University of Colorado Boulder
ECEE Department

ECEN 2350 - Digital Logic - Fall 2023
Location: Engineering Center, ECCR 1B40, MWF 1:25PM - 2:15PM
Instructor: Dr. Mona ElHelbawy
Lab #2
Lab Title: Design a 4-bit RCA using Vivado SystemVerilog
Date of Experiment: September 25th, 2023
Names: Connor Sorrell

Part 1

Description:

- In this lab, we will experiment and work with three different modules. These will be a 2-bit half adder, a 3-bit full adder, and a ripple carry adder (RCA).

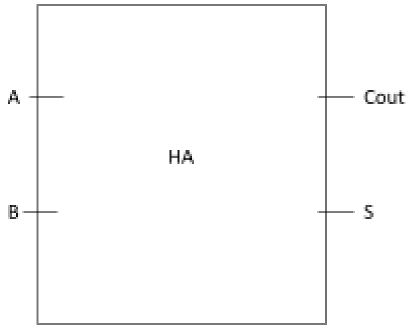


Figure 1.1.0: Black Box Diagram showing the 2 inputs (A,B) and the output S, and carryout, C. This circuit adds two bits.

Truth Table:

- This table shows the boolean logic and function values for each combination of inputs. As seen, the half adder adds the two inputs. If the sum is 1, the output, S will be 1. If the sum is 0, the output, S is 0. If the sum is 2, the output, S, will be 1, and the carry out will be 1.

index	A	B	S	Cout
0	0	0	0	0
1	0	1	1	0
2	1	0	1	0
3	1	1	1	1

Figure 1.1.1: Truth table representing the boolean logic present in the half adder. (This will add two binary bits)

	B	0	1
A			
0		0	1
1		1	0

Boolean Expression: S = A ^ B

Figure 1.1.2: K-map for S

<u>K-map for C:</u>

	B	0	1
A			
0		0	0
1		0	1

Boolean Expression: C = A * B

Figure 1.1.3: K-map for C

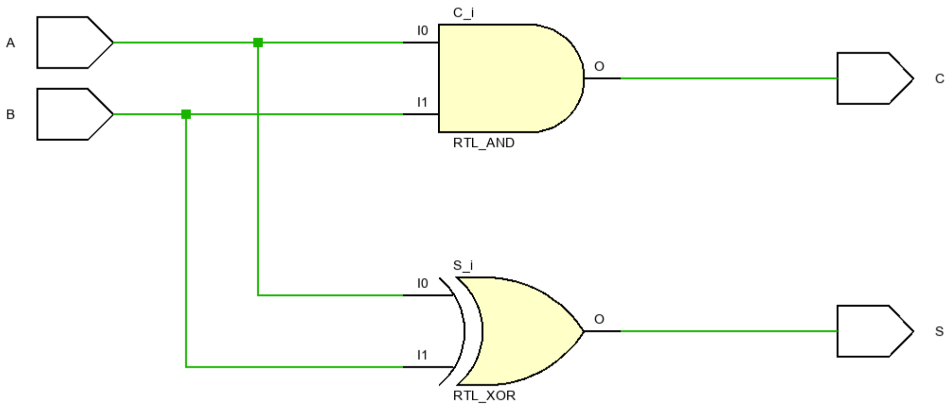
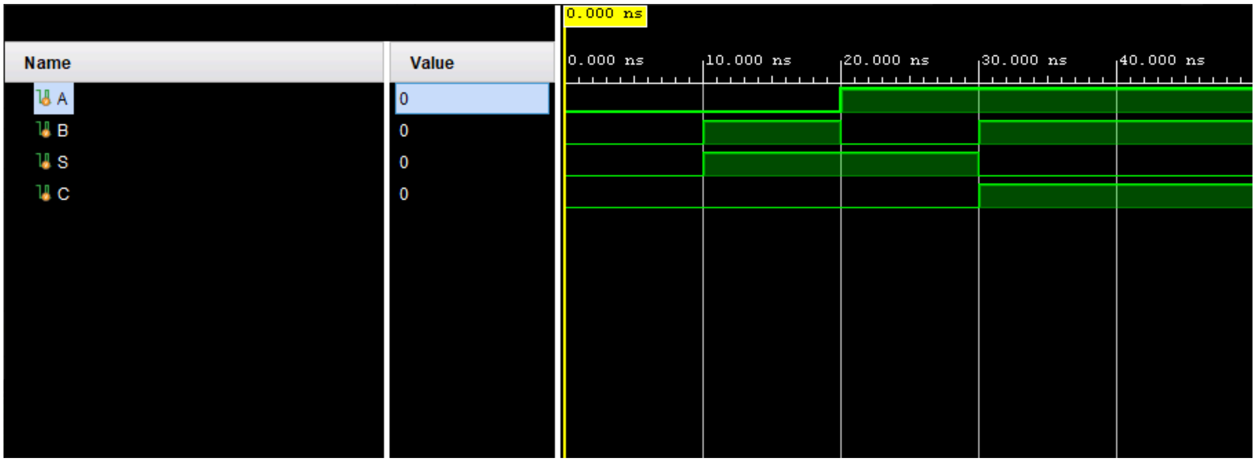
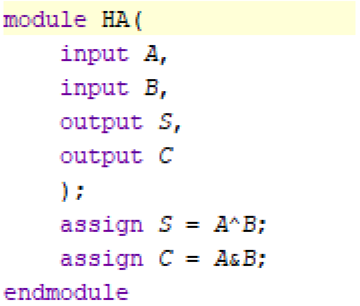


Figure 1.1.4: Schematic of the Half Adder circuit. Showcases two inputs, an XOR gate leading to an output, and an AND gate leading to the carryout output.



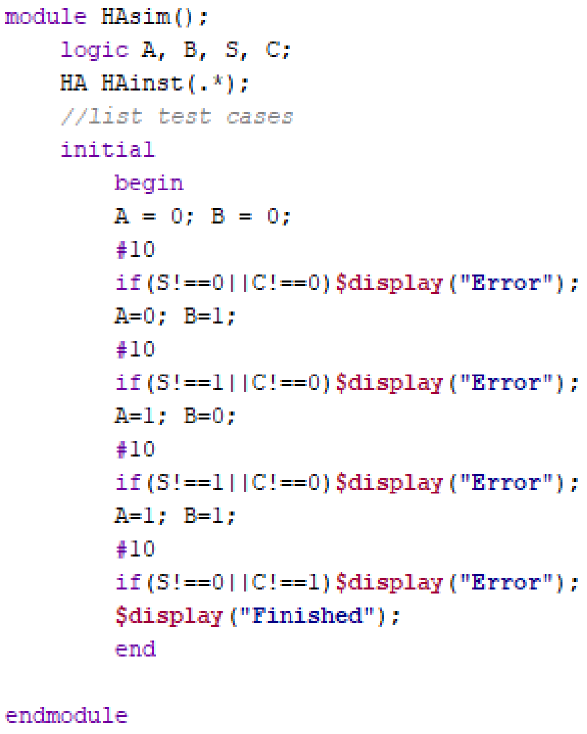
Name	Value
A	0
B	0
S	0
C	0

Figure 1.1.5: Timing diagram for the Half Adder. Showcases all possible inputs and their outputs.



```
module HA(
    input A,
    input B,
    output S,
    output C
    );
    assign S = A^B;
    assign C = A&B;
endmodule
```

Figure 1.1.6: Source code for the half adder circuit. Shows the 2 inputs, 2 outputs, and the boolean logic.



```
module HAsim();
    logic A, B, S, C;
    HA HAinst(.*);
    //list test cases
    initial
        begin
        A = 0; B = 0;
        #10
        if(S!==0||C!==0)$display("Error");
        A=0; B=1;
        #10
        if(S!==1||C!==0)$display("Error");
        A=1; B=0;
        #10
        if(S!==1||C!==0)$display("Error");
        A=1; B=1;
        #10
        if(S!==0||C!==1)$display("Error");
        $display("Finished");
        end

endmodule
```

Figure 1.1.7: Simulation code for the half adder circuit. Shows each test case.

Part 2

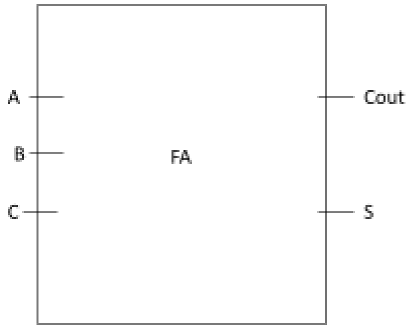


Figure 1.2.0: Black Box Diagram showing the 3 inputs (A,B,C) and output S, and carryout, C. This circuit will add three binary bits.

Truth Table:

- This table shows the boolean logic and function values for each combination of inputs. As seen, the full adder adds the three inputs. If the sum is 1, the output, S will be 1. If the sum is 0, the output, S is 0. If the sum is 2, the output, S, will be 1, and the carry out will be 1.

index	A	B	S	Cout
0	0	0	0	0
1	0	1	1	0
2	1	0	1	0
3	1	1	1	1

Figure 1.2.1: Truth table representing the boolean logic present in the full adder. (This will add three binary bits)

	B/C	00	01	11	10
A					
0		0	1	0	1
1		1	0	1	0

Boolean Expression: S = A ^ B ^ C

Figure 1.2.2: K-map for S

	B/C	00	01	11	10
A					
0		0	0	1	0
1		0	1	1	1

Boolean Expression: C = BC + AB + AC

Figure 1.2.3: K-map for C

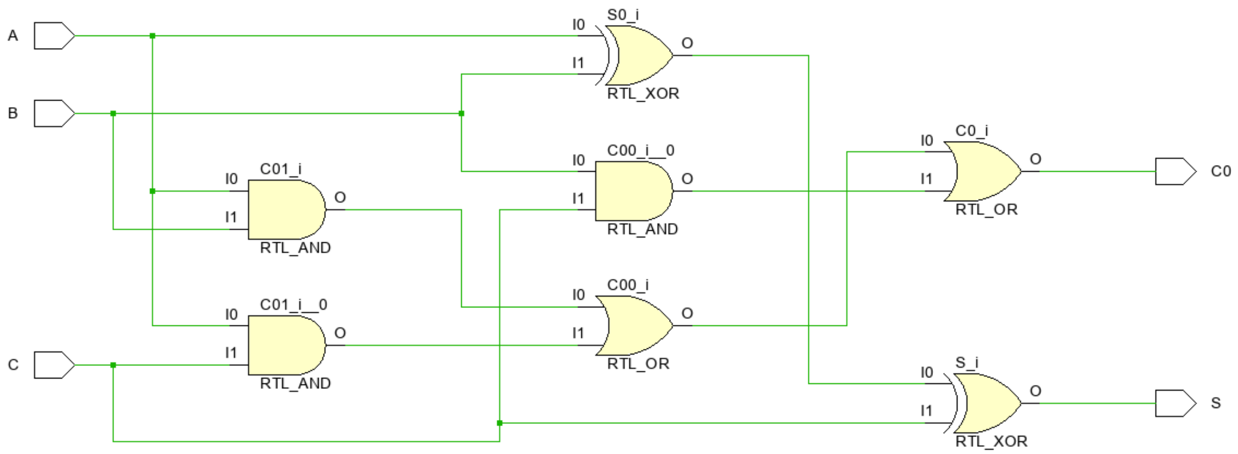
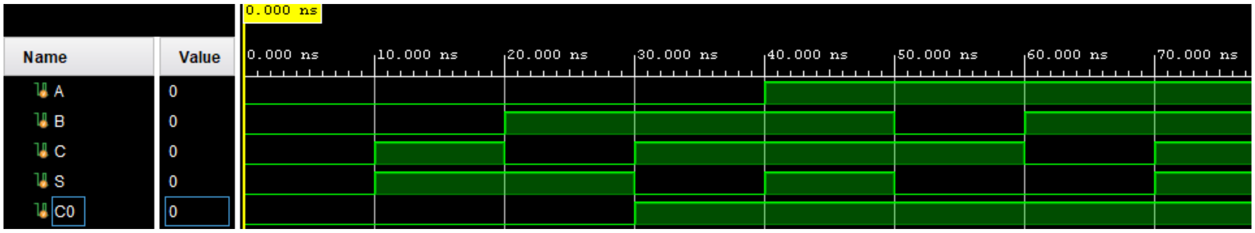
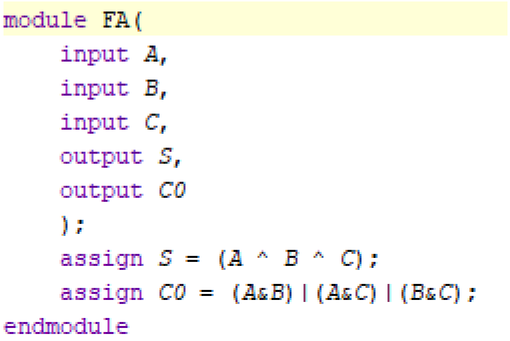


Figure 1.2.4: Schematic of the Full Adder circuit.



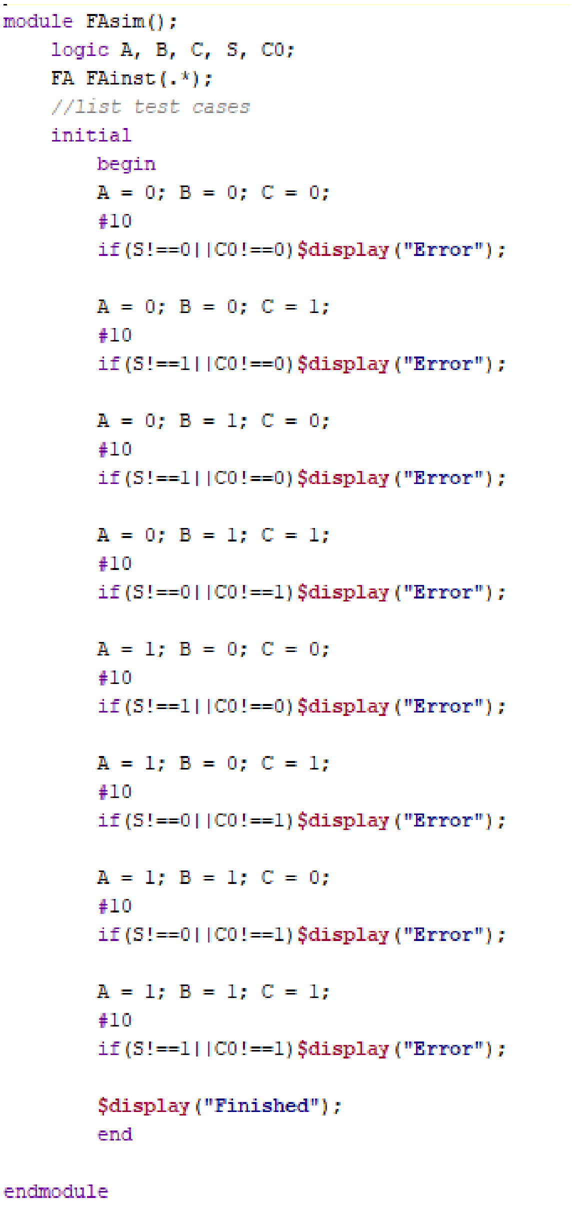
Name	Value
A	0
B	0
C	0
S	0
C0	0

Figure 1.2.5: Timing diagram for the Full Adder. Showcases all possible inputs and their outputs.



```
module FA(
    input A,
    input B,
    input C,
    output S,
    output C0
    );
    assign S = (A ^ B ^ C);
    assign C0 = (A&B)|(A&C)|(B&C);
endmodule
```

Figure 1.2.6: Source code of the Full Adder. Showcases the three inputs, two outputs, and the boolean logic behind the circuit.



```
module FAsim();
    logic A, B, C, S, C0;
    FA FAinst(.*);
    //list test cases
    initial
        begin
        A = 0; B = 0; C = 0;
        #10
        if(S!==0||C0!==0)$display("Error");

        A = 0; B = 0; C = 1;
        #10
        if(S!==1||C0!==0)$display("Error");

        A = 0; B = 1; C = 0;
        #10
        if(S!==1||C0!==0)$display("Error");

        A = 0; B = 1; C = 1;
        #10
        if(S!==0||C0!==1)$display("Error");

        A = 1; B = 0; C = 0;
        #10
        if(S!==1||C0!==0)$display("Error");

        A = 1; B = 0; C = 1;
        #10
        if(S!==0||C0!==1)$display("Error");

        A = 1; B = 1; C = 0;
        #10
        if(S!==0||C0!==1)$display("Error");

        A = 1; B = 1; C = 1;
        #10
        if(S!==1||C0!==1)$display("Error");

        $display("Finished");
        end

endmodule
```

Figure 1.2.7: Simulation code for the Full Adder. Showcases each test case.

Part 3

Description:

- In this lab, we will experiment and work with three different modules. These will be a 2-bit half adder, a 3-bit full adder, and a ripple carry adder (RCA).

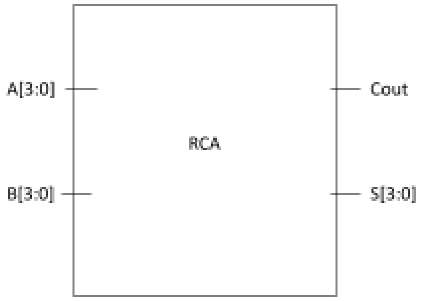


Figure 1.3.0: Black Box Diagram showing the 2 inputs (A[3:0],B[3:0]) and the output S[3:0], and carryout, C

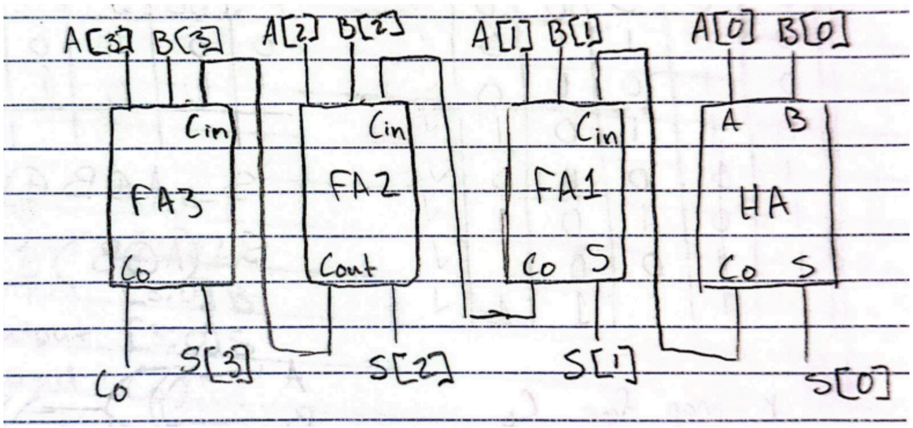


Figure 1.3.1: RCA structural model diagram

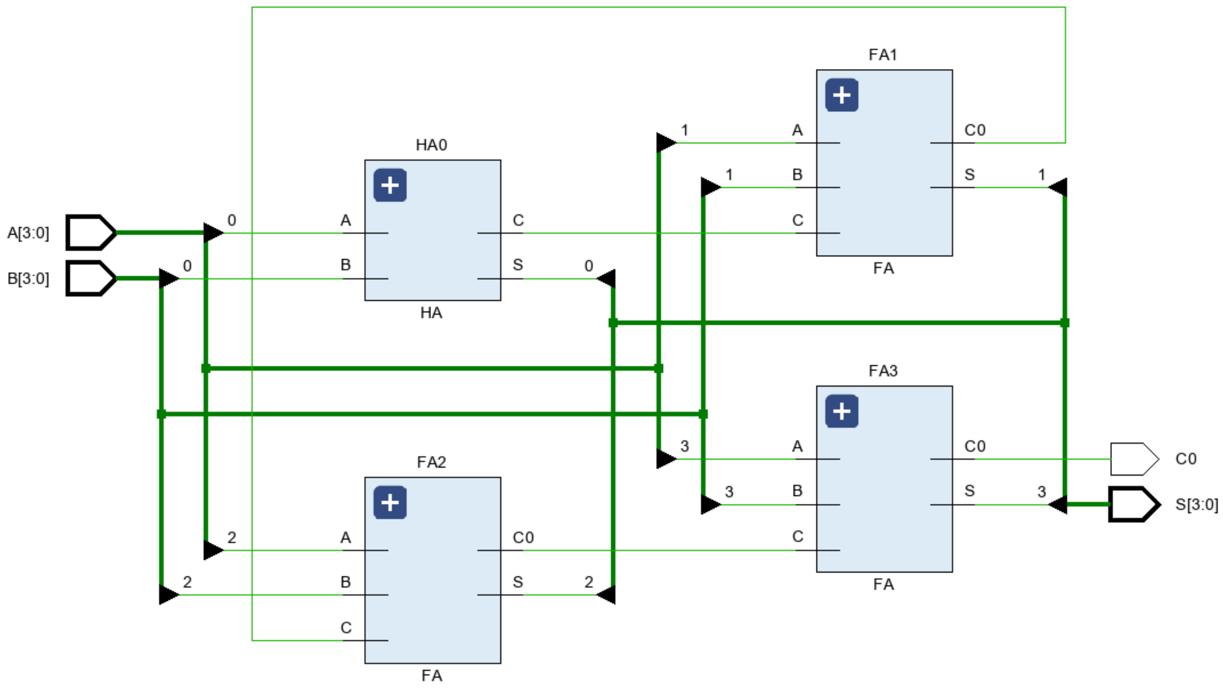
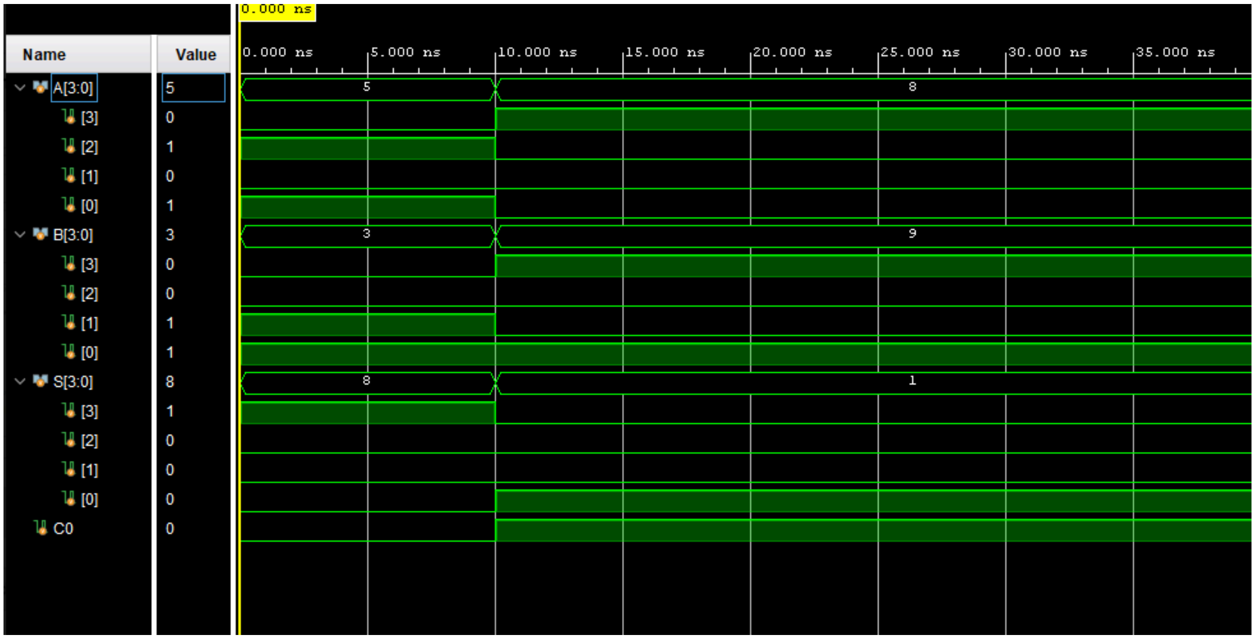
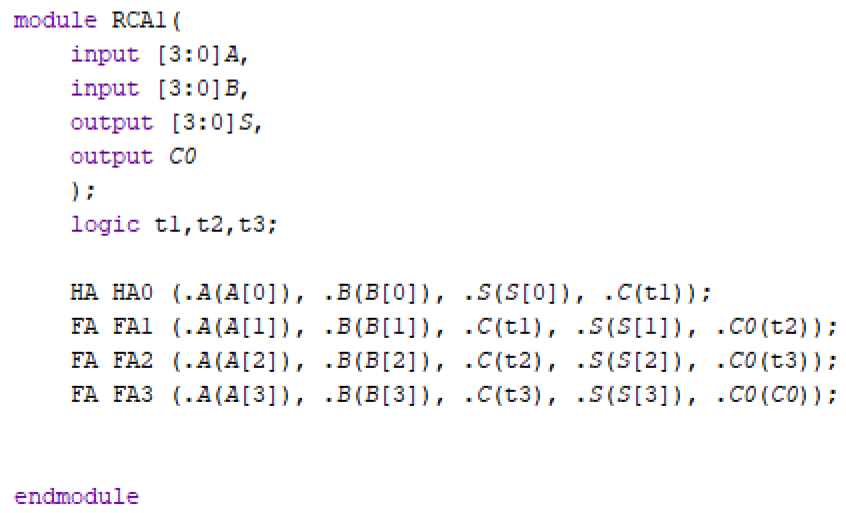


Figure 1.3.2: Schematic diagram of the RCA circuit. Showcases the structure of the circuit, the inputs and outputs.



Name	Value
A[3:0]	5
[3]	0
[2]	1
[1]	0
[0]	1
B[3:0]	3
[3]	0
[2]	0
[1]	1
[0]	1
S[3:0]	8
[3]	1
[2]	0
[1]	0
[0]	0
C0	0

Figure 1.3.3: Timing diagram of the RCA circuit. Showcases the possible inputs and their respective results.

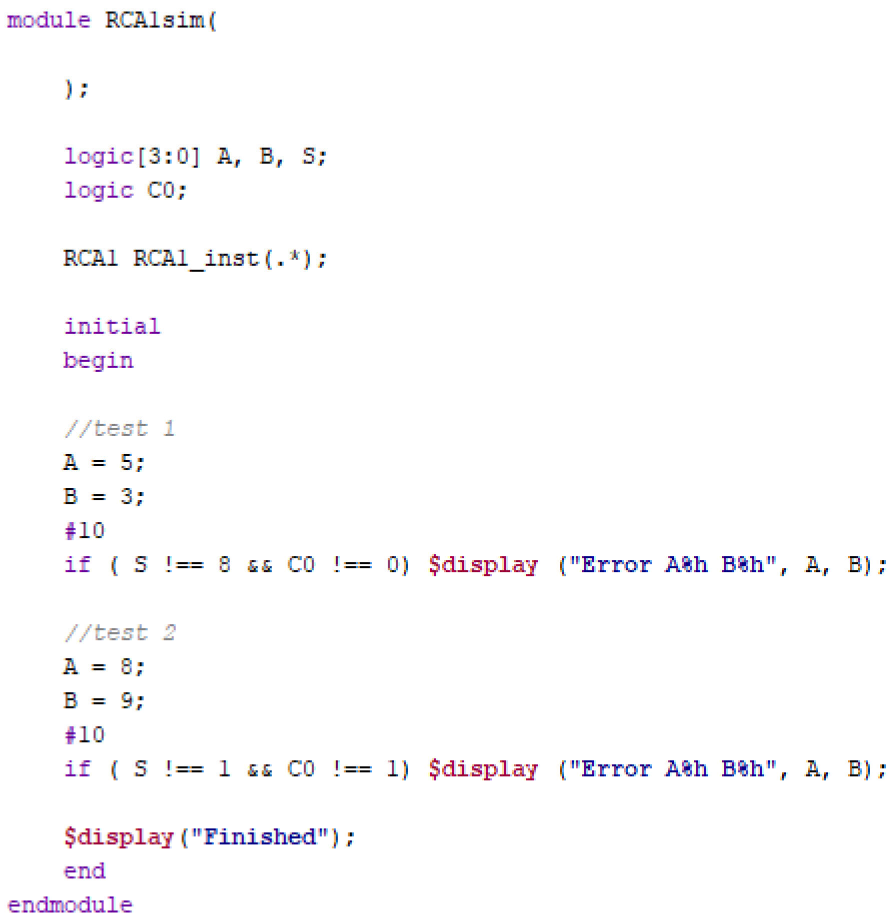


```
module RCA1(
    input [3:0]A,
    input [3:0]B,
    output [3:0]S,
    output CO
    );
    logic t1,t2,t3;

    HA HA0 (.A(A[0]), .B(B[0]), .S(S[0]), .C(t1));
    FA FA1 (.A(A[1]), .B(B[1]), .C(t1), .S(S[1]), .CO(t2));
    FA FA2 (.A(A[2]), .B(B[2]), .C(t2), .S(S[2]), .CO(t3));
    FA FA3 (.A(A[3]), .B(B[3]), .C(t3), .S(S[3]), .CO(CO));


endmodule
```

Figure 1.3.4: Source code of the ripple carry adder. Showcases the inputs, and logic behind the circuit.



```
module RCA1sim(

    );

    logic[3:0] A, B, S;
    logic C0;

    RCA1 RCA1_inst(.*);

    initial
    begin

    //test 1
    A = 5;
    B = 3;
    #10
    if ( S !== 8 && C0 !== 0) $display ("Error A%h B%h", A, B);

    //test 2
    A = 8;
    B = 9;
    #10
    if ( S !== 1 && C0 !== 1) $display ("Error A%h B%h", A, B);

    $display("Finished");
    end
endmodule
```

Figure 1.3.5: Simulation code of the RCA circuit. Showcases the two test cases.