Project 2
ECE 485: Computer Organization and Design

Alan Palayil and Zepei Cen
Professor: Won-Jae Yi

Acknowledgement: 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.

Electronic Signature: ***Alan Palayil A20447935*** (10/28/2022)

Zepei Cen A20437716 (10/28/2022)

Contents

Abstract:

In this project, we were asked to create a behavioral model of 1-bit Ripple Carry Adder (RCA) which meant simply mapping out what inputs of A, B, and Carry-In would result in what Sum and Carry-Out. It was basically a truth table in essence where the design of how an RCA was created was not considered. Only the inputs and outputs were considered, so we only cared about the functionality. Then, we were asked to create a 4-bit RCA structural model in which I implemented the 1-bit RCA as a core-component and connected four of them to create a 4-bit RCA with input values and outputs. We implement the 4-bit RCA component in a 32-bit RCA using eight 4-bit RCAs in them.

We also work on Carry Lookahead Adder (CLA) to design a 16-bit and 32-bit CLA. All the model designs are run through a test bench with sample data inputs to check the correct outputs.

Introduction:

The purpose of this Project 2 is to prepare you towards Project 3, in which we will design and implement our own custom 32-bit RISC processor, a stripped-down MIPS processor. The goal of this project is to design a 1-bit, 4-bit, and 32-bit RCA along with a 4-bit, 16-bit, and 32-bit CLA which are the basic functionalities of an Arithmetic Logic Unit (ALU).

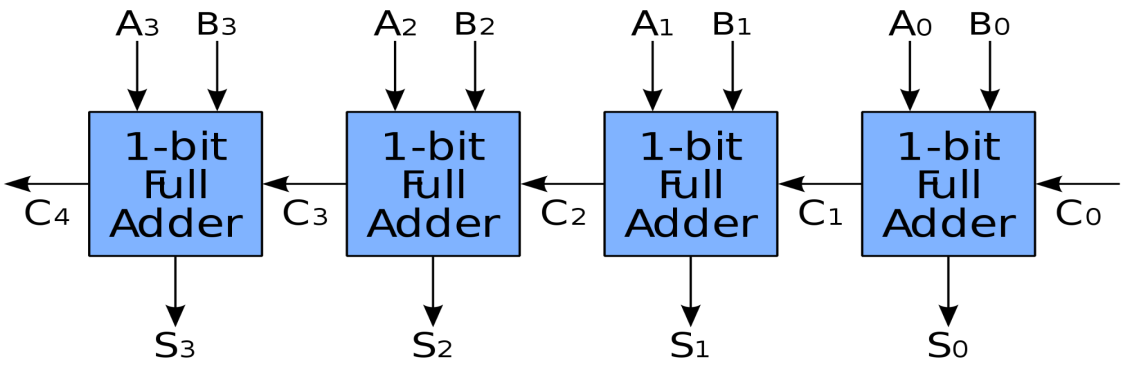
In this project, you will design and implement the required RCA and CLA in behavioral model and structural model using VHDL. After designing the models, we test them using our own testbench.

Background:

As discussed in class, the 1-bit Full Adder which consists of 3 inputs (A, B, and Cin) and two outputs (S and Cout).

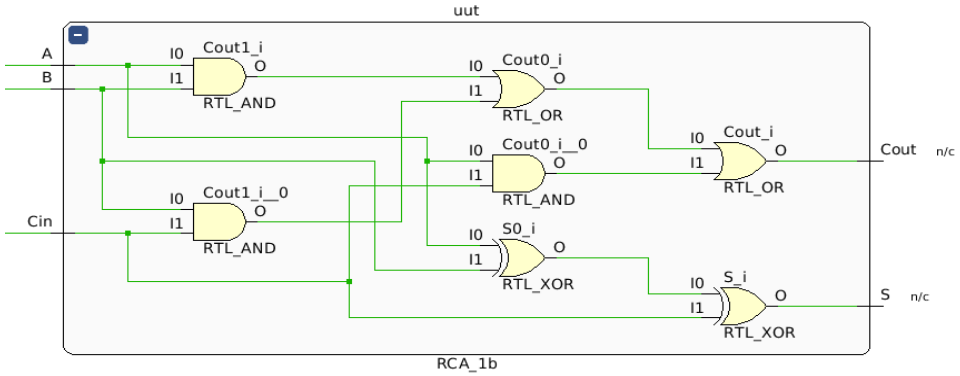
i. Description of a Full-Adder/1-bit RCA using behavioral model

A Full Adder is a digital circuit that performs additions. It is implemented using logic gates in hardware. It can add three 1-bit binary numbers (A, B, and Carry in), two operands, and a carry bit. The Adder outputs two numbers , a sum (S) and Carry out (Cout). We build the 1-bit RCA using the RCA logic. We use the 1-bit RCA behavioral to build the 4-bit RCA structural model which in turn was used to build the 32-bit RCA.

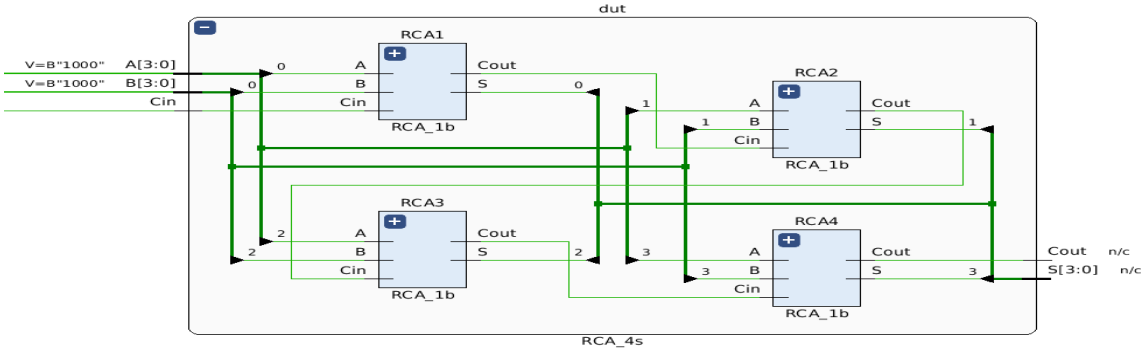


ii. Description of your VHDL simulator and environment

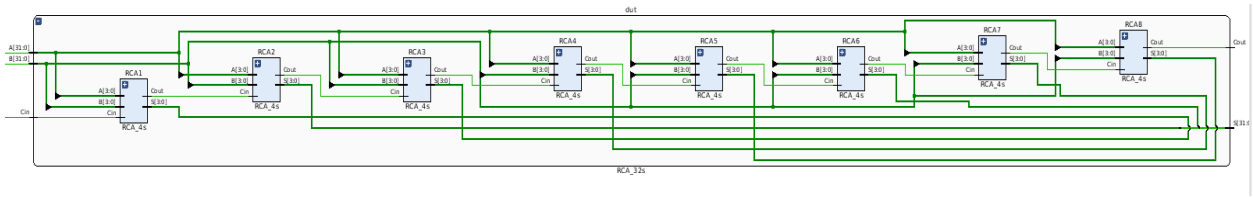
Vivado and ModelSim are environments in which we can code in various HDL languages such as Verilog and VHDL to create circuit implementations that can be compiled, and the files can be checked to see if there are any errors to fix. We can design behavioral and structural models to replicate or create various components. We can also create simulated schematics and use the test bench files to simulate the output using waveform generators for the requested time period.



Schematic of 1-bit RCA behavioral model made by Vivado



Schematic of 4-bit RCA structural model made by Vivado



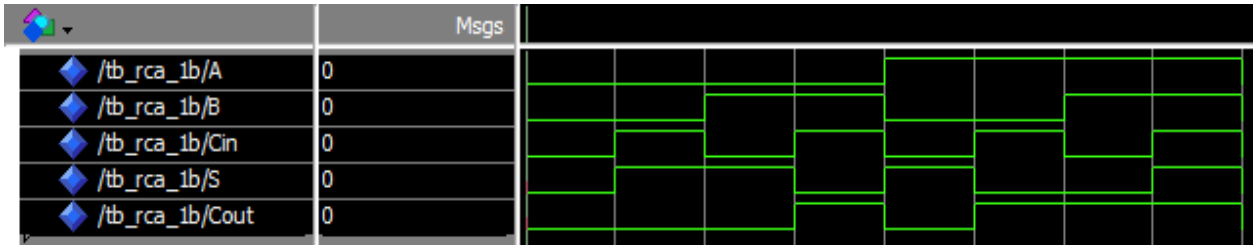
Schematic of 32-bit RCA structural model made by Vivado

i. Description of your VHDL testbench code

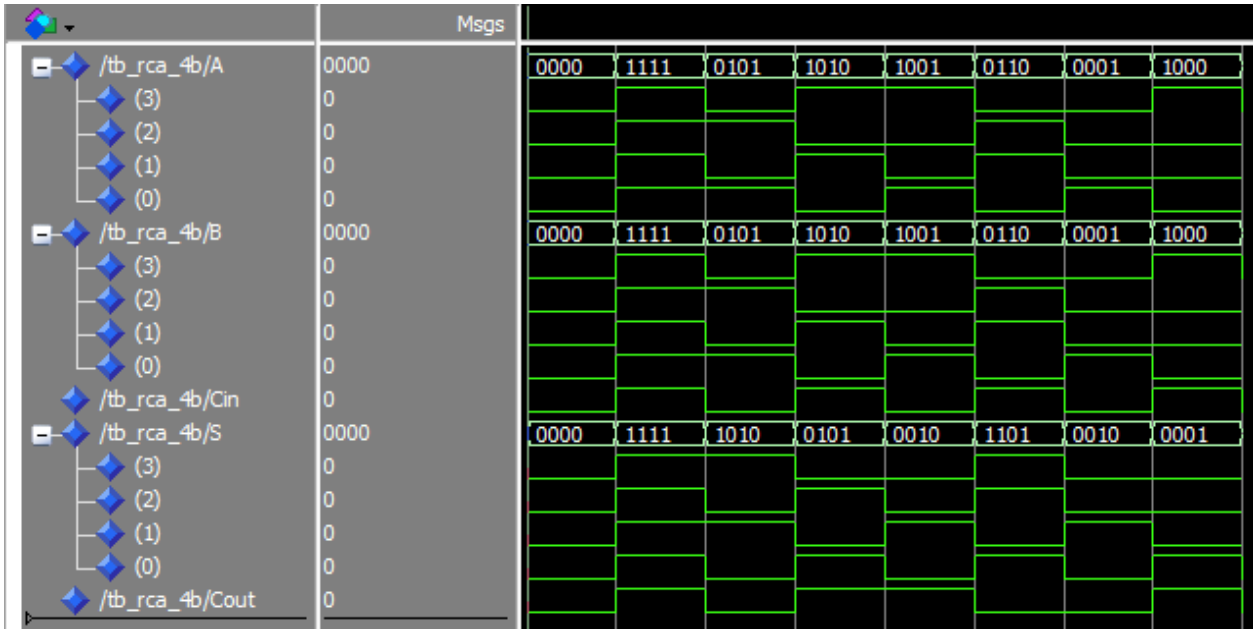
The testbench files are inputted with the behavioral and structural models. I initialized all the signals as '0'. Then I created the port map in which I mapped each signal input to the data inputs for the models. I then created the sample data inputs then went through each selection for 50 ns every time. The purpose of this was to have the waveform show outputs clearly.

Simulation Results and Discussion:

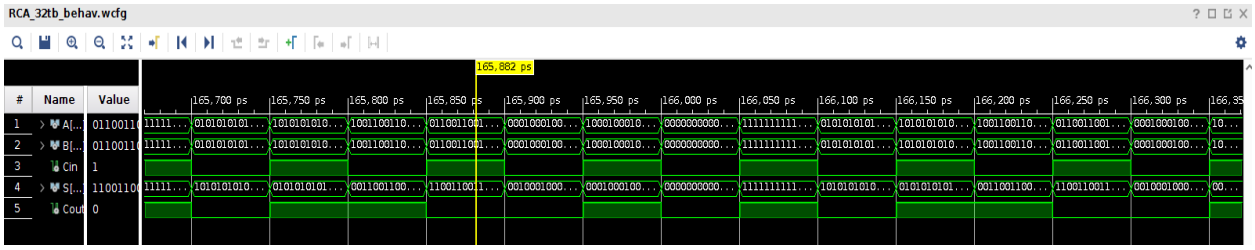
i. Screenshots of your test cases



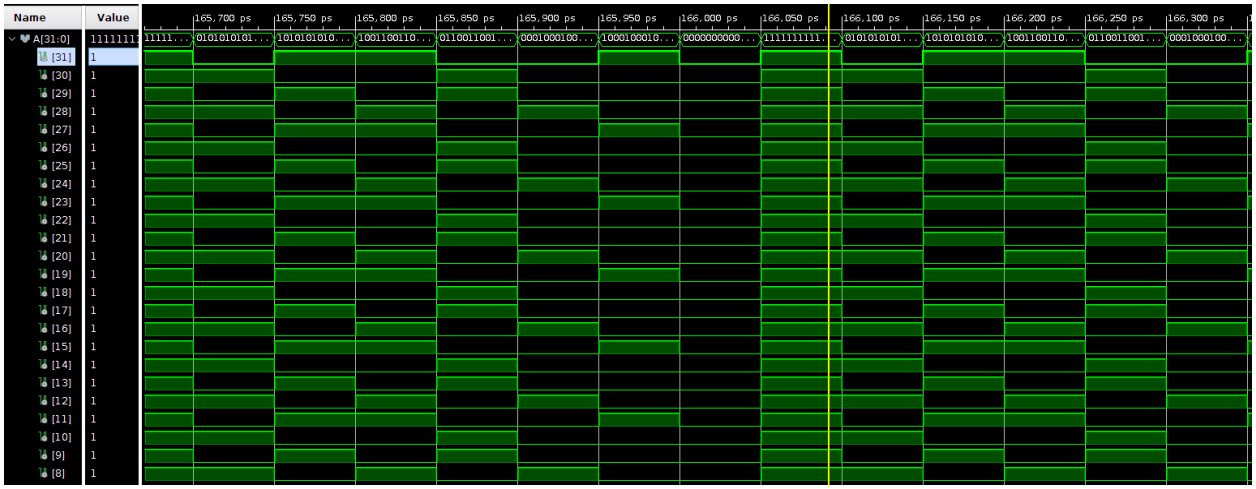
1-bit RCA (Figure 1.)



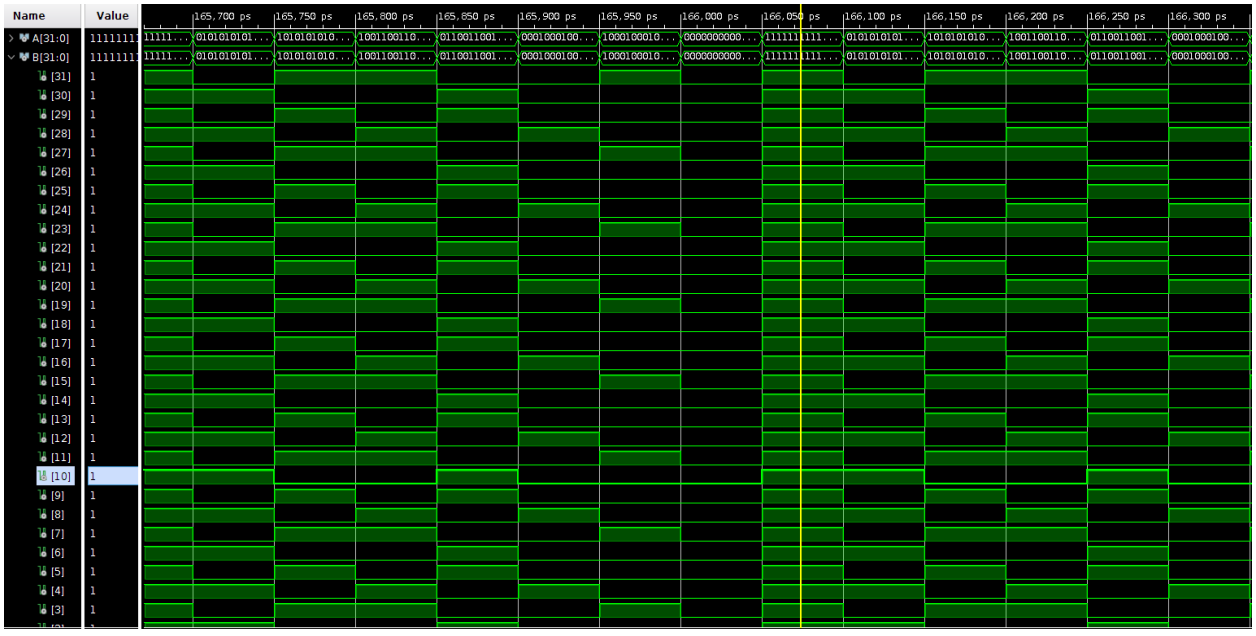
4-bit RCA (Figure 2.)



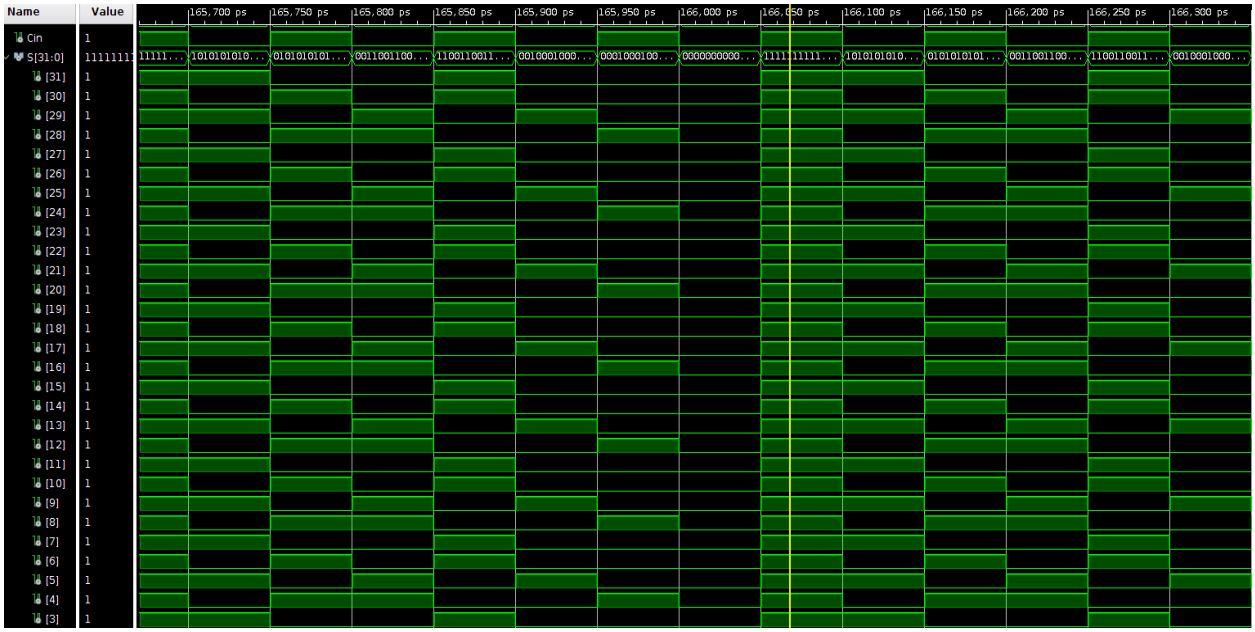
32-bit RCA A[31:0] expanded:



32-bit RCA B[31:0] expanded:



32-bit RCA S[31:0] expanded:



32-bit RCA (Figure 3.)

ii. Descriptions of each screenshot

- The 1-bit RCA (Figure 1) done in ModelSim, We chose data inputs '0' and '1' in iteration in A, B, abd Cin. We can see from Cout and S how they fluctuate between 0 and 1.
- The 4-bit RCA (Figure 2) done in ModelSim, We chose random data input. I just alternated between 0 and 1 for the inputs while going over the time, the signals fluctuated between 0 and 1.
- The 32-bit RCA (Figure 3) was done in Vivado, similarly to 4-bit RCA, we chose random data input. I just alternated between 0 and 1 for the inputs while going over the time, the signals fluctuated between 0 and 1.

iii. Discussion of any issues faced and improvements that could be made

- While designing the files in Vivado, Alan started getting breakpoints errors in all the files which caused him a setback. The error was opt_design which creates random breakpoints in his process in the architecture. He continued running the simulations on ModelSim for running the 1-bit and 4-bit RCA while trying the files on Vivado simultaneously. For unknown reasons, 32-bit RCA compiles in Vivado with all the existing files.

Conclusion:

This project gave us an insight on how we can set up a Ripple Carry Adder (RCA) using a full adder (FA) to see how the behavioral and structural models of 1-bit, 4-bit, and 32-bit RCA run

under the waveform generator. We also worked on the Carry Lookahead Adder (CLA) using Full Adder to design and create 4-bit, 16-bit, and 32-bit CLA. While working on this , we learned that the Carry Lookahead Adder is just an improved version of an Ripple Carry Adder as it creates the carry-in input value for all the adders simultaneously since the carry in bits are already calculated before being entered into the Adder meaning that the result of the from a Carry Lookahead Adder can be calculated almost instantly, whilst the Ripple Carry Adder waits for the carry out of the previous adder to be used to set the value of the carry in meaning that the run time is dependent on the amount of adders the more adders the longer it takes to complete the operations.

Distribution of Work:

Alan Palayil	Designed 32 bit RSA, 1 bit, 4 bit , 16 bit CLA
Zepei Cen	Designed 1 bit , 4 bit RSA , 32 bit CLA

List of References:

- VHDL Tutorial by van der Spiegel, VHDL Tutorial by Ashenden, ModelSim Tutorial, and Vivado Tutorial
    - I was taught how to create port maps.This is crucial for receiving the appropriate inputs to achieve the desired outputs.
- https://allaboutfpga.com/4-bit-ripple-carry-adder-vhdl-code/
    - Taught me how to use array indices in port maps references. This was significant since it made the whole code simpler to use arrays.
- https://www.ics.uci.edu/~jmoorkan/vhdlref/vhdl.html
    - Provided me with numerous illustrations of how to set up signals and port maps. This was crucial so I could ascertain just what my code required.
- https://www.ics.uci.edu/~jmoorkan/vhdlref/arrays.html
    - I learned many methods for inserting values into an array from doing this. This was crucial for setting up my arrays.
- https://stackoverflow.com/questions/40723529/assign-values-to-an-array-partially-in-vhdl
    - Taught me how to choose a certain set of indices from an array in part. It was crucial to remember to only select the necessary indices when creating port maps.
- https://stackoverflow.com/questions/42544675/can-i-access-2-indices-of-an-array-at-the-same-time-vhdl/42549118
    - Taught me how to choose a certain set of indices from an array in part. It was crucial to remember to only select the necessary indices when creating port maps.

- https://www.nandland.com/vhdl/examples/example-array-type-vhdl.html
    - Taught me how to create signal arrays from scratch. This was crucial for setting up the test benches' signals.

Appendix:

i. Entire Source Code of project with comments

1-bit RCA Behavioral Model:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity RCA_1b is
    port(
    A,B,Cin: in std_logic; --Data Input
    Cout,S: out std_logic --Data Output
    );
end RCA_1b;

architecture Behavioral of RCA_1b is

begin
    S<=A XOR B XOR Cin;
    Cout<=(A AND B) OR (B AND Cin) or (A AND Cin);

end Behavioral;
```

4-bit RCA Structural Model:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity RCA_4s 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 RCA_4s;

architecture Structural of RCA_4s is
component RCA_1b
        port(
A,B,Cin  : in  std_logic;
               S,Cout
  : out std_logic
          );
    end component;

    --Set signals
    signal C : std_logic_vector (3 downto 1);
begin
    --Utilize 1-bit RCA to create 4-bit RCA
    RCA1:
RCA_1b port map(A(0), B(0), Cin,  S(0), C(1));
    RCA2:
RCA_1b port map(A(1), B(1), C(1), S(1), C(2));
    RCA3:
RCA_1b port map(A(2), B(2), C(2), S(2), C(3));
    RCA4:
RCA_1b port map(A(3), B(3), C(3), S(3), Cout);
end Structural;
```

32-bit RCA Structural Model:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity RCA_32s is
    port(   A
  : in  std_logic_vector (31 downto 0);
           B
  : in  std_logic_vector (31 downto 0);
           Cin
: in
std_logic;
           S
  : out std_logic_vector (31 downto 0);
           Cout
: out std_logic
        );
end RCA_32s;

architecture Structural of RCA_32s is
```

```
component RCA_4s
        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 component;

    --Signals
    signal  C : std_logic_vector (7 downto 1);

begin
    --utilize eight 4-bit RCAs
    RCA1:
RCA_4s port map(A(3  downto 0),   B(3 downto 0),  Cin,
S(3
downto 0),  C(1));
    RCA2:
RCA_4s port map(A(7  downto 4),   B(7 downto 4),  C(1), S(7  downto 4),  C(2));
    RCA3:
RCA_4s port map(A(11 downto 8),  B(11 downto 8),  C(2), S(11 downto 8),  C(3));
    RCA4:
RCA_4s port map(A(15 downto 12), B(15 downto 12), C(3), S(15 downto 12), C(4));
    RCA5:
RCA_4s port map(A(19 downto 16), B(19 downto 16), C(4), S(19 downto 16), C(5));
    RCA6:
RCA_4s port map(A(23 downto 20), B(23 downto 20), C(5), S(23 downto 20), C(6));
    RCA7:
RCA_4s port map(A(27 downto 24), B(27 downto 24), C(6), S(27 downto 24), C(7));
    RCA8:
RCA_4s port map(A(31 downto 28), B(31 downto 28), C(7), S(31 downto 28), Cout);
end Structural;
```

4-bit CLA Structural Model:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity CLA_4s 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 CLA_4s;

architecture Structural of CLA_4s is

function Carry(idx : integer; A : std_logic_vector; B : std_logic_vector; Cin :
std_logic) return std_logic is
    begin
        if (idx = 0) then
            return Cin;
        else
            return (A(idx-1) and B(idx-1)) or ((A(idx-1) or B(idx-1)) and
Carry(idx-1, A, B, Cin));
        end if;
    end function;
begin
    process(A,B,Cin) is
    begin
        for i in 0 to 3 loop
            S(i)
<= (A(i) xor B(i)) xor Carry(i, A, B, Cin);
        end loop;
        Cout
<= Carry(4, A, B, Cin);
    end process;

end Structural;
```

16-bit CLA Structural Model:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity CLA_16s is
    port(   A
  : in  std_logic_vector (15 downto 0);
```

```
            B
  : in  std_logic_vector (15 downto 0);
            Cin
: in
std_logic;
            S
  : out std_logic_vector (15 downto 0);
            Cout
: out std_logic
        );
end CLA_16s;

architecture Structural of CLA_16s is
function Carry(idx : integer; A : std_logic_vector; B : std_logic_vector; Cin :
std_logic) return std_logic is
    begin
        if (idx = 0) then
            return Cin;
        else
            return (A(idx-1) and B(idx-1)) or ((A(idx-1) or B(idx-1)) and
Carry(idx-1, A, B, Cin));
        end if;
    end function;
begin
    process(A,B,Cin) is
    begin
        for i in 0 to 15 loop
            S(i)
<= (A(i) xor B(i)) xor Carry(i, A, B, Cin);
        end loop;
        Cout
<= Carry(16, A, B, Cin);
    end process;
end Structural;
```

32-bit CLA Structural Model:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity CLA_32s is
    port(   A
  : in  std_logic_vector (31 downto 0);
            B
  : in  std_logic_vector (31 downto 0);
            Cin
```

```
: in
std_logic;
           S
  : out std_logic_vector (31 downto 0);
           Cout
: out std_logic
       );
end CLA_32s;

architecture Structural of CLA_32s is
function Carry(idx : integer; A : std_logic_vector; B : std_logic_vector; Cin :
std_logic) return std_logic is
    begin
        if (idx = 0) then
            return Cin;
        else
            return (A(idx-1) and B(idx-1)) or ((A(idx-1) or B(idx-1)) and
Carry(idx-1, A, B, Cin));
        end if;
    end function;
begin
    process(A,B,Cin) is
    begin
        for i in 0 to 31 loop
            S(i)
<= (A(i) xor B(i)) xor Carry(i, A, B, Cin);
        end loop;
        Cout
<= Carry(32, A, B, Cin);
    end process;
end Structural;
```

ii. Entire Testbench Code with comments used to verify design

Testbench Code of 1-bit RCA using Behavioral Model:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity RCA_1tb is
end RCA_1tb;

architecture TestBench of RCA_1tb is
component RCA_1b
 port(
    A,B,Cin: in std_logic; --Data Input
    Cout,S: out std_logic --Data Output
```

```
    );
end component;
--Input signals
signal A:std_logic :='0';
signal B:std_logic :='0';
signal Cin:std_logic :='0';
--Output signals
signal Cout:std_logic :='0';
signal S:std_logic :='0';

begin
uut: RCA_1b port map(
A=>A,
B=>B,
Cin=>Cin,
Cout=>Cout,
S=>S
);

Test: process

begin
        A <= '0'; B <= '0'; Cin <= '0'; wait for 50 ps;
        A <= '0'; B <= '0'; Cin <= '1'; wait for 50 ps;
        A <= '0'; B <= '1'; Cin <= '0'; wait for 50 ps;
        A <= '0'; B <= '1'; Cin <= '1'; wait for 50 ps;
        A <= '1'; B <= '0'; Cin <= '0'; wait for 50 ps;
        A <= '1'; B <= '0'; Cin <= '1'; wait for 50 ps;
        A <= '1'; B <= '1'; Cin <= '0'; wait for 50 ps;
        A <= '1'; B <= '1'; Cin <= '1'; wait for 50 ps;
end process;

end TestBench;
```

Testbench Code of 4-bit RCA Structural Model:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity RCA_4tb is
end RCA_4tb;

architecture TestBench of RCA_4tb is
component RCA_4s
        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 component;

    --Signals
    signal A    : std_logic_vector (3 downto 0) := "0000";
    signal B    : std_logic_vector (3 downto 0) := "0000";
    signal Cin  : std_logic
                := '0';
    signal S    : std_logic_vector (3 downto 0) := "0000";
    signal Cout : std_logic
                := '0';

begin
    dut:
RCA_4s
        port map(
            A
  => A,
            B
  => B,
            Cin
=> Cin,
            S
  => S,
            Cout
=> Cout
        );

    test
: process
    begin
        --Tests
        A
<= "0000"; B <= "0000"; Cin <= '0'; wait for 50 ps;
        A
<= "1111"; B <= "1111"; Cin <= '1'; wait for 50 ps;
```

```
        A
<= "0101"; B <= "0101"; Cin <= '0'; wait for 50 ps;
        A
<= "1010"; B <= "1010"; Cin <= '1'; wait for 50 ps;
        A
<= "1001"; B <= "1001"; Cin <= '0'; wait for 50 ps;
        A
<= "0110"; B <= "0110"; Cin <= '1'; wait for 50 ps;
        A
<= "0001"; B <= "0001"; Cin <= '0'; wait for 50 ps;
        A
<= "1000"; B <= "1000"; Cin <= '1'; wait for 50 ps;
    end process;
end TestBench;
```

Testbench Code of 32-bit RCA Structural Model:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity RCA_32tb is
end RCA_32tb;

architecture TestBench of RCA_32tb is
component RCA_32s
        port(   A
  : in  std_logic_vector (31 downto 0);
                B
  : in  std_logic_vector (31 downto 0);
                Cin
: in
std_logic;
                S
  : out std_logic_vector (31 downto 0);
                Cout
: out std_logic
            );
    end component;

    --Signals
    signal A    : std_logic_vector (31 downto 0) :=
"00000000000000000000000000000000";
    signal B    : std_logic_vector (31 downto 0) :=
"00000000000000000000000000000000";
    signal Cin  : std_logic
                  := '0';
```

```
    signal S    : std_logic_vector (31 downto 0) :=
"00000000000000000000000000000000";
    signal Cout : std_logic
                  := '0';

begin
    dut:
RCA_32s
        port map(
            A
  => A,
            B
  => B,
            Cin
=> Cin,
            S
  => S,
            Cout
=> Cout
        );

    test
: process
    begin
        --Tests
        A
<= "00000000000000000000000000000000"; B <= "00000000000000000000000000000000";
Cin <= '0'; wait for 50 ps;
        A
<= "11111111111111111111111111111111"; B <= "11111111111111111111111111111111";
Cin <= '1'; wait for 50 ps;
        A
<= "01010101010101010101010101010101"; B <= "01010101010101010101010101010101";
Cin <= '0'; wait for 50 ps;
        A
<= "10101010101010101010101010101010"; B <= "10101010101010101010101010101010";
Cin <= '1'; wait for 50 ps;
        A
<= "10011001100110011001100110011001"; B <= "10011001100110011001100110011001";
Cin <= '0'; wait for 50 ps;
        A
<= "01100110011001100110011001100110"; B <= "01100110011001100110011001100110";
Cin <= '1'; wait for 50 ps;
        A
```

```
<= "0001000100010001000100010001000100010001"; B <= "0001000100010001000100010001000100010001";
Cin <= '0'; wait for 50 ps;
        A
<= "1000100010001000100010001000100010001000"; B <= "1000100010001000100010001000100010001000";
Cin <= '1'; wait for 50 ps;
    end process;
end TestBench;
```

Testbench Code of 4-bit CLA Structural Model:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity CLA_4tb is
end CLA_4tb;

architecture TestBench of CLA_4tb is
component CLA_4s
        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 component;

    signal A    : std_logic_vector (3 downto 0) := "0000";
    signal B    : std_logic_vector (3 downto 0) := "0000";
    signal Cin  : std_logic
                := '0';
    signal S    : std_logic_vector (3 downto 0) := "0000";
    signal Cout : std_logic
                := '0';

begin
    dut:
CLA_4s
        port map(
            A
  => A,
```

```
            B
  => B,
            Cin
=> Cin,
            S
  => S,
            Cout
=> Cout
        );

    test
: process
    begin
        A
<= "0000"; B <= "0000"; Cin <= '0'; wait for 50 ps;
        A
<= "1111"; B <= "1111"; Cin <= '1'; wait for 50 ps;
        A
<= "0101"; B <= "0101"; Cin <= '0'; wait for 50 ps;
        A
<= "1010"; B <= "1010"; Cin <= '1'; wait for 50 ps;
        A
<= "1001"; B <= "1001"; Cin <= '0'; wait for 50 ps;
        A
<= "0110"; B <= "0110"; Cin <= '1'; wait for 50 ps;
        A
<= "0001"; B <= "0001"; Cin <= '0'; wait for 50 ps;
        A
<= "1000"; B <= "1000"; Cin <= '1'; wait for 50 ps;
    end process;

end TestBench;
```

Testbench Code of 16-bit CLA Structural Model:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity CLA_16tb is
end CLA_16tb;

architecture TestBench of CLA_16tb is
component CLA_16s
        port(   A
  : in  std_logic_vector (15 downto 0);
                B
```

```
  : in  std_logic_vector (15 downto 0);
               Cin
: in  std_logic;
               S
  : out std_logic_vector (15 downto 0);
               Cout
: out std_logic
           );
    end component;

    signal A    : std_logic_vector (15 downto
0) := "0000000000000000";
    signal B    : std_logic_vector (15 downto
0) := "0000000000000000";
    signal Cin  : std_logic
                  := '0';
    signal S    : std_logic_vector (15 downto
0) := "0000000000000000";
    signal Cout : std_logic
                  := '0';

begin
    dut:
CLA_16s
        port map(
            A
  => A,
            B
  => B,
            Cin
=> Cin,
            S
  => S,
            Cout
=> Cout
        );

    test
: process
    begin
        A
<= "0000000000000000"; B <= "0000000000000000"; Cin <= '0'; wait for 50 ps;
        A
<= "1111111111111111"; B <= "1111111111111111"; Cin <= '1'; wait for 50 ps;
        A
```

```
<= "0101010101010101"; B <= "0101010101010101"; Cin <= '0'; wait for 50 ps;
        A
<= "1010101010101010"; B <= "1010101010101010"; Cin <= '1'; wait for 50 ps;
        A
<= "1001100110011001"; B <= "1001100110011001"; Cin <= '0'; wait for 50 ps;
        A
<= "0110011001100110"; B <= "0110011001100110"; Cin <= '1'; wait for 50 ps;
        A
<= "0001000100010001"; B <= "0001000100010001"; Cin <= '0'; wait for 50 ps;
        A
<= "1000100010001000"; B <= "1000100010001000"; Cin <= '1'; wait for 50 ps;
    end process;
end TestBench;
```

Testbench Code of 32-bit CLA Structural Model:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity CLA_32tb is
end CLA_32tb;

architecture TestBench of CLA_32tb is
component CLA_32s
        port(   A
  : in  std_logic_vector (31 downto 0);
                B
  : in  std_logic_vector (31 downto 0);
                Cin
: in
std_logic;
                S
  : out std_logic_vector (31 downto 0);
                Cout
: out std_logic
            );
    end component;

    signal A    : std_logic_vector (31 downto 0) :=
"00000000000000000000000000000000";
    signal B    : std_logic_vector (31 downto 0) :=
"00000000000000000000000000000000";
    signal Cin  : std_logic
                  := '0';
    signal S    : std_logic_vector (31 downto 0) :=
"00000000000000000000000000000000";
```

```
    signal Cout : std_logic
                  := '0';

begin
    dut:
CLA_32s
        port map(
            A
  => A,
            B
  => B,
            Cin
=> Cin,
            S
  => S,
            Cout
=> Cout
        );

    test
: process
    begin
        A
<= "00000000000000000000000000000000"; B <= "00000000000000000000000000000000";
Cin <= '0'; wait for 50 ps;
        A
<= "11111111111111111111111111111111"; B <= "11111111111111111111111111111111";
Cin <= '1'; wait for 50 ps;
        A
<= "01010101010101010101010101010101"; B <= "01010101010101010101010101010101";
Cin <= '0'; wait for 50 ps;
        A
<= "10101010101010101010101010101010"; B <= "10101010101010101010101010101010";
Cin <= '1'; wait for 50 ps;
        A
<= "10011001100110011001100110011001"; B <= "10011001100110011001100110011001";
Cin <= '0'; wait for 50 ps;
        A
<= "01100110011001100110011001100110"; B <= "01100110011001100110011001100110";
Cin <= '1'; wait for 50 ps;
        A
<= "00010001000100010001000100010001"; B <= "00010001000100010001000100010001";
Cin <= '0'; wait for 50 ps;
        A
```

```
<= "10001000100010001000100010001000"; B <= "10001000100010001000100010001000";
Cin <= '1'; wait for 50 ps;
    end process;

end TestBench;
```