourbit rca Entity declaration
END four_bit_rca_tb;
ARCHITECTURE behavior OF four_bit_rca_tb IS          -- architecture declaration
   -- Component Declaration for the Unit Under Test (UUT)
 COMPONENT fourbitrca
  PORT(A : IN  std_logic_vector(3 downto 0);             -- A is used as input port
             B : IN  std_logic_vector(3 downto 0);          -- B is used as input port
             Cin : IN  std_logic;                                  -- Cin is used as input port
              S : OUT  std_logic_vector(3 downto 0);       -- S is used as output port
              Carryout : OUT  std_logic                        -- Carryout is used as output port
      );
   END COMPONENT;
 --Inputs
  signal A : std_logic_vector(3 downto 0) := (others => '0');
  signal B : std_logic_vector(3 downto 0) := (others => '0');
  signal Cin : std_logic := '0';
           --Outputs
  signal S : std_logic_vector(3 downto 0);
  signal Carryout : std_logic;
  BEGIN
  uut: fourbitrca PORT MAP (
        A => A,
        B => B,
        Cin => Cin,
        S => S,
        Carryout => Carryout
      );
   stim_proc_A: process
  begin
   A<="0000";
   wait for 100 ns;
   A<="0001";
   wait;
   end process;
   stim_proc_B: process
   begin
   B<="1110";
   wait for 100 ns;
```

```
B<="0010";
 wait;
end process;
stim_proc_Cin: process
begin
Cin<='0';
 wait for 100 ns;
Cin<='1';
wait;
end process;
END;
```

Design a 32-bit RCA using Structural model based on your 4-bit RCA

VHDL code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;                    -- Using STD_LOGIC_LIBRARY_1164 package

USE ieee.std_logic_arith.ALL;                   -- Using STD_LOGIC_LIBRARY_arith package

entity thirtytwo_bit_rca is                     -- Entity declaration

port(A:IN STD_LOGIC_VECTOR(31 downto 0);        -- Declaring A as 32 bit input port
B:IN STD_LOGIC_VECTOR(31 downto 0);             -- Declaring B as 32 bit input port
Cin:IN STD_LOGIC;                               -- Declaring Cin as input port
S:OUT STD_LOGIC_VECTOR(31 downto 0);            -- Declaring S as 32 bit output port
Carryout:OUT std_logic                          -- Declaring Carryout as output port
);
end thirtytwo_bit_rca;

architecture Structural of thirtytwo_bit_rca is
--Component Declaration
component fa is
port(a,b,c:in std_logic;                        -- a,b and c is used as input port
Cout:out std_logic;                             -- Cout is used as output port
s:out std_logic                                 -- s is used as output port
);
end component;
signal carry : STD_LOGIC_VECTOR (30 downto 0);
begin
--Port mapping
fa0:fa port map(a=>A(0), b=>B(0), c=>Cin, Cout=>carry(0), s=>S(0));
fa1:fa port map(a=>A(1), b=>B(1), c=>carry(0), Cout=>carry(1), s=>S(1));
fa2:fa port map(a=>A(2), b=>B(2), c=>carry(1), Cout=>carry(2), s=>S(2));
fa3:fa port map(a=>A(3), b=>B(3), c=>carry(2), Cout=>Carry(3), s=>S(3));
fa4:fa port map(a=>A(4), b=>B(4), c=>carry(3), Cout=>Carry(4), s=>S(4));
fa5:fa port map(a=>A(5), b=>B(5), c=>carry(4), Cout=>Carry(5), s=>S(5));
fa6:fa port map(a=>A(6), b=>B(6), c=>carry(5), Cout=>Carry(6), s=>S(6));
```

```
fa7:fa port map(a=>A(7), b=>B(7), c=>carry(6), Cout=>Carry(7), s=>S(7));
fa8:fa port map(a=>A(8), b=>B(8), c=>carry(7), Cout=>Carry(8), s=>S(8));
fa9:fa port map(a=>A(9), b=>B(9), c=>carry(8), Cout=>Carry(9), s=>S(9));
fa10:fa port map(a=>A(10), b=>B(10), c=>carry(9), Cout=>Carry(10), s=>S(10));
fa11:fa port map(a=>A(11), b=>B(11), c=>carry(10), Cout=>Carry(11), s=>S(11));
fa12:fa port map(a=>A(12), b=>B(12), c=>carry(11), Cout=>Carry(12), s=>S(12));
fa13:fa port map(a=>A(13), b=>B(13), c=>carry(12), Cout=>Carry(13), s=>S(13));
fa14:fa port map(a=>A(14), b=>B(14), c=>carry(13), Cout=>Carry(14), s=>S(14));
fa15:fa port map(a=>A(15), b=>B(15), c=>carry(14), Cout=>Carry(15), s=>S(15));
fa16:fa port map(a=>A(16), b=>B(16), c=>carry(15), Cout=>Carry(16), s=>S(16));
fa17:fa port map(a=>A(17), b=>B(17), c=>carry(16), Cout=>Carry(17), s=>S(17));
fa18:fa port map(a=>A(18), b=>B(18), c=>carry(17), Cout=>Carry(18), s=>S(18));
fa19:fa port map(a=>A(19), b=>B(19), c=>carry(18), Cout=>Carry(19), s=>S(19));
fa20:fa port map(a=>A(20), b=>B(20), c=>carry(19), Cout=>Carry(20), s=>S(20));
fa21:fa port map(a=>A(21), b=>B(21), c=>carry(20), Cout=>Carry(21), s=>S(21));
fa22:fa port map(a=>A(22), b=>B(22), c=>carry(21), Cout=>Carry(22), s=>S(22));
fa23:fa port map(a=>A(23), b=>B(23), c=>carry(22), Cout=>Carry(23), s=>S(23));
fa24:fa port map(a=>A(24), b=>B(24), c=>carry(23), Cout=>Carry(24), s=>S(24));
fa25:fa port map(a=>A(25), b=>B(25), c=>carry(24), Cout=>Carry(25), s=>S(25));
fa26:fa port map(a=>A(26), b=>B(26), c=>carry(25), Cout=>Carry(26), s=>S(26));
fa27:fa port map(a=>A(27), b=>B(27), c=>carry(26), Cout=>Carry(27), s=>S(27));
fa28:fa port map(a=>A(28), b=>B(28), c=>carry(27), Cout=>Carry(28), s=>S(28));
fa29:fa port map(a=>A(29), b=>B(29), c=>carry(28), Cout=>Carry(29), s=>S(29));
fa30:fa port map(a=>A(30), b=>B(30), c=>carry(29), Cout=>Carry(30), s=>S(30));
fa31:fa port map(a=>A(31), b=>B(31), c=>carry(30), Cout=>Carryout, s=>S(31));
end Structural;
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;                    -- Using STD_LOGIC_LIBRARY_1164 package
--Entity Declaration
entity fa is
port(a,b,c:in std_logic;                        -- a,b and c are used as inputs
     Cout:out std_logic;                        --  Cout used as output
     s s:out std_logic                          -- s used as output
        );
end fa;
architecture fa_architecture of fa is           -- architecture declaration
begin
s<=a xor b xor c;                               -- Sum output equation
Cout<=(a and b) or (c and a) or (c and b);      -- Cout output equation
end fa_architecture;
```

Test Bench:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;                    -- Using STD_LOGIC_LIBRARY_1164 package
USE ieee.std_logic_arith.ALL;                   -- Using STD_LOGIC_LIBRARY_arith package
```

```
--Entity Declaration
ENTITY thirtytwo_bit_rca_tb IS
END thirtytwo_bit_rca_tb;

ARCHITECTURE behavior OF thirtytwo_bit_rca_tb IS                    -- architecture declaration
 -- Component Declaration for the Unit Under Test (UUT)
 COMPONENT thirtytwo_bit_rca
 PORT( A : IN  std_logic_vector(31 downto 0);                       -- A is used as input port
          B : IN  std_logic_vector(31 downto 0);                    -- B is used as input port
          Cin : IN  std_logic;                                      -- Cin is used as input port
          S : OUT  std_logic_vector(31 downto 0);                   -- S is used as output port
          Carryout : OUT  std_logic                                 -- Carryout is used as input port
          );
   END COMPONENT;
   --Inputs
  signal A : std_logic_vector(31 downto 0) := (others => '0');
  signal B : std_logic_vector(31 downto 0) := (others => '0');
  signal Cin : std_logic := '0';

 --Outputs
  signal S : std_logic_vector(31 downto 0);
  signal Carryout : std_logic;


BEGIN

-- Instantiate the Unit Under Test (UUT)
  uut: thirtytwo_bit_rca PORT MAP (
        A => A,
        B => B,
        Cin => Cin,
        S => S,
        Carryout => Carryout
      );
stim_proc: process
  begin
           Cin <= '0';
           wait for 10 ns;
           A <= "00000000000000000000000000000000";
           wait for 10 ns;
           B <= "00000000000000000000000000000000";
           wait for 10 ns;
           Cin <= '1';
           wait for 10 ns;
           A <= "00000000000000000000000000000010";
           wait for 10 ns;
           B <= "00000000000000000000000000001000";
```

```
        wait for 10 ns;
        A <= "11000000000000000000000000000010";
        wait for 10 ns;
        B <= "11000000000000000000000000001000";
        wait for 10 ns;
        A <= x"0AF70000";
        wait for 10 ns;
        B <= x"23640000";
        wait for 10 ns;
        end process;
        END;
```

Design a 4-bit CLA based on the information given.

VHDL code:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;                          -- Using STD_LOGIC_LIBRARY_1164 package

-- entity declaration
ENTITY four_bit_cla IS
  PORT
    (a     : IN   STD_LOGIC_VECTOR(3 DOWNTO 0);       -- a is used as input port
     b     : IN   STD_LOGIC_VECTOR(3 DOWNTO 0);       -- b is used as input port
     carry_in  : IN   STD_LOGIC;                      -- carry_in is used as input port
     sum       : OUT  STD_LOGIC_VECTOR(3 DOWNTO 0);  -- sum is used as input port
     carry_out :  OUT  STD_LOGIC                      -- carry_out is used as output port
    );
END four_bit_cla ;

ARCHITECTURE behavioral OF four_bit_cla  IS                -- architecture declaration
--Signal declaration
SIGNAL   sum_new  :   STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL   g        :   STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL   p        :   STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL   carry    :   STD_LOGIC_VECTOR(3 DOWNTO 1);
BEGIN
  sum_new <= a XOR b;   -- sum_new signal equation
  g <= a AND b;         -- generate signal equation
  p <= a OR b;          -- propagate signal equation
  PROCESS (g,p,carry)
  BEGIN
  carry(1) <= g(0) OR (p(0) AND carry_in);
    inst: FOR i IN 1 TO 2 LOOP                     -- for loop
        carry(i+1) <= g(i) OR (p(i) AND carry(i));
        END LOOP;
  carry_out <= g(3) OR (p(3) AND carry(3));
  END PROCESS;
```

```
   sum(0) <= sum_new(0) XOR carry_in;
   sum(3 DOWNTO 1) <= sum_new(3 DOWNTO 1) XOR carry(3 DOWNTO 1);
END behavioral;
```

Test Bench:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;                    -- Using STD_LOGIC_LIBRARY_1164 package
--Entity Declaration
ENTITY four_bit_cla_tb IS
END four_bit_cla_tb;

ARCHITECTURE behavior OF four_bit_cla_tb IS
  -- Component Declaration for the Unit Under Test (UUT)
   COMPONENT four_bit_cla
   PORT(
       a : IN  std_logic_vector(3 downto 0);          -- a is used as input
       b : IN  std_logic_vector(3 downto 0);          -- b is used as input
       carry_in : IN  std_logic;                      -- carry_in is used as input
       sum : OUT  std_logic_vector(3 downto 0);       -- sum is used as output
       carry_out : OUT  std_logic                     -- carry_out is used as output
      );
   END COMPONENT;


  --Input signals
  signal a : std_logic_vector(3 downto 0) := (others => '0');
  signal b : std_logic_vector(3 downto 0) := (others => '0');
  signal carry_in : std_logic := '0';
 --Output signals
  signal sum : std_logic_vector(3 downto 0);
  signal carry_out : std_logic;
  BEGIN
          -- Instantiate the Unit Under Test (UUT)
  uut: four_bit_cla PORT MAP (
       a => a,
       b => b,
       carry_in => carry_in,
       sum => sum,
       carry_out => carry_out
      );
  -- Stimulus process
  stim_proc: process
  begin
  carry_in <= '0';
```

```
  a <= "0001";
          wait for 10 ns;
          b <= "0001";
          wait for 10 ns;
          carry_in <= '1';
          wait for 10 ns;
          a <= "1000";
          wait for 10 ns;
          b <= "1000";
          wait for 10 ns;
          a <= "0101";
          wait for 10 ns;
          b <= "0101";
          wait for 10 ns;
     end process;
    END;
```

Design a 16-bit CLA based on the information.

VHDL code:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;                          -- Using STD_LOGIC_LIBRARY_1164 package
ENTITY sixteen_bit_cla IS                              -- entity declaration
   PORT
      ( a     : IN   STD_LOGIC_VECTOR(15 DOWNTO 0);          -- a is used as input port
       b     : IN   STD_LOGIC_VECTOR(15 DOWNTO 0);          -- b is used as input port
       carry_in  : IN   STD_LOGIC;                            -- carry_in is used as input port
       sum      : OUT  STD_LOGIC_VECTOR(15 DOWNTO 0);     -- sum is used as output port
       carry_out :  OUT  STD_LOGIC                            -- carry_out is used as input port
      );
END sixteen_bit_cla ;
--Architecture declaration
ARCHITECTURE behavioral OF sixteen_bit_cla  IS
--Signal declaration
SIGNAL   sum_new  :   STD_LOGIC_VECTOR(15 DOWNTO 0);
SIGNAL   g             :   STD_LOGIC_VECTOR(15 DOWNTO 0);
SIGNAL   p             :   STD_LOGIC_VECTOR(15 DOWNTO 0);
SIGNAL   carry         :   STD_LOGIC_VECTOR(15 DOWNTO 1);

BEGIN
   sum_new <= a XOR b;       -- sum_new signal logic equation
   g <= a AND b;                -- generate signal logic equation
   p <= a OR b;                 -- propagate signal logic equation
   PROCESS (g,p,carry)
   BEGIN
   carry(1) <= g(0) OR (p(0) AND carry_in);
      inst: FOR i IN 1 TO 14 LOOP        -- for loop
```

```
        carry(i+1) <= g(i) OR (p(i) AND carry(i));
        END LOOP;
  carry_out <= g(15) OR (p(15) AND carry(15));
  END PROCESS;

  sum(0) <= sum_new(0) XOR carry_in;
  sum(15 DOWNTO 1) <= sum_new(15 DOWNTO 1) XOR carry(15 DOWNTO 1);
  END behavioral;
```

Test Bench:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;                    -- Use Library IEEE_STD_LOGIC_1164 package
ENTITY sixteen_bit_cla_tb IS                    -- entity declaration
END sixteen_bit_cla_tb;

ARCHITECTURE behavior OF sixteen_bit_cla_tb IS      -- architecture declaration
-- Component Declaration for the Unit Under Test (UUT)
  COMPONENT sixteen_bit_cla
  PORT(
      a : IN  std_logic_vector(15 downto 0);          -- a is used as input port
      b : IN  std_logic_vector(15 downto 0);          -- b is used as input port
     carry_in : IN  std_logic;                        -- carry_in is used as input port
      sum : OUT  std_logic_vector(15 downto 0);       -- sum is used as output port
      carry_out : OUT  std_logic                      -- carry_out is used as input port
     );
  END COMPONENT;


 --Inputs
 signal a : std_logic_vector(15 downto 0) := (others => '0');
 signal b : std_logic_vector(15 downto 0) := (others => '0');
 signal carry_in : std_logic := '0';
--Outputs
 signal sum : std_logic_vector(15 downto 0);
 signal carry_out : std_logic;
 BEGIN
 -- Instantiate the Unit Under Test (UUT)
 uut: sixteen_bit_cla  PORT MAP (
       a => a,
       b => b,
       carry_in => carry_in,
       sum => sum,
       carry_out => carry_out
      );
 -- Stimulus process
 stim_proc: process
 begin
```

```
carry_in <= '0';
a  <= "0000000000000000";
wait for 10 ns;
b  <= "0000000000000000";
wait for 10 ns;
a  <= "0000000000000001";
wait for 10 ns;
b  <= "0000000000000111";
wait for 10 ns;
a  <= "1100000000000110";
wait for 10 ns;
b  <= "1100000000001011";
wait for 10 ns;
end process;
END;
```

Design a 32-bit CLA based on the information given.

VHDL code:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;                    -- Use Library IEEE_STD_LOGIC_1164 package

ENTITY thirty_two_bit_cla IS                    -- entity declaration
   PORT
      (a     : IN   STD_LOGIC_VECTOR(31 DOWNTO 0);        -- a is used as input port
       b     : IN   STD_LOGIC_VECTOR(31 DOWNTO 0);        -- b is used as input port
       carry_in  : IN   STD_LOGIC;                        -- carry_in is used as input port
       sum       : OUT  STD_LOGIC_VECTOR(31 DOWNTO 0);    -- sum is used as output port
       carry_out :  OUT  STD_LOGIC                        -- carry_out is used as input port
      );
END thirty_two_bit_cla ;

--architecture declaration

ARCHITECTURE behavioral OF thirty_two_bit_cla  IS

--signal declaration

SIGNAL   sum_new   :   STD_LOGIC_VECTOR(31 DOWNTO 0);
SIGNAL   g         :   STD_LOGIC_VECTOR(31 DOWNTO 0);
SIGNAL   p         :   STD_LOGIC_VECTOR(31 DOWNTO 0);
SIGNAL   carry     :   STD_LOGIC_VECTOR(31 DOWNTO 1);

BEGIN
   sum_new <= a XOR b;                  -- sum_new signal logic equation
   g <= a AND b;                        -- generate signal logic equation
   p <= a OR b;                         -- propagate signal logic equation
   PROCESS (g,p,carry)
```

```
   BEGIN
   carry(1) <= g(0) OR (p(0) AND carry_in);
      inst: FOR i IN 1 TO 30 LOOP                    for loop
           carry(i+1) <= g(i) OR (p(i) AND carry(i));
           END LOOP;
   carry_out <= g(31) OR (p(31) AND carry(31));
   END PROCESS;

   sum(0) <= sum_new(0) XOR carry_in;
sum(31 DOWNTO 1) <= sum_new(31 DOWNTO 1) XOR carry(31 DOWNTO 1);
END behavioral;
```

Test Bench:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;                    -- Use Library IEEE_STD_LOGIC_1164 package
ENTITY thirthy_two_bit_cla_tb IS                        -- entity declaration
END thirtytwo _bit_cla_tb;

--architecture declaration

ARCHITECTURE behavior OF thirty_two_bit_cla_tb IS
   -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT thirty_two_bit_cla
   PORT(
       a : IN  std_logic_vector(31 downto 0);          -- a is used as input port
       b : IN  std_logic_vector(31 downto 0);          -- b is used as input port
       carry_in : IN  std_logic;                       -- carry_in is used as input port
       sum : OUT  std_logic_vector(31 downto 0);       -- sum is used as output port
       carry_out : OUT  std_logic                      -- carry_out is used as output port
      );
   END COMPONENT;
  --Inputs
  signal a : std_logic_vector(31 downto 0) := (others => '0');
  signal b : std_logic_vector(31 downto 0) := (others => '0');
  signal carry_in : std_logic := '0';
 --Outputs
  signal sum : std_logic_vector(31 downto 0);
  signal carry_out : std_logic;
  BEGIN
 -- Instantiate the Unit Under Test (UUT)
  uut: thirty_two_bit_cla PORT MAP (
        a => a,
        b => b,
        carry_in => carry_in,
        sum => sum,
        carry_out => carry_out
       );
```

```
-- Stimulus process
stim_proc: process
begin
carry_in <= '0';
 a <= "00000000000000000000000000000000";
 wait for 10 ns;
 b <= "00000000000000000000000000000000";
 wait for 10 ns;
 carry_in <= '1';
 wait for 10 ns;
  a <= "00000000000000000000000000000010";
  wait for 10 ns;
  b <= "00000000000000000000000000001000";
  wait for 10 ns;
  a <= "11000000000000000000000000000010";
  wait for 10 ns;
  b <= "11000000000000000000000000001000";
  wait for 10 ns;
 A <= x"0AF70000";
 wait for 10 ns;
 B <= x"23640000";
 wait for 10 ns;
 end process;
 END;
```