# Lab 5 Report

## Name: Christian Han

## UT EID: cjh3752

## Section: 15300

## Checklist:

### Part 1 –

i. Design file (.v) for the Ripple Carry Adder

```
`timescale 1ns / 1ps
module RCA_4bits(
   input clk,
   input enable,
   input [3:0] A,
   input [3:0] B,
   input Cin,
   output [4:0] Q //load registers, should contain the 4sum bits and Cout
   );

wire Cout0, Cout1, Cout2;
wire [4:0] Sum;

full_adder c1 (.A(A[0]), .B(B[0]), .Cin(Cin), .S(Sum[0]), .Cout(Cout0));
full_adder c2 (.A(A[1]), .B(B[1]), .Cin(Cout0), .S(Sum[1]), .Cout(Cout1));
full_adder c3 (.A(A[2]), .B(B[2]), .Cin(Cout1), .S(Sum[2]), .Cout(Cout2));
full_adder c4 (.A(A[3]), .B(B[3]), .Cin(Cout2), .S(Sum[3]), .Cout(Sum[4]));

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

endmodule
```

ii. Test-bench

```
`timescale 1ns / 1ps
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;

#50
enable = 1;
A = 4'b0001;
B = 4'b0101;
Cin = 1'b0;
```

```
#50
enable = 1;
A = 4'b0111;
B = 4'b0111;
Cin = 1'b0;

#50
enable = 1;
A = 4'b1000;
B = 4'b0111;
Cin = 1'b1;

#50
enable = 1;
A = 4'b1100;
B = 4'b0100;
Cin = 1'b0;

#50
enable = 1;
A = 4'b1000;
B = 4'b1000;
Cin = 1'b1;

#50
enable = 1;
A = 4'b1001;
B = 4'b1010;
Cin = 1'b1;

#50
enable = 1;
A = 4'b1111;
B = 4'b1111;
Cin = 1'b0;
```

```
end

always
#5 clk = ~clk;

endmodule
```

iii. Complete Table 1 from the simulation

| A[3:0] | B[3:0] | Cin | Sum[3:0] | Cout |
|---|---|---|---|---|
| 0001 | 0101 | 0 | 6 | 0 |
| 0111 | 0111 | 0 | E | 0 |
| 1000 | 0111 | 1 | 0 | 1 |
| 1100 | 0100 | 0 | 0 | 1 |
| 1000 | 1000 | 1 | 1 | 1 |
| 1001 | 1010 | 1 | 4 | 1 |
| 1111 | 1111 | 0 | E | 1 |

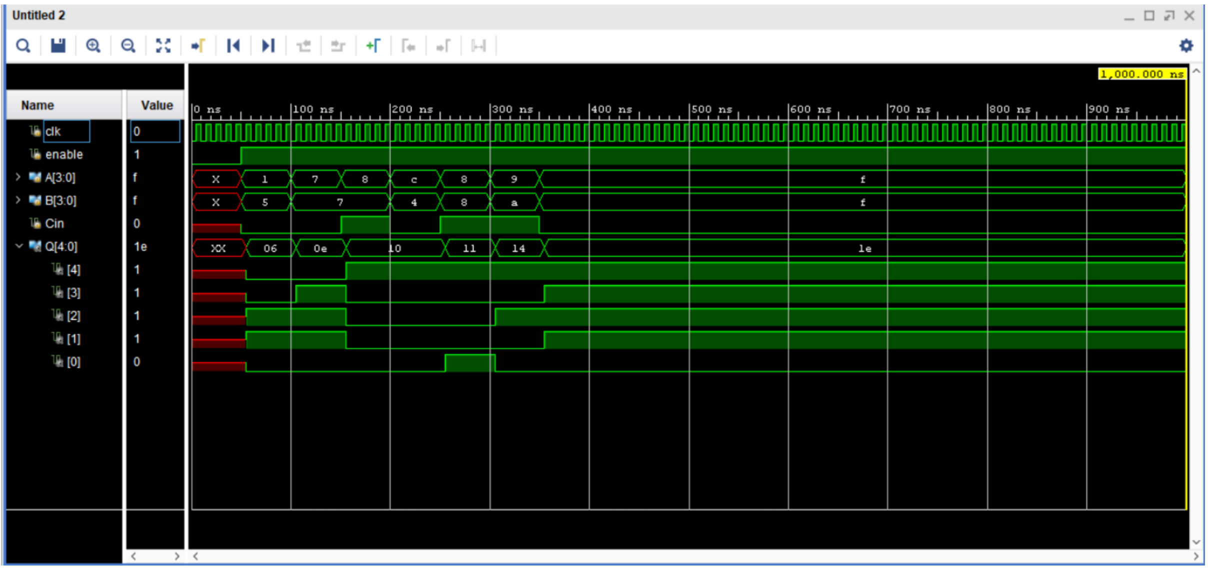
**Table 1.** Testcases for Ripple Carry Adder Verification

iv. Constraints File (Just the uncommented portion)

```
## 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}]
```

v. Simulation waveform for the above test-cases



## Part 2 –

vi. All the equations for $C_i$'s and $S_i$'s

$P_i \rightarrow P_0$

$G_i \rightarrow G_0$

$P_{i+1} \rightarrow P_1$

$C_1 = G_0 + P_0 C_0$ where $g_0 = a_0 b_0$

$P_0 = (a_0 + b_0) \mid (a_0 \oplus b_0)$ *

not equivalent

$C_2 = G_1 + P_1 C_1$

$G_1 + P_1 (G_0 + P_0 C_0)$

$C_2 = g_1 + P_1 G_0 + P_1 P_0 C_0$

$C_3 = g_2 + P_2 C_2$

$= g_2 + P_2 (g_1 + P_1 G_0 + P_1 P_0 C_0)$

$= g_2 + P_2 g_1 + P_2 P_1 G_0 + \underline{P_2 P_1 P_0 C_0}$

$C_4 = g_3 + P_3 g_2 + P_3 P_2 g_1 + P_3 P_2 P_1 g_0 + P_3 P_2 P_1 P_0 C_0$

$C_4 = g_3 + P_3 (g_2 + P_2 g_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0)$

$C_4 = g_3 + P_3 g_2 + P_3 P_2 g_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_0$

$S_1 = P_1 \oplus C_1$

$S_2 = P_2 \oplus C_2$

$S_3 = P_3 \oplus C_3$

$S_4 = P_4 \oplus C_4$

vii. Design files (.v) for the Carry Lookahead Adder and Register Logic

```
`timescale 1ns / 1ps
module CLA_4bits(
   input clk,
   input enable,
   input [3:0] A,
   input [3:0] B,
   input Cin,
```

```
    output [4:0] Q //load registers, should contain the 4 sum bits and Cout
);

wire [3:0] G;//generate
wire [3:0] P;//propagate
wire [4:0] Sum;//sum bits
wire [4:0] C;// carry bits
wire Cout;

assign C[0] = Cin;

assign G[0] = A[0] && B[0];
assign P[0] = A[0] ^ B[0];

assign G[1] = A[1] && B[1];
assign P[1] = A[1] ^ B[1];

assign G[2] = A[2] && B[2];
assign P[2] = A[2] ^ B[2];

assign G[3] = A[3] && B[3];
assign P[3] = A[3] ^ B[3];

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

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

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 d1 (.clk(clk), .enable(enable), .Data(Sum), .Q(Q));
endmodule


`timescale 1ns / 1ps
module register_logic(
   input clk,
   input enable,
   input [4:0] Data,
   output reg [4:0] Q
   );

always @ (posedge clk)
begin //check logic of enable
   if (enable)
      Q = Data;
end
endmodule
```

viii. Test-bench

```
`timescale 1ns / 1ps

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 = 1;

#50
enable = 1;
A = 4'b0000;
B = 4'b0101;
Cin = 1'b0;

#50
enable = 1;
A = 4'b0101;
B = 4'b0111;
Cin = 1'b0;

#50
enable = 1;
A = 4'b1000;
B = 4'b0111;
Cin = 1'b1;

#50
enable = 1;
A = 4'b1001;
```

```
B = 4'b0100;
Cin = 1'b0;

#50
enable = 1;
A = 4'b1000;
B = 4'b1000;
Cin = 1'b1;

#50
enable = 1;
A = 4'b1101;
B = 4'b1010;
Cin = 1'b1;

#50
enable = 1;
A = 4'b1110;
B = 4'b1111;
Cin = 1'b0;

end

always
#5 clk = ~clk;

endmodule
```

ix. Complete Table 2 from the simulation

| A[3:0] | B[3:0] | Cin | Sum[3:0] | Cout |
|---|---|---|---|---|
| 0000 | 0101 | 0 | 5 | 0 |
| 0101 | 0111 | 0 | C | 0 |
| 1000 | 0111 | 1 | 0 | 1 |
| 1001 | 0100 | 0 | D | 0 |
| 1000 | 1000 | 1 | 1 | 1 |
| 1101 | 1010 | 1 | 8 | 1 |
| 1110 | 1111 | 0 | D | 1 |

**Table 2.** Testcases for Carry Lookahead Adder Verification

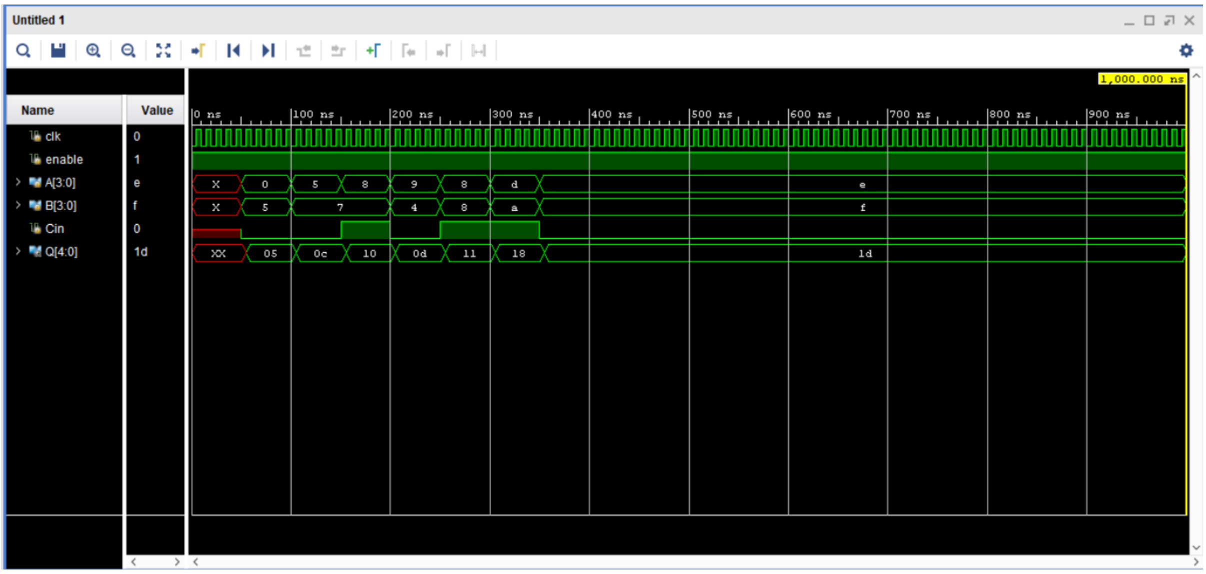
x. Constraints File (Just the uncommented portion)

```
## 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]}]

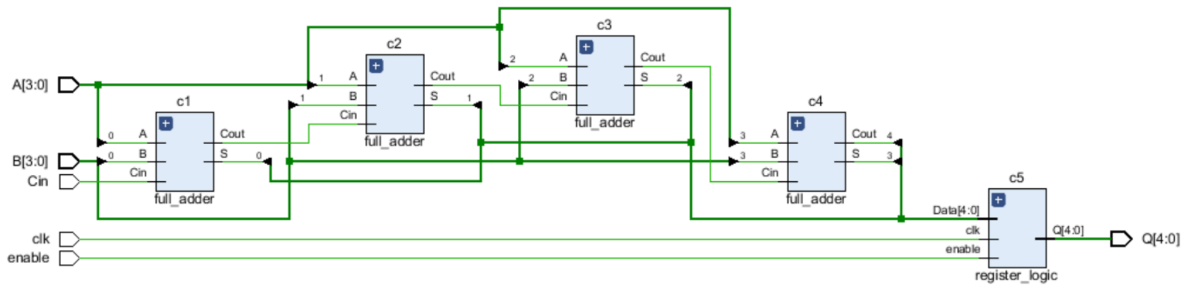
##Buttons
set_property PACKAGE_PIN U18 [get_ports {enable}]
set_property IOSTANDARD LVCMOS33 [get_ports {enable}]

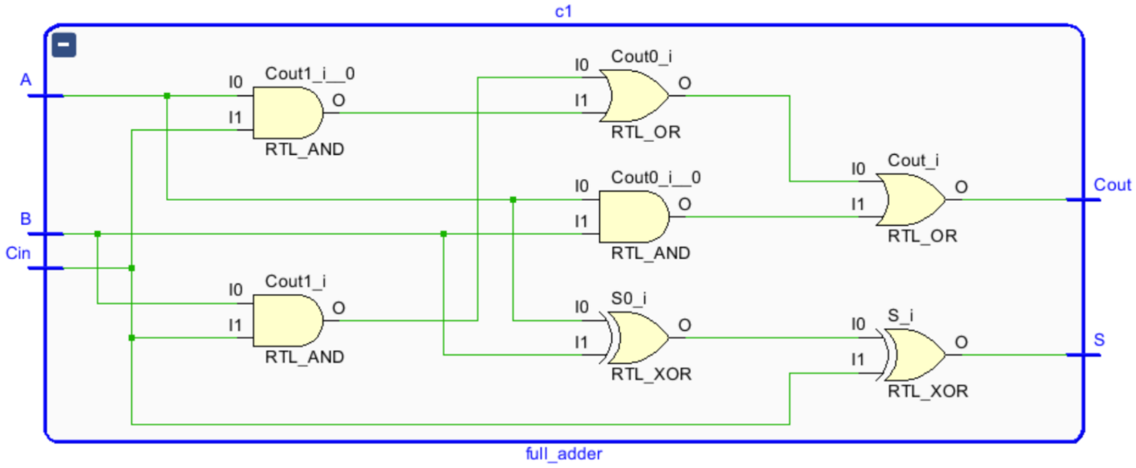
xi. Simulation waveform for the above test-cases

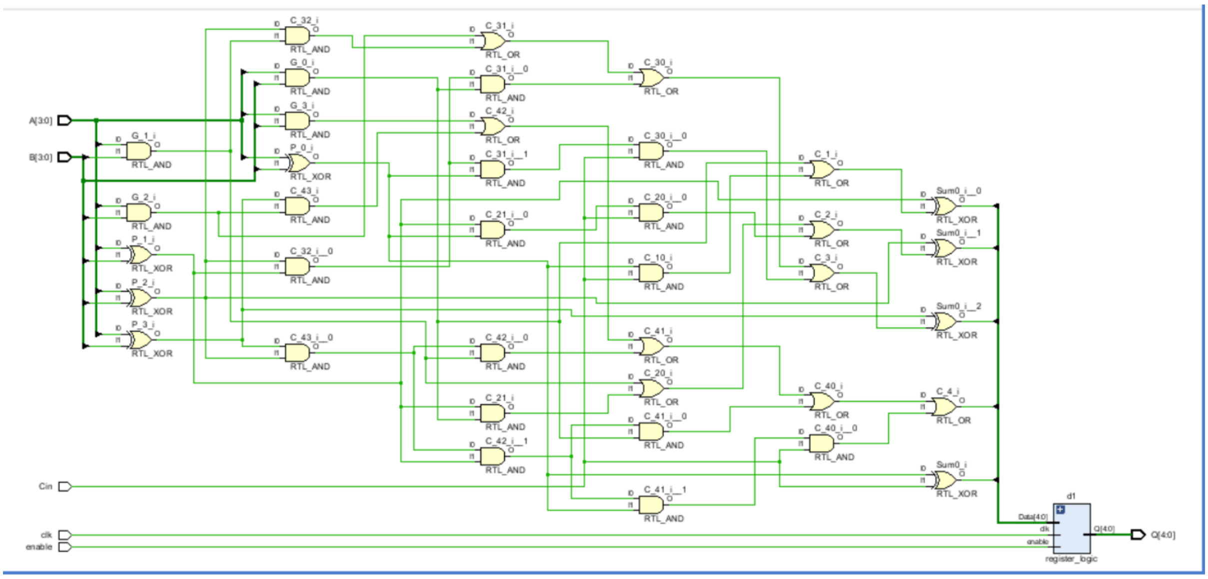



## Part 3 –

xii. Screenshots of the gate-level schematics for both the adder techniques



xiii. Delay and area for both the adder techniques showing all the work

CLA

$D_{critical\ path} = 3 + 3 + 3 + 3 + 3 + 2 = 17ns$

$= 168$

| | |
|---|---|
| AND | $(20 \times 4) = 80$ |
| OR | $(10 \times 4) = 40$ |
| XOR | $(8 \times 6) = 48$ |

$80 + 88 = 168$

$\boxed{D_{crit\ path} = 17ns \quad Area = 168}$

RCA

$D_{crit\ path} = (2+2+3)$

| | |
|---|---|
| AND | $3 \times 4 = 12$ |
| OR | $2 \times 4 = 8$ |
| XOR | $2 \times 6 = 12$ |

20 / 32

4

$32 \times 4 = 128$

$\boxed{D_{crit\ path} = 28ns \quad Area = 128}$

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

RCA is slower than a CLA, but has less area. The CLA is faster than the RCA, but uses more area. A CLA computes the carry before the sum is even generated, while you must wait for the carry to ripple through an RCA.

***Note*** *–> The Verilog codes and the uncommented portions of the constraint files should be copied in your lab report and the* ***actual Verilog (.v), Constraint (.xdc) files and Bitstream (.bit) files*** *need to be zipped and submitted as well on Canvas. You are not allowed to change your codes after final submission as the TAs may download the submitted codes or bitstream files from Canvas during checkouts. For the truth Table, K-maps minimizations and algebraic expressions, you are free to draw them on paper and then put the pictures in your lab report, but please make sure it is legible for the TAs to grade it properly.*