Lab 5 Report

Name: Aryan Agarwal, Sanjay Sivakumar

UT EID: aab5473

Section: MW 11 – 12:30

Checklist:

Part 1 –

i. Design file (.v) for the Ripple Carry Adder
ii. Test-bench
iii. Complete Table 1 from the simulation

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

Table 1. Testcases for Ripple Carry Adder Verification

iv. Constraints File (Just the uncommented portion)
v. Simulation waveform for the above test-cases

Part 2 –

vi. All the equations for C_i's and S_i's
vii. Design files (.v) for the Carry Lookahead Adder and Register Logic
viii. Test-bench
ix. Complete Table 2 from the simulation

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

Table 2. Testcases for Carry Lookahead Adder Verification

x. Constraints File (Just the uncommented portion)
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
xiv. Brief conclusion regarding the pros and cons of each of the techniques

***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.*

Part 1

Verilog:

```
`timescale 1ns / 1ps
module RCA_4bits(
   input clk,
   input enable,
   input [3:0] A,B,
   input Cin,
   output [4:0] Q
   );

   wire [2:0] CarryBit;
   wire [4:0] DataReg;

   full_adder c0(.A(A[0]), .B(B[0]), .Cin(Cin), .S(DataReg[0]), .Cout(CarryBit[0]));
   full_adder c1(.A(A[1]), .B(B[1]), .Cin(CarryBit[0]), .S(DataReg[1]), .Cout(CarryBit[1]));
   full_adder c2(.A(A[2]), .B(B[2]), .Cin(CarryBit[1]), .S(DataReg[2]), .Cout(CarryBit[2]));
   full_adder c3(.A(A[3]), .B(B[3]), .Cin(CarryBit[2]), .S(DataReg[3]), .Cout(DataReg[4]));

   register_logic r(.clk(clk), .enable(enable), .Data(DataReg), .Q(Q));

endmodule
```

Testbench:

```
`timescale 1ns / 1ps

module RCA_tb;

reg clk;
reg enable;
reg [3:0] A, 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 = 1;
Cin = 0;

A = 4'b0001;
B = 4'b0101;

#10;

A = 4'b0111;
B = 4'b0111;

#10

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

#10;
Cin = 0;

A = 4'b1100;
B = 4'b0100;

#10;
Cin = 1;
A = 4'b1000;
B = 4'b1000;

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

#10
Cin = 0;
A = 4'b1111;
B = 4'b1111;

end
```

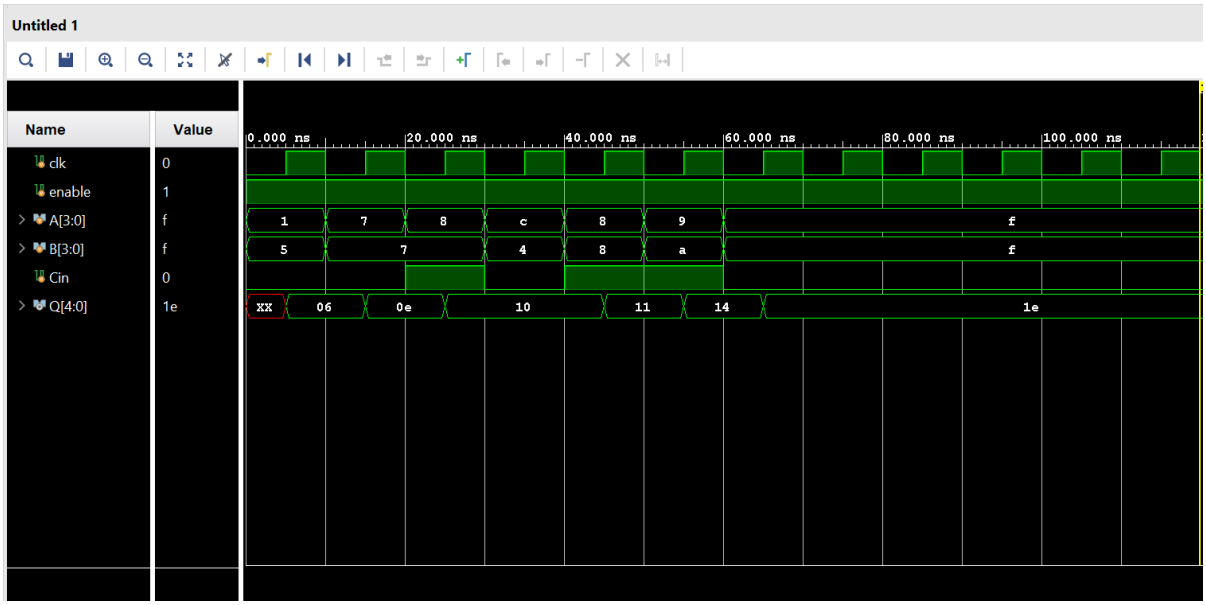
```
always
#5 clk = ~clk;

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

Constraints:

```
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]

#Part 1
set_property PACKAGE_PIN V17 [get_ports {A[0]}]
## Sets the switch to use 3.3V logic
        set_property IOSTANDARD LVCMOS33 [get_ports {A[0]}]

## Connects pin V16 (SW1 on the board) to input b in our gate module
set_property PACKAGE_PIN V16 [get_ports {A[1]}]
## Sets the switch to use 3.3V logic
        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 R2 [get_ports {Cin}]
        set_property IOSTANDARD LVCMOS33 [get_ports {Cin}]
```

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

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]}]
```



Part 2

$G = AB$

$P = A \oplus B$

$C0 = Cin$

$C1 = G0 + P0C0$

$C2 = G1 + P1C1$

$C3 = G2 + P2C2$

$C4 = G3 + P3C3$

$S = A \wedge B \wedge C[3:0]$

$Q[3:0] = S$

$Q[4] = C4$

Verilog:

CLA:

```
`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, S;
    wire [4:0] C;
    wire Cout;
    wire [4:0] DataReg;
```

```
    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] + (P[1]*G[0]) + (P[0]*C[0]*P[1]);
    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 Cout = C[4];

    assign S[0] = P[0] ^ C[0];
    assign S[1] = P[1] ^ (G[0] + (P[0]*C[0]));
    assign S[2] = P[2] ^ (G[1] + (P[1]*G[0]) + (P[0]*C[0]*P[1]));
    assign S[3] = P[3] ^ (G[2] + (P[2]*G[1]) + (P[2]*P[1]*G[0]) + (P[2]*P[1]*P[0]*C[0]));

    assign DataReg[3:0] = S;
    assign DataReg[4] = Cout;

    register_logic r(.clk(clk), .enable(enable), .Data(DataReg), .Q(Q));

endmodule
```

Test Bench:

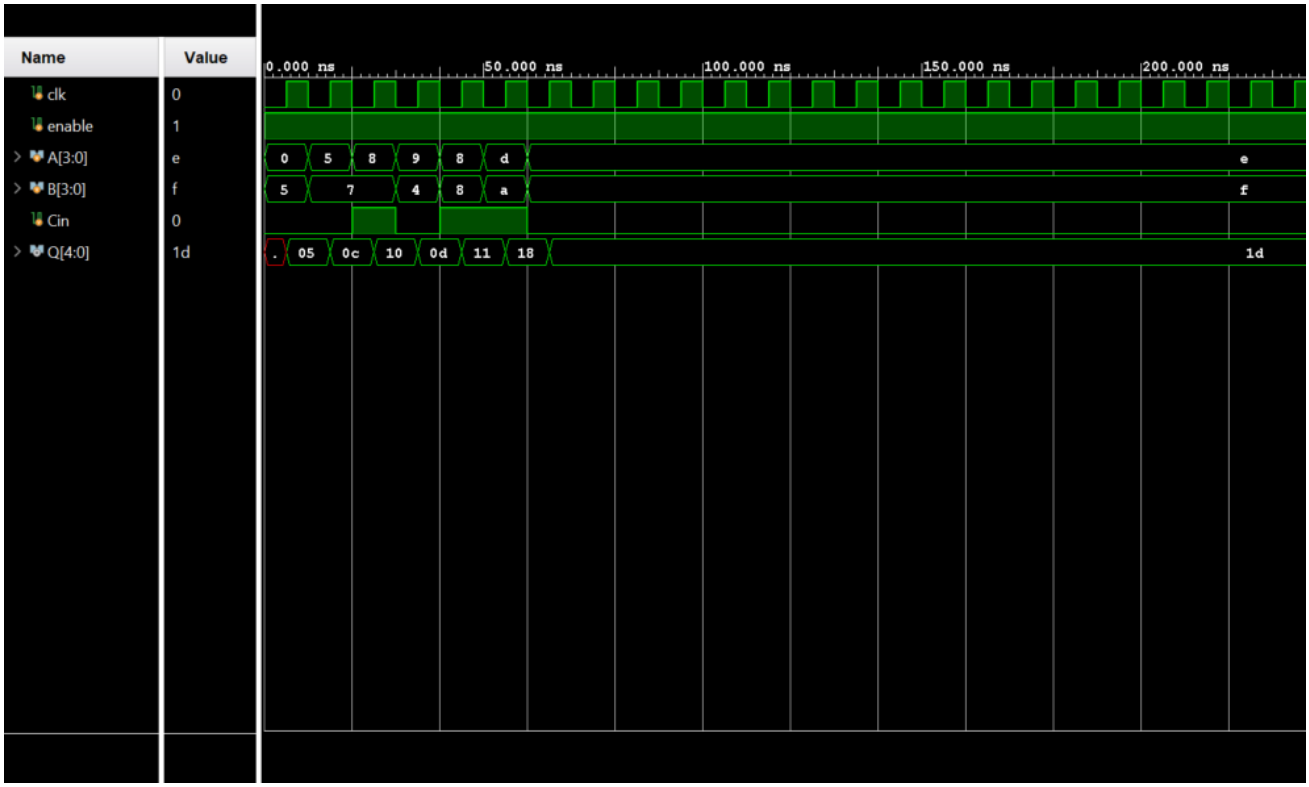
```
`timescale 1ns / 1ps

module CLA_tb;

reg clk;
reg enable;
reg [3:0] A, 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;
Cin = 0;

A = 4'b0000;
B = 4'b0101;

#10;

A = 4'b0101;
B = 4'b0111;

#10

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

#10;
Cin = 0;
```

```
A = 4'b1001;
B = 4'b0100;

#10;
Cin = 1;
A = 4'b1000;
B = 4'b1000;

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

#10
Cin = 0;
A = 4'b1110;
B = 4'b1111;

end

always
#5 clk = ~clk;

Endmodule
```

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

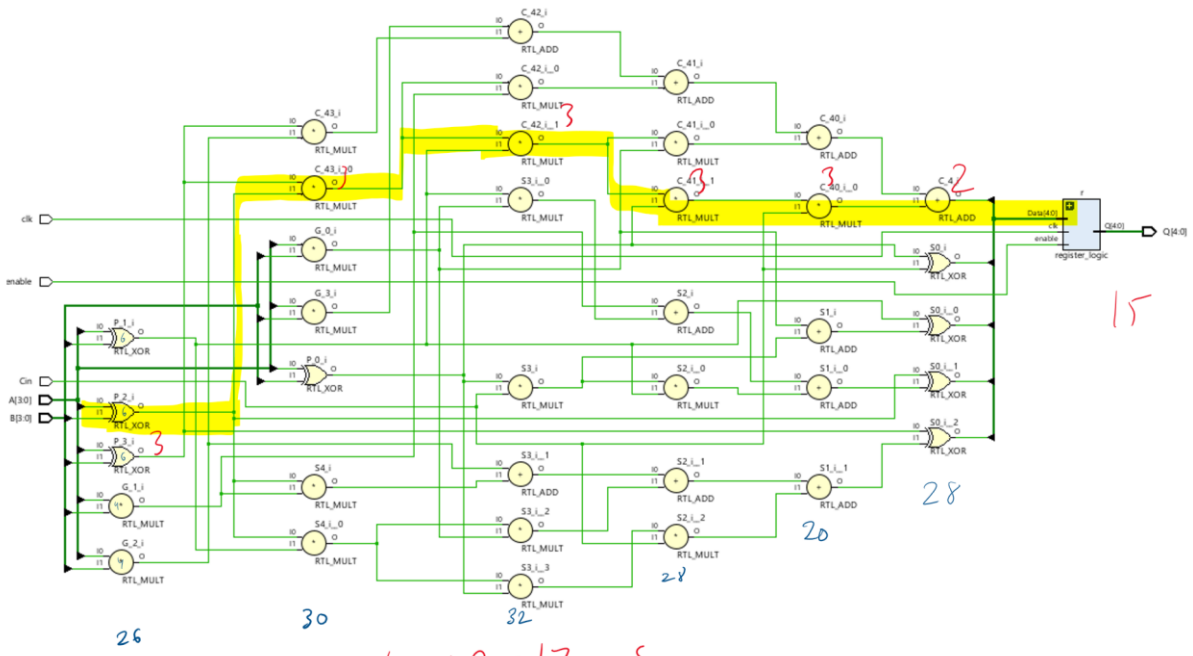
Constraints:

```
set_property PACKAGE_PIN V17 [get_ports {A[0]}]
## Sets the switch to use 3.3V logic
        set_property IOSTANDARD LVCMOS33 [get_ports {A[0]}]

## Connects pin V16 (SW1 on the board) to input b in our gate module
set_property PACKAGE_PIN V16 [get_ports {A[1]}]
## Sets the switch to use 3.3V logic
        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 U18 [get_ports enable]
        set_property IOSTANDARD LVCMOS33 [get_ports enable]

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]}]
```


Name
Value
clk
0
enable
1
A[3:0]
e
B[3:0]
f
Cin
0
Q[4:0]
1d
0.000 ns
50.000 ns
100.000 ns
150.000 ns
200.000 ns
0
5
8
9
8
d
e
5
7
4
8
a
f
.
05
0c
10
0d
11
18
1d

Part 3

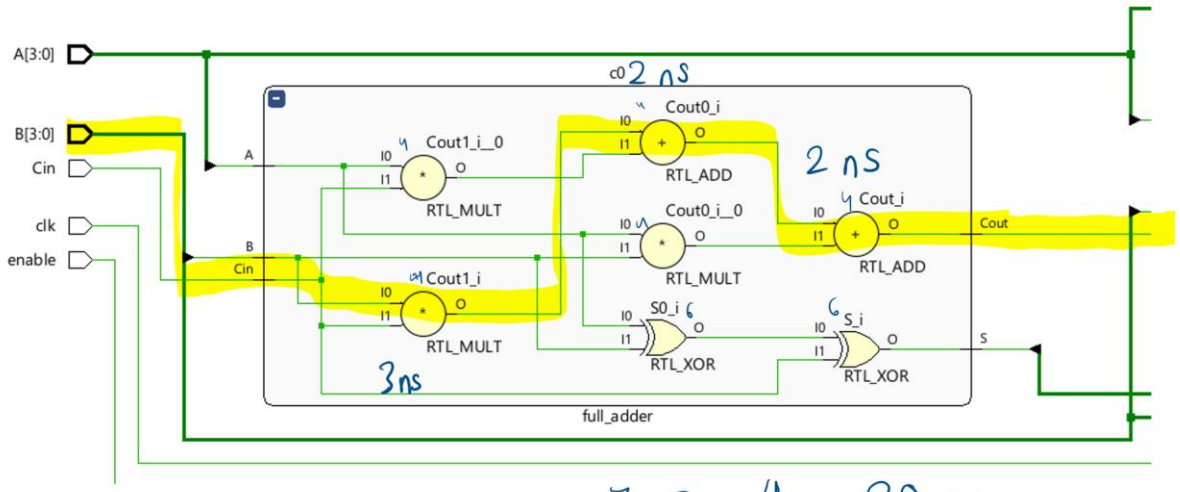
Carry Lookahead Adder:



15 + 2 = 17 ns

26 + 30 + 32 + 28 + 20 + 28 = 164 = Area

Ripple Carry Adder:



7 ns · 4 = 28 ns

4 RCA modules

critical path

12 + 20 = 32

32 · 4 = 128 Area

The CLA utilizes more gates which results in more area. However, because it takes into account all possibilities, it will perform the functions quicker. The RCA utilizes less gates and is more versatile. As such, it is smaller and uses less area. However, the RCA takes more time than the CLA to perform computations. As such, if the implementation needs the actions to be fastest, a CLA would be the best. However, if the implementation requires less space, the RCA would be the best choice.