EE 316 – Lab 5 Report: Calculator Design with Adders & Registers

Ayan Basu [EID: ab73287] (Section: 17760)

Saptarshi Mondal [EID: sm72999] (Section: 17760)

Part 1

Design File (.v) for Ripple Carry Adder

```
module RCA_4bits(
   input clk,
   input enable,
   input [3:0] A, B,
   input Cin,
   output [4:0] Q
   );
   wire ripple0, ripple1, ripple2, ripple3;

   //assign Cin = 1'b0;

   // Module instantiation for 4 bit RCA
   full_adder a0(.A(A[0]), .B(B[0]), .Cin(Cin), .S(Q[0]), .Cout(ripple0));
   full_adder a1(.A(A[1]), .B(B[1]), .Cin(ripple0), .S(Q[1]), .Cout(ripple1));
   full_adder a2(.A(A[2]), .B(B[2]), .Cin(ripple1), .S(Q[2]), .Cout(ripple2));
   full_adder a3(.A(A[3]), .B(B[3]), .Cin(ripple2), .S(Q[3]), .Cout(Q[4]));

   // Module Instantiation for Register
   register_logic register(.clk(clk), .enable(enable), .Data(Q), .Q());

endmodule
```

Test Bench File

```
`timescale
1ns / 1ps
            module tb_RCA;

            reg clk;
            reg enable;
            reg [3:0] A;
            reg [3:0] B;
            reg Cin;
            wire [4:0] Q;

            RCA_4bits ul (
            .clk(clk),
            .enable(enable),
            .A(A),
            .B(B),
            .Cin(Cin),
            .Q(Q)
            );

            initial begin

            clk = 0;

            enable = 0;
            A = 4'b0001;
            B = 4'b0101;
            Cin = 1'b0;
            enable = 1;
            #10

            enable = 0;
            A = 4'b0111;
            B = 4'b0111;
            Cin = 1'b0;
            enable = 1;
            #10

            enable = 0;
            A = 4'b1000;
            B = 4'b0111;
            Cin = 1'b1;
            enable = 1;
```

```
#10

enable = 0;
A = 4'b1100;
B = 4'b0100;
Cin = 1'b0;
enable = 1;
#10

enable = 0;
A = 4'b1000;
B = 4'b1000;
Cin = 1'b1;
enable = 1;
#10

enable = 0;
A = 4'b1001;
B = 4'b1010;
Cin = 1'b1;
enable = 1;
#10

enable = 0;
A = 4'b1111;
B = 4'b1111;
Cin = 1'b0;
enable = 1;

end

always
#5 clk = ~clk;

endmodule
```

Completed Table

A[3:0]	B[3:0]	Cin	Sum[3:0]	Cout
0001	0101	0	0110	0
0111	0111	0	1110	0
1000	0111	1	0000	1
1100	0100	0	0000	1
1000	1000	1	0001	1
1001	1010	1	0100	1
1111	1111	0	1110	1

Constraints File

Clock signal

set_property PACKAGE_PIN W5 [get_ports clk]

set_property IOSTANDARD LVCMOS33 [get_ports clk]

create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]

set_property PACKAGE_PIN U18 [get_ports enable]

set_property IOSTANDARD LVCMOS33 [get_ports enable]

LEDs

set_property PACKAGE_PIN U16 [get_ports {Q[0]}]

set_property IOSTANDARD LVCMOS33 [get_ports {Q[0]}]

set_property PACKAGE_PIN E19 [get_ports {Q[1]}]

set_property IOSTANDARD LVCMOS33 [get_ports {Q[1]}]

set_property PACKAGE_PIN U19 [get_ports {Q[2]}]

set_property IOSTANDARD LVCMOS33 [get_ports {Q[2]}]

set_property PACKAGE_PIN V19 [get_ports {Q[3]}]

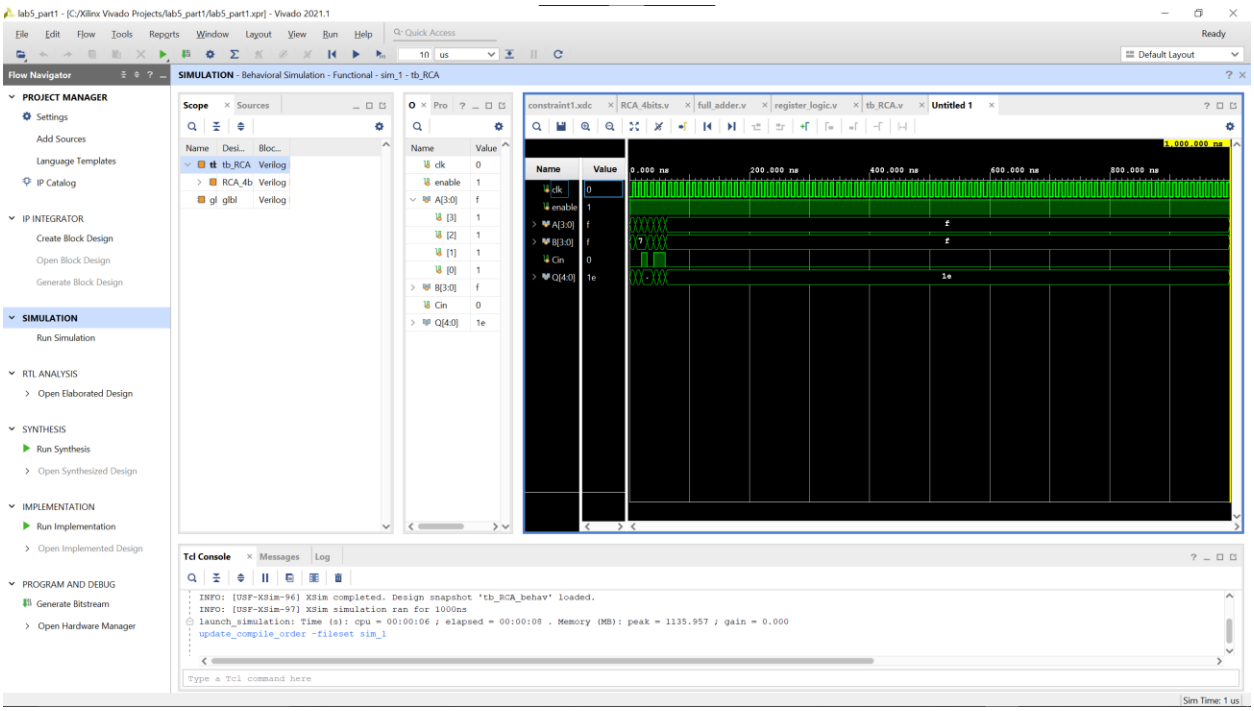
set_property IOSTANDARD LVCMOS33 [get_ports {Q[3]}]

set_property PACKAGE_PIN W18 [get_ports {Q[4]}]

set_property IOSTANDARD LVCMOS33 [get_ports {Q[4]}]

```
## Switches
   set_property PACKAGE_PIN V17 [get_ports {A[0]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {A[0]}]
   set_property PACKAGE_PIN V16 [get_ports {A[1]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {A[1]}]
   set_property PACKAGE_PIN W16 [get_ports {A[2]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {A[2]}]
   set_property PACKAGE_PIN W17 [get_ports {A[3]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {A[3]}]
   set_property PACKAGE_PIN W15 [get_ports {B[0]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {B[0]}]
   set_property PACKAGE_PIN V15 [get_ports {B[1]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {B[1]}]
   set_property PACKAGE_PIN W14 [get_ports {B[2]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {B[2]}]
   set_property PACKAGE_PIN W13 [get_ports {B[3]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {B[3]}]
   set_property PACKAGE_PIN V2 [get_ports {Cin}]
      set_property IOSTANDARD LVCMOS33 [get_ports {Cin}]
```

Simulation Waveforms



Part 2

Equations from Cs & Ss

C[0] = Cin

C[1] = G[0] + (P[0]C[0])

C[2] = G[1] + (P[1]G[0]) + (P[1]P[0]C[0])

C[3] = G[2] + (P[2]G[1]) + (P[2]P[1]G[0]) + (P[2]P[1]P[0]C[0])

C[4] = G[3] + (P[3]G[2]) + (P[3]P[2]G[1]) + (P[3]P[2]P[1]G[0]) + (P[3]P[2]P[1]P[0]C[0])

Design File (.v) for Carry Look Adder

```
module CLA_4bits(
    input clk,
    input enable,
    input [3:0] A, B,
    input Cin,
    output [4:0] Q
    );
```

```
    wire [3:0] G, P, S;
    wire [4:0] C;
    wire Cout;
    wire [4:0] Data;

    //assign Q = {Cout, S};

    // P = a ^ b
    // G = a&b
    // S = P ^ C
    // C = G + (P & G)

    assign P[0] = A[0] ^ B[0];
    assign P[1] = A[1] ^ B[1];
    assign P[2] = A[2] ^ B[2];
    assign P[3] = A[3] ^ B[3];

    assign G[0] = A[0] & B[0];
    assign G[1] = A[1] & B[1];
    assign G[2] = A[2] & B[2];
    assign G[3] = A[3] & B[3];

    assign C[0] = Cin;
    assign C[1] = G[0] | (P[0]&C[0]);
    assign C[2] = (P[1]&P[0]&C[0]) | (P[1]&G[0]) | G[1];
    assign C[3] = (P[2]&P[1]&P[0]&C[0]) | (P[2]&P[1]&G[0]) | (P[2]&G[1]) | G[2];
    assign C[4] = (P[3]&P[2]&P[1]&P[0]&C[0]) | (P[3]&P[2]&P[1]&G[0]) | (P[3]&P[2]&G[1]) | (P[3]&G[2]) |
G[3];
    assign Cout = C[4];
```

```
    assign S[0] = P[0] ^ C[0];
    assign S[1] = P[1] ^ C[1];
    assign S[2] = P[2] ^ C[2];
    assign S[3] = P[3] ^ C[3];

    assign Data = {Cout, S};
//    assign Data[4] = Cout;
//    assign Data[3:0] = S;

    register_logic register(.clk(clk), .enable(enable), .Data(Data), .Q(Q));

endmodule
```

Design File (.v) for Register Logic

```
module register_logic(
    input clk,
    input enable,
    input [4:0] Data,
    output reg [4:0] Q
    );

    //RCA_4bits rca(.clk(clk), .enable(enable),

    always @ (posedge clk)
        if(enable)
            Q = Data;
endmodule
```

Test Bench File

```
module tb_CLA;

reg clk;
reg enable;
reg [3:0] A;
reg [3:0] B;
reg Cin;
wire [4:0] Q;

CLA_4bits ul (
.clk(clk),
.enable(enable),
.A(A),
.B(B),
.Cin(Cin),
.Q(Q)
);

initial begin

clk = 0;

enable = 0;
A = 4'b0000;
B = 4'b0101;
Cin = 1'b0;
enable = 1;
#10
```

```
enable = 0;
A = 4'b0101;
B = 4'b0111;
Cin = 1'b0;
enable = 1;
#10

enable = 0;
A = 4'b1000;
B = 4'b0111;
Cin = 1'b1;
enable = 1;
#10

enable = 0;
A = 4'b1001;
B = 4'b0100;
Cin = 1'b0;
enable = 1;
#10

enable = 0;
A = 4'b1000;
B = 4'b1000;
Cin = 1'b1;
enable = 1;
#10
```

```
enable = 0;

A = 4'b1101;

B = 4'b1010;

Cin = 1'b1;

enable = 1;

#10


enable = 0;

A = 4'b1110;

B = 4'b1111;

Cin = 1'b0;

enable = 1;


end


always

#5 clk = ~clk;


Endmodule
```

Completed Table for Part 2

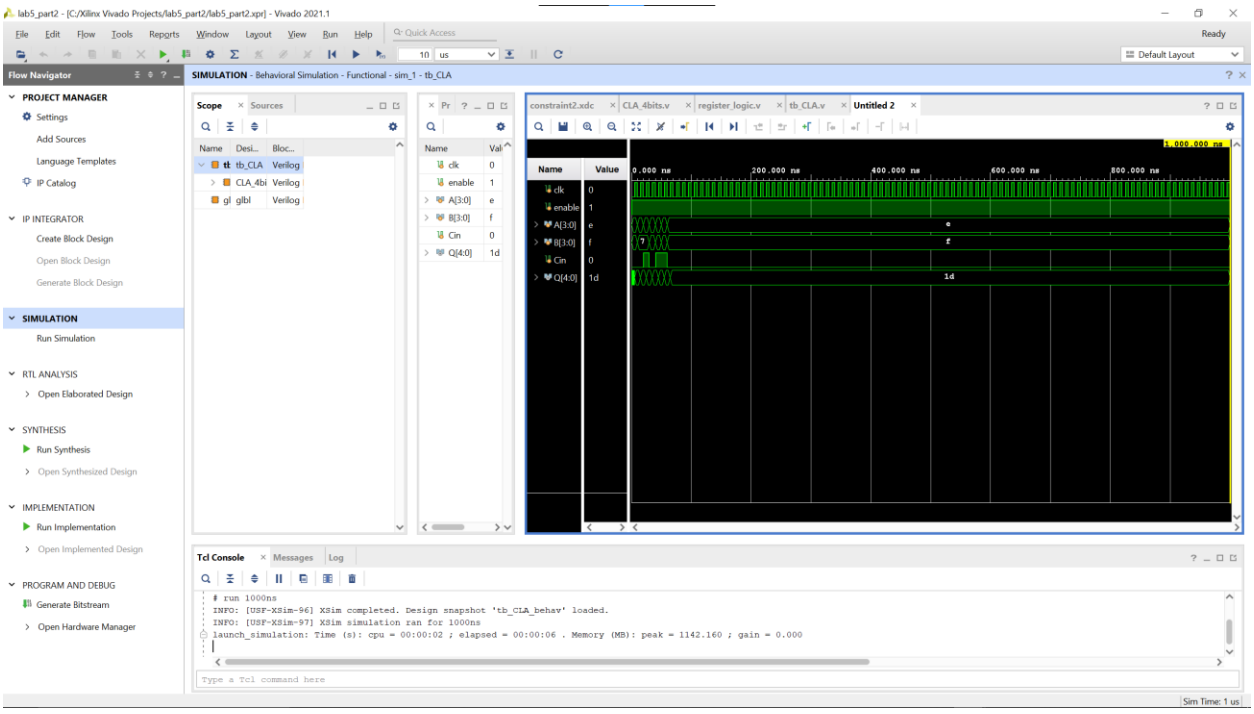
A[3:0]	B[3:0]	Cin	Sum[3:0]	Cout
0000	0101	0	0101	0
0101	0111	0	1100	0
1000	0111	1	0000	1
1001	0100	0	1101	0
1000	1000	1	0001	1
1101	1010	1	1000	1
1110	1111	0	1101	1

Constraints File

```
## Clock signal
set_property PACKAGE_PIN W5 [get_ports clk]
        set_property IOSTANDARD LVCMOS33 [get_ports clk]
        create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]

set_property PACKAGE_PIN U18 [get_ports enable]
   set_property IOSTANDARD LVCMOS33 [get_ports enable]

## LEDs
set_property PACKAGE_PIN U16 [get_ports {Q[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {Q[0]}]
set_property PACKAGE_PIN E19 [get_ports {Q[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {Q[1]}]
set_property PACKAGE_PIN U19 [get_ports {Q[2]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {Q[2]}]
set_property PACKAGE_PIN V19 [get_ports {Q[3]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {Q[3]}]
set_property PACKAGE_PIN W18 [get_ports {Q[4]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {Q[4]}]

## Switches
   set_property PACKAGE_PIN V17 [get_ports {A[0]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {A[0]}]
   set_property PACKAGE_PIN V16 [get_ports {A[1]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {A[1]}]
   set_property PACKAGE_PIN W16 [get_ports {A[2]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {A[2]}]
```

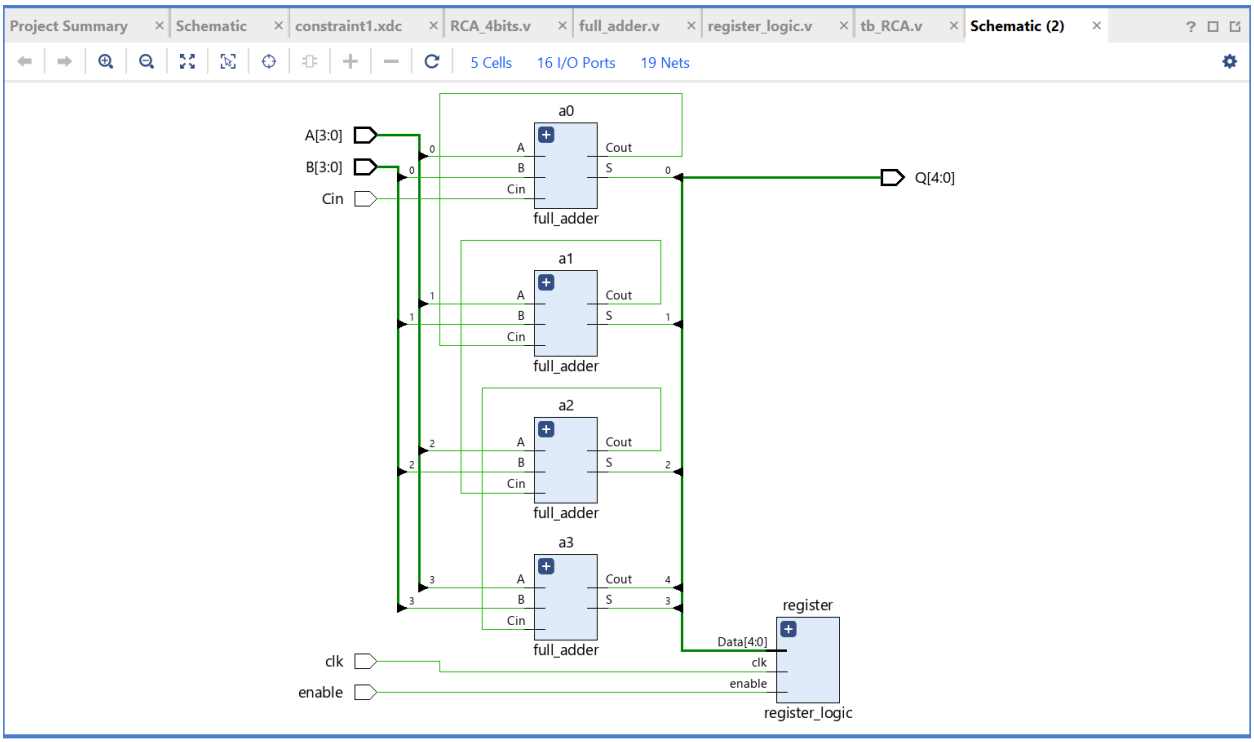
```
set_property PACKAGE_PIN W17 [get_ports {A[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {A[3]}]
set_property PACKAGE_PIN W15 [get_ports {B[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {B[0]}]
set_property PACKAGE_PIN V15 [get_ports {B[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {B[1]}]
set_property PACKAGE_PIN W14 [get_ports {B[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {B[2]}]
set_property PACKAGE_PIN W13 [get_ports {B[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {B[3]}]
set_property PACKAGE_PIN V2 [get_ports {Cin}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Cin}]
```

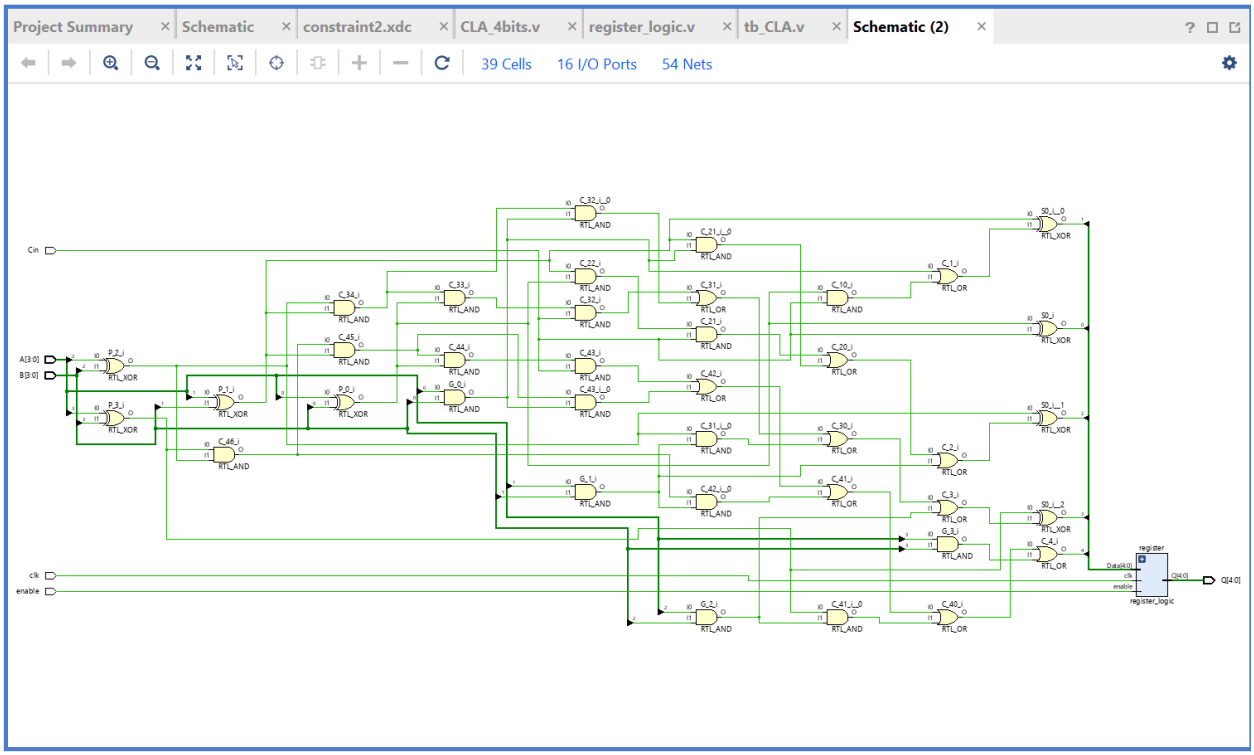
Simulated Waveform



Part 3

Screenshots of Schematic Diagrams from Both Adder Techniques





<u>Delay and area for both the adder techniques showing all the work</u>

- Ripple Carry Adder
    - Delay = 28 ns (By following Cin to Cout) Each full adder has a critical path going from 2nd AND -> 1st OR -> 1st OR (3+2+2 = 7ns)*4 = 28 ns
    - Each full adder contains 3 ANDs, 2 ORs, 2 XORs. Using the table and the given areas for each logic gate, we can calculate the area: 3(4) + 2(4) + 2(6) = 12 + 8 + 12 = 32 (for 1 full adder)
      4 bit RCA area = 32 * 4 = 128
- Carry Look-ahead Adder
    - Delay = 23 ns(By following the red-dotted path) (3+3+3+3+3+2+2+2+2 = 23 ns)
    - The above RTL design of the CLA contains 20 ANDs, 10 ORs, 8 XORs.
    - 4 bit CLA area = 4(20) + 4(10) + 6(8) = 168

Brief conclusion regarding the pros and cons of each of the techniques

Each of the adder implementations have their respective pros and cons. Moreover, the Ripple Carry Adder is slower and uses more space. However, it is cheaper to create because it has less gates. When doing a smaller bit width, the Ripple Carry Adder is preferable. The Ripple Carry Adder is easier to trace logic and examine. On the other hand, the Carry Look-ahead Adder is faster since the g(x) and p(x) are calculated at the same time

- Ripple Carry Adder (RCA)
    - Pros
        - Small area = less logic gates, which means that production costs will be cheaper
    - Cons
        - Relatively slow; each Cin is an output from a previous adder (other than the LSB), therefore calculating the sum will take some time in a “ripple” manner
- Carry Look-ahead Adder (CLA)
    - Pros
        - More efficient; since each carry signal is calculated in advance based on input signals.
    - Cons
        - As we have higher bit CLAs, the logic blocks get more and more complex, thus having to implement more logic gates than the RCA