Part 1

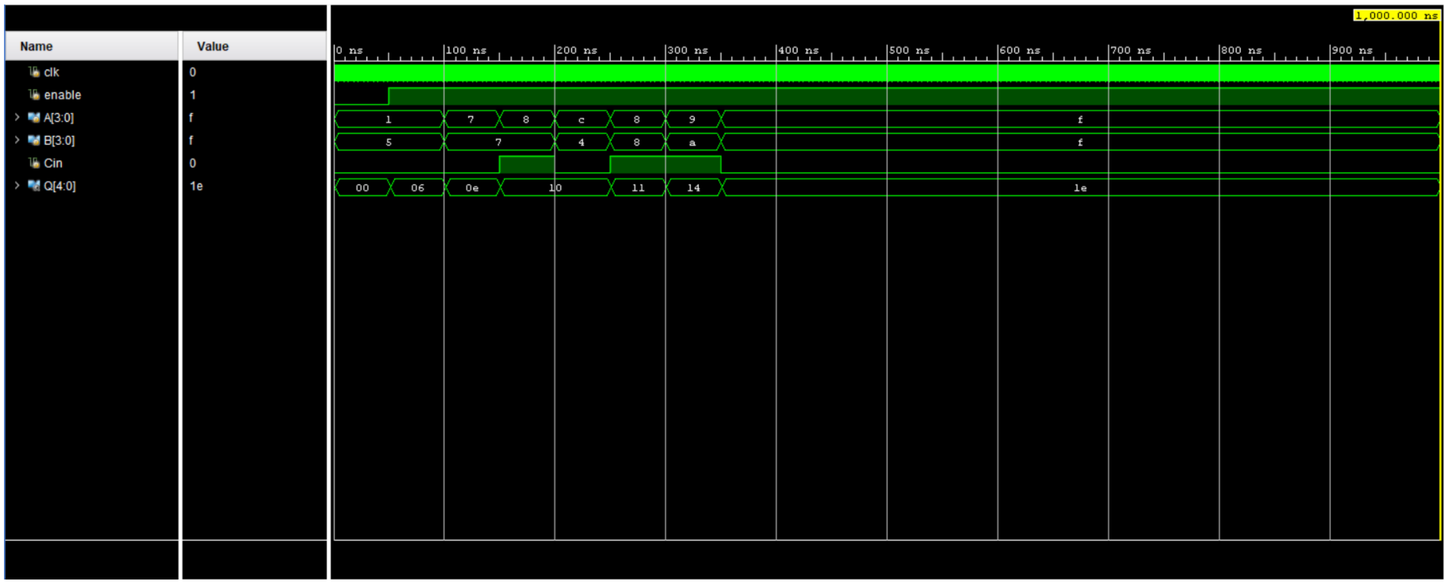
```
module RCA_4bits(
    input clk,
    input enable,
    input[3:0] A,
    input[3:0] B,
    input Cin,
    output[4:0] Q
    );

wire s0, s1, s2, s3, o0, o1, o2, o3;
reg[4:0] Data;

full_adder c1 (.A(A[0]), .B(B[0]), .Cin(Cin), .S(s0), .Cout(o0));
full_adder c2 (.A(A[1]), .B(B[1]), .Cin(o0), .S(s1), .Cout(o1));
full_adder c3 (.A(A[2]), .B(B[2]), .Cin(o1), .S(s2), .Cout(o2));
full_adder c4 (.A(A[3]), .B(B[3]), .Cin(o2), .S(s3), .Cout(o3));

always @(*) begin
    Data = {o3, s3, s2, s1, s0};
end

register_logic c5 (.clk(clk), .enable(enable), .Data(Data), .Q(Q));

endmodule

module full_adder(
    input A,
    input B,
    input Cin,
    output S,
    output Cout
    );

assign S = (A ^ B ^ Cin);
assign Cout = ((A & B) | (A & Cin) | (B & Cin));

endmodule

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

always @(posedge clk) begin
    if(enable) Q = Data;
    else Q = 0;
end

endmodule
```

```
module tb_RCA_4bits;

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

RCA_4bits uut (
   .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;

#50;

enable = 1;

#50;

A = 4'b0111;
B = 4'b0111;
Cin = 1'b0;

#50;

A = 4'b1000;
B = 4'b0111;
Cin = 1'b1;

#50;

A = 4'b1100;
B = 4'b0100;
Cin = 1'b0;

#50;

A = 4'b1000;
B = 4'b1000;
```

```
Cin = 1'b1;

#50;

A = 4'b1001;
B = 4'b1010;
Cin = 1'b1;

#50;

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

#50;

end

always
#1 clk = ~clk;

endmodule
```

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

```
# 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]
## 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}]
## 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]}]
##Buttons
set_property PACKAGE_PIN U18 [get_ports {enable}]
        set_property IOSTANDARD LVCMOS33 [get_ports {enable}]
```



Part 2

$C_1 = P_0 C_0 + G_0$

$C_2 = P_1 C_1 + G_1 = P_1 P_0 C_0 + P_1 G_0 + G_1$

$C_3 = P_2 C_2 + G_2 = P_2(P_1 P_0 C_0 + P_1 G_0 + G_1) + G_2 = P_2 P_1 P_0 C_0 + P_2 P_1 G_0 + P_2 G_1 + G_2$

$C_4 = P_3 C_3 + G_3 = P_3(P_2 P_1 P_0 C_0 + P_2 P_1 G_0 + P_2 G_1 + G_2) = P_3 P_2 P_1 P_0 C_0 + P_3 P_2 P_1 G_0 + P_3 P_2 G_1 + P_3 G_2 + G_3$

$S_0 = P_0 \oplus C_0$

$S_1 = P_1 \oplus C_1$

$S_2 = P_2 \oplus C_2$

$S_3 = P_3 \oplus C_3$

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

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

assign C[0] = Cin;

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 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 C[1] = (P[0] & C[0]) | G[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 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];

register_logic c1 (.clk(clk), .enable(enable), .Data({C[4], S}), .Q(Q));

endmodule

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

always @(posedge clk) begin
   if(enable) Q = Data;
   else Q = 0;
end

endmodule
```

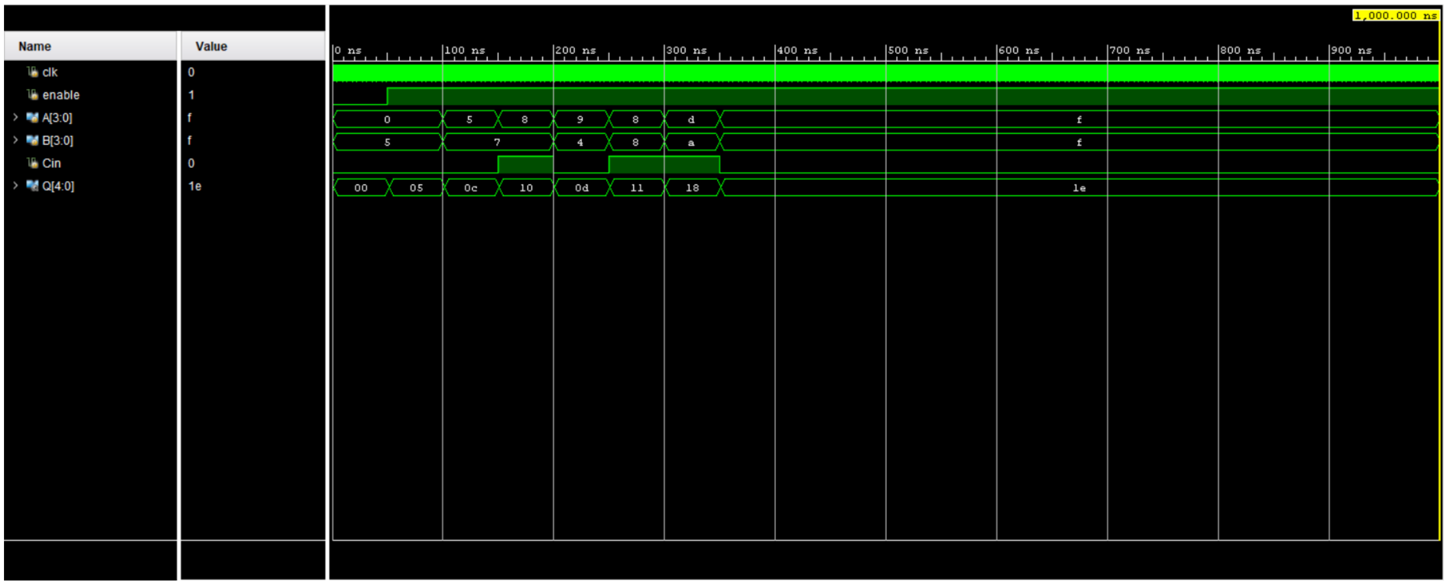
```
module tb_CLA_4bits;

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

CLA_4bits uut (
    .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;

#50;

enable = 1;

#50;

A = 4'b0101;
B = 4'b0111;
Cin = 1'b0;

#50;

A = 4'b1000;
B = 4'b0111;
Cin = 1'b1;

#50;

A = 4'b1001;
B = 4'b0100;
Cin = 1'b0;

#50;

A = 4'b1000;
B = 4'b1000;
```

```
Cin = 1'b1;

#50;

A = 4'b1101;
B = 4'b1010;
Cin = 1'b1;

#50;

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

#50;

end
```

A[3:0]	B[3:0]	Cin	Sum[3:0]	Cout
0000	0101	0	0110	0
0101	0111	0	1110	0
1000	0111	1	0000	1
1001	0100	0	0000	1
1000	1000	1	0001	1
1101	1010	1	0100	1
1111	1111	0	1110	1

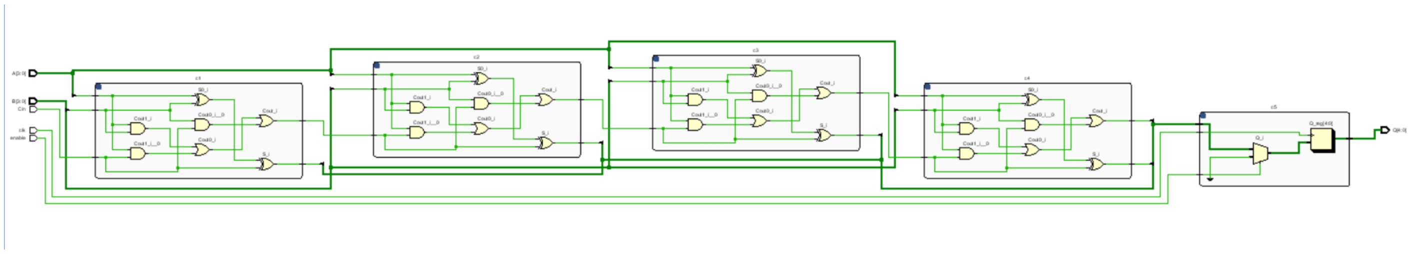
```
# Clock signal
set_property PACKAGE_PIN W5 [get_ports clk]
        set_property IOSTANDARD LVCMOS33 [get_ports clk]
        create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]
## Switches
set_property PACKAGE_PIN V17 [get_ports {A[0]}]
        set_property IOSTANDARD 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}]
## 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]}]
##Buttons
set_property PACKAGE_PIN U18 [get_ports {enable}]
        set_property IOSTANDARD LVCMOS33 [get_ports {enable}]
```

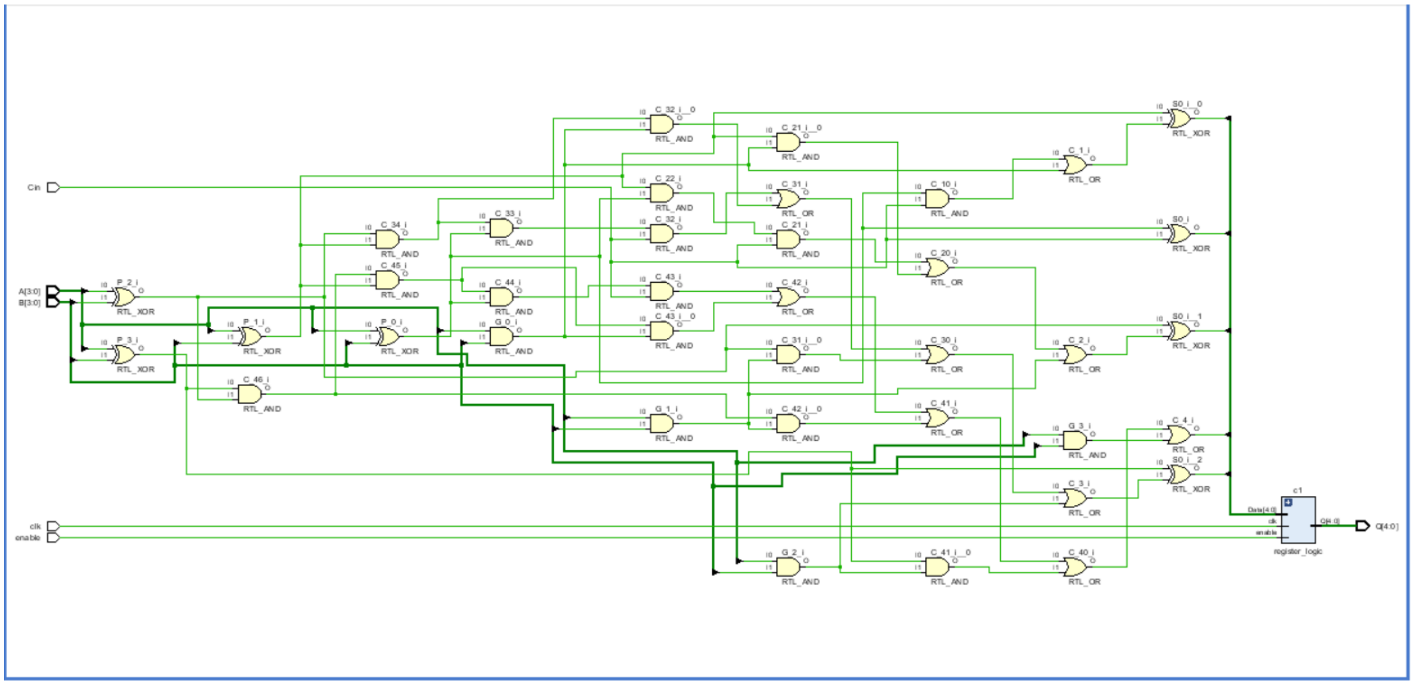


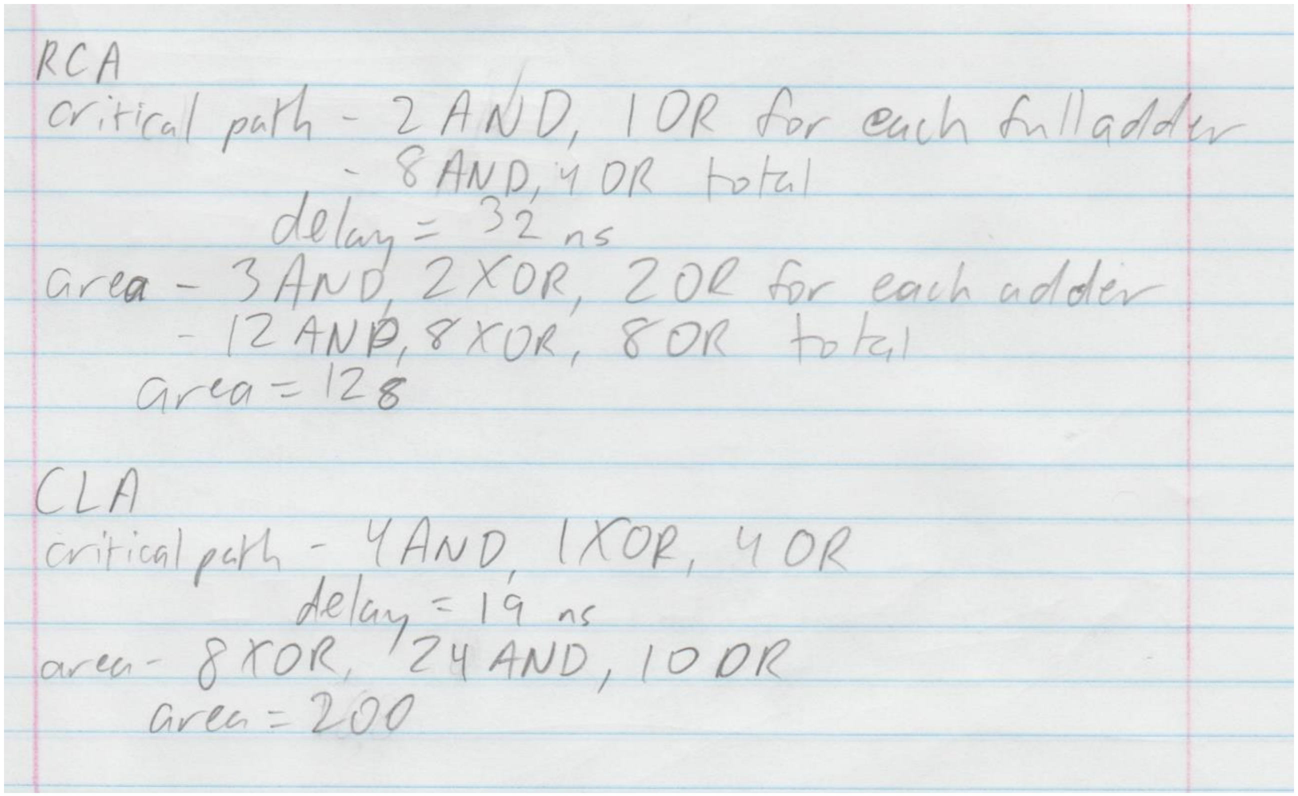
Part 3

RCA



CLA




RCA
critical path - 2 AND, 1 OR for each fulladder
- 8 AND, 4 OR total
delay = 32 ns
area - 3 AND, 2 XOR, 2 OR for each adder
- 12 AND, 8 XOR, 8 OR total
area = 128

CLA
critical path - 4 AND, 1 XOR, 4 OR
delay = 19 ns
area - 8 XOR, 24 AND, 10 OR
area = 200

The ripple carry adder is very easy to design due to the small number of gates and being able to string together many identical full adders to add longer operands, it uses fewer gates so would be cheaper to build and takes up less space. The carry look ahead adder uses more gates, takes up more space, and the design would have to be modified to change the length of the operands. However, the carry look ahead adder has a significantly shorter delay than the ripple carry adder, because the ripple carry has to add each bit one at a time to get the carry into the next, and the carry look ahead only needs the original carry in.