Lab 5

Tuesday, November 1, 2022 2:25 PM

Clark Simpler
cws2522
17510

Part 1

i. **RCA_4bits.v**

```
`timescale 1ns / 1ps

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

    wire co0, c01, co2, co3;
    wire [4:0] sum;

    //Adding all 4 bits
    full_adder s0(.A(A[0]), .B(B[0]), .Cin(Cin), .S(sum[0]), .Cout(co0));
    full_adder s1(.A(A[1]), .B(B[1]), .Cin(co0), .S(sum[1]), .Cout(co1));
    full_adder s2(.A(A[2]), .B(B[2]), .Cin(co1), .S(sum[2]), .Cout(co2));
    full_adder s3(.A(A[3]), .B(B[3]), .Cin(co2), .S(sum[3]), .Cout(co3));
    assign sum[4] = co3;

    register_logic r0(.clk(clk), .enable(enable), .Data(sum), .Q(Q));


endmodule
```

full_adder.v

```
`timescale 1ns / 1ps

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

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

endmodule
```

register_logic.v

```
`timescale 1ns / 1ps

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

    initial Q = 0;
    wire [4:0] Qin;
    assign Qin = enable ? Data : Q;

    //Sequential logic
    always @(posedge clk) begin
        Q <= Qin;
    end


endmodule
```

ii. **TB**

```
`timescale 1ns / 1ps

module TB_RCA_4bits;

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

RCA_4bits u1 (
    .clk(clk),
    .enable(enable),
    .A(A),
    .B(B),
    .Cin(Cin),
    .Q(Q)
    );

initial
begin

clk = 0;
enable = 0;

//Test 1
A = 4'b0001;
B = 4'b0111;
Cin = 0;

#10

enable = 1;

//Test 2
#10
A = 4'b0111;
B = 4'b0111;
Cin = 0;

//Test 3
#10
A = 4'b1000;
B = 4'b0111;
Cin = 1;

//Test 4
#10
A = 4'b1100;
B = 4'b0100;
Cin = 0;

//Test 5
#10
A = 4'b1000;
B = 4'b1000;
Cin = 1;

//Test 6
#10
A = 4'b1001;
B = 4'b1010;
Cin = 1;

//Test 7
#10
A = 4'b1111;
B = 4'b1111;
Cin = 0;

end

always
#5 clk = ~clk;

endmodule
```

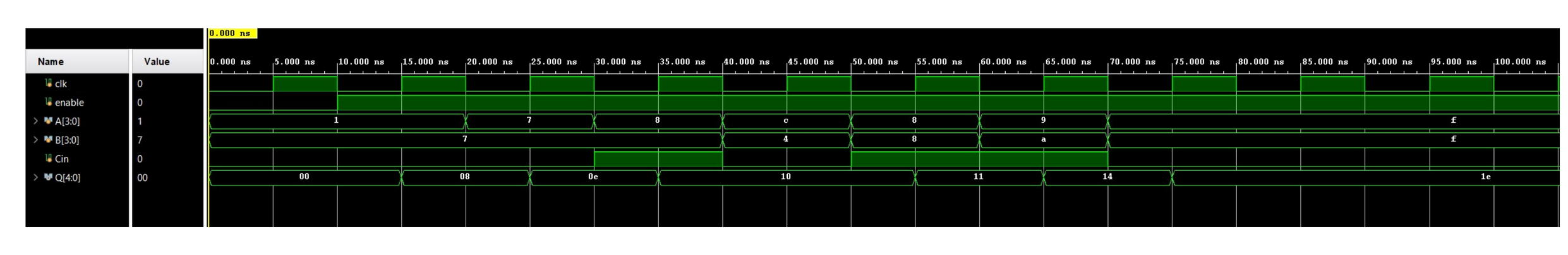
iii.

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	0010	1
1111	1111	0	1110	1

iv.

```
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 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}]

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]}]

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

v.



Part 2

vi. Cout:

$C_1 = G_0 + P_0 C_0$

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

$C_3 = G_2 + G_1 P_2 + G_0 P_1 P_2 + P_0 P_1 P_2 C_0$

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

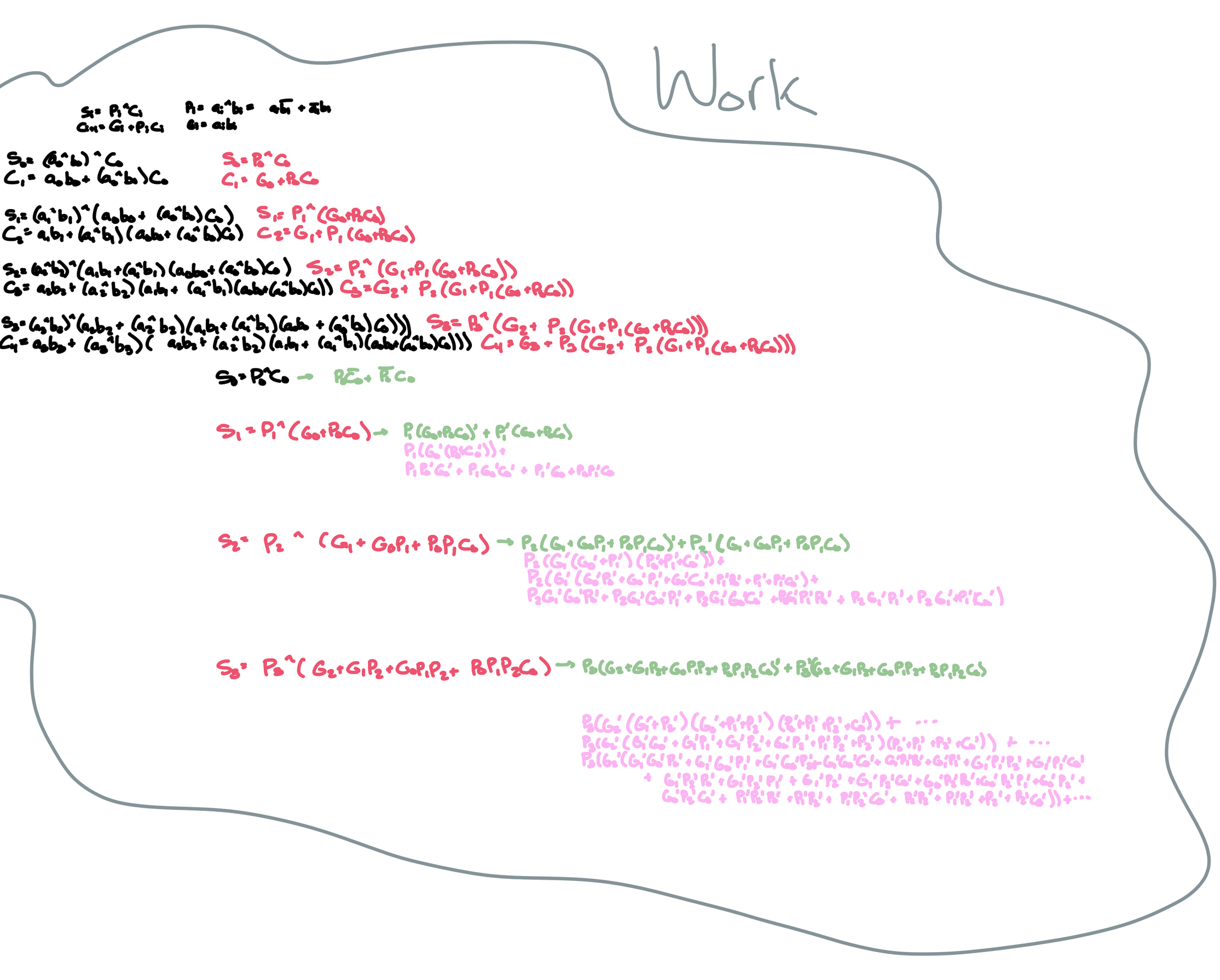
Sum:

$S_0 = P_0 \hat{} C_0$

$S_1 = P_1 \hat{} C_1$

$S_2 = P_2 \hat{} C_2$

$S_3 = P_3 \hat{} C_3$



vii. **CLA_4bits.v**

register_logic.v

Same as above

```
`timescale 1ns / 1ps

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

    wire [3:0] G, P;
    wire [4:0] C;
    wire [4:0] Sum;
    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] = G[0] | (P[0] & C[0]);
    assign C[2] = G[1] | (G[0] & P[1]) | (P[0] & P[1] & C[0]);
    assign C[3] = G[2] | (G[1] & P[2]) | (G[0] & P[1] & P[2]) | (P[0] & P[1] & P[2] & C[0]);
    assign C[4] = G[3] | (G[2] & P[3]) | (G[1] & P[2] & P[3]) | (G[0] & P[1] & P[2] & P[3]) | (P[0] & P[1] & P[2] & P[3] & C[0]);

    assign Sum[0] = P[0] ^ C[0];
    assign Sum[1] = P[1] ^ C[1];
    assign Sum[2] = P[2] ^ C[2];
    assign Sum[3] = P[3] ^ C[3];
    assign Sum[4] = C[4];

    register_logic r0(.clk(clk), .enable(enable), .Data(Sum), .Q(Q));


endmodule
```

viii. **TB**

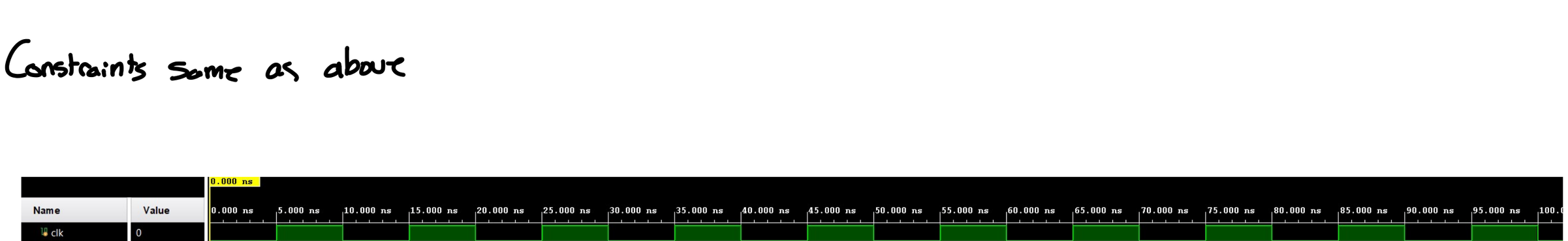
```
`timescale 1ns / 1ps

module TB_CLA_4bits;

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

CLA_4bits u1 (
    .clk(clk),
    .enable(enable),
    .A(A),
    .B(B),
    .Cin(Cin),
    .Q(Q)
    );

initial
begin

clk = 0;
enable = 0;

//Test 1
A = 4'b0000;
B = 4'b0101;
Cin = 0;

#10

enable = 1;

//Test 2
#10
A = 4'b0101;
B = 4'b0111;
Cin = 0;

//Test 3
#10
A = 4'b1000;
B = 4'b0111;
Cin = 1;

//Test 4
#10
A = 4'b1001;
B = 4'b0100;
Cin = 0;

//Test 5
#10
A = 4'b1000;
B = 4'b1000;
Cin = 1;

//Test 6
#10
A = 4'b1101;
B = 4'b1010;
Cin = 1;

//Test 7
#10
A = 4'b1111;
B = 4'b1111;
Cin = 0;

end

always
#5 clk = ~clk;

endmodule
```

ix.

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
1001	1010	1	0100	1
1111	1111	0	1110	1

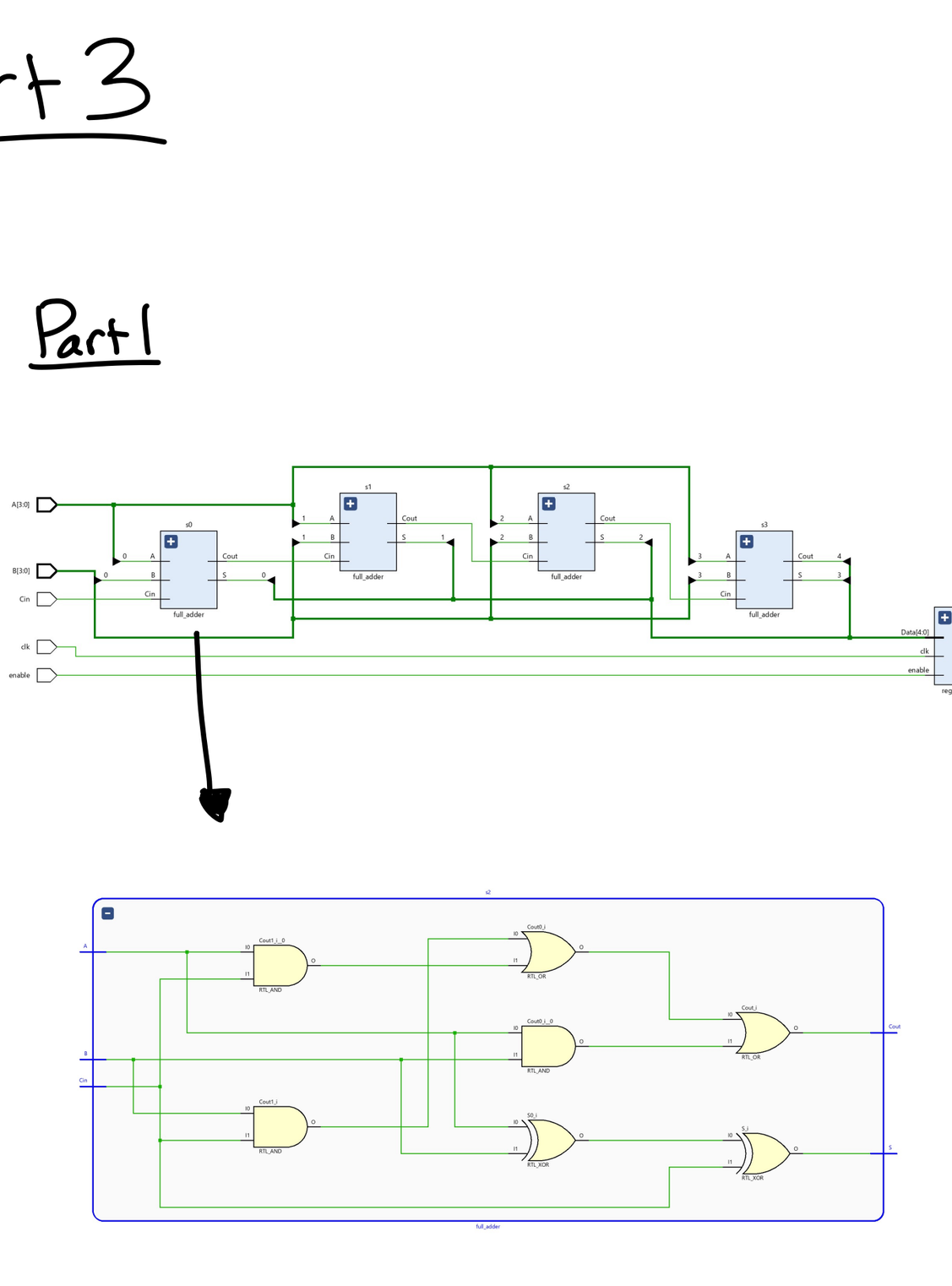
x. Constraints same as above

xi.

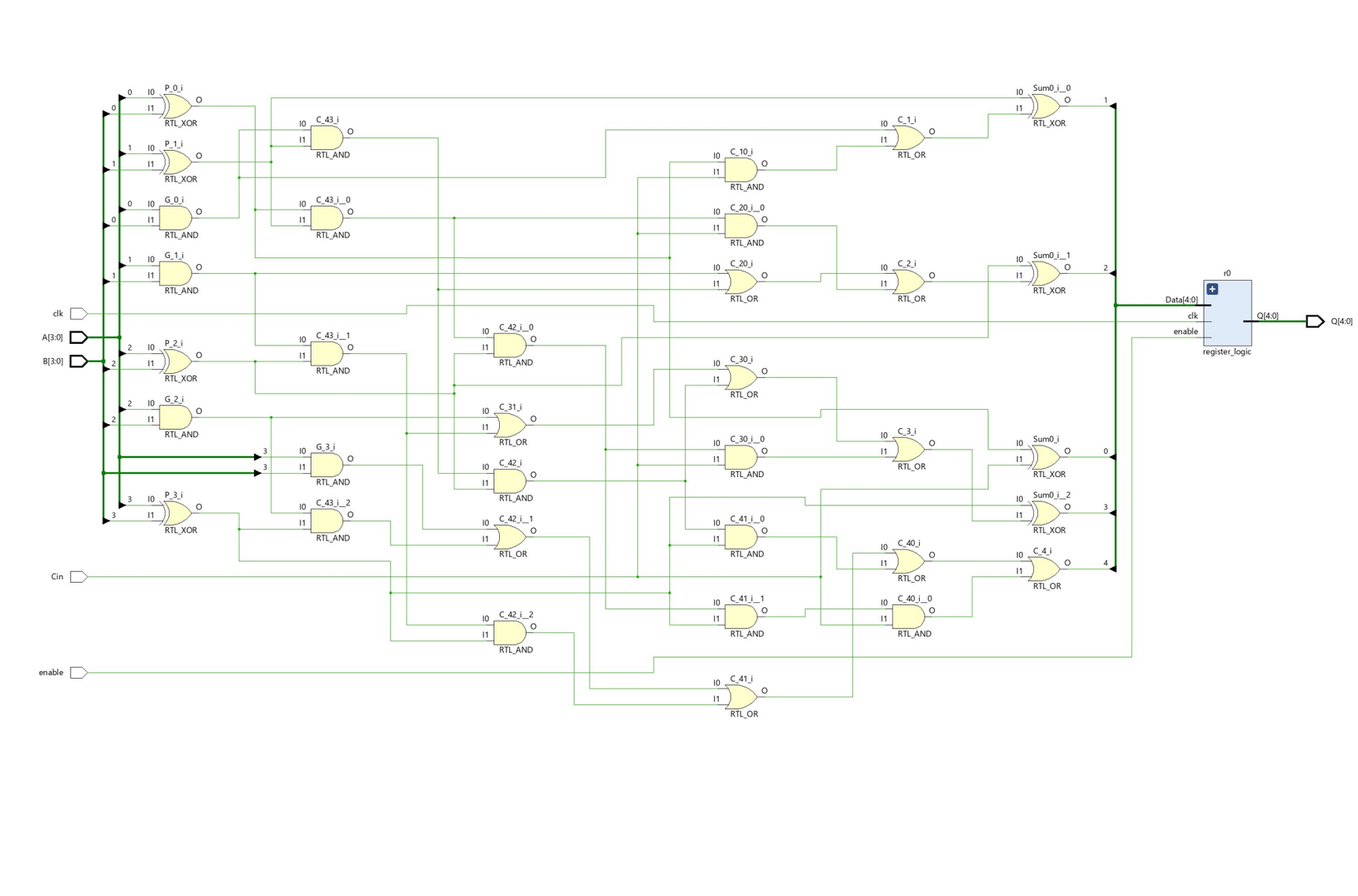


Part 3

xii. Part 1



Part 2



xiii.

RCA

Delay: $3+2+2 = 7 \times 4 = 28$ (AND OR OR)

Area: $3(4)+2(4)+2(6) = 32 \times 4 = 128$

CLA

Delay: $3+3+3+2+2+2 = 15$ (XOR AND AND OR OR OR)

Area: $8(6) + 17(4) + 10(4) = 156$ (XOR, AND, OR)

xiv. CLA is faster because don't have to wait for the Cout to ripple through. However to do this involves more gates so our area increases for CLA

RCA: good for constrained space, better readability, performance not critical

CLA: performance is most important