Lab 5 Report

Name:

UT EID:

Section:

Checklist:

Part 1 –

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

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

  wire s0, s1, s2, s3, c0, c1, c2, c3;
  reg [4:0] total;

Full_Adder f1 (.A(A[0]), .B(B[0]), .Cin(Cin), .S(s0), .Cout(c0));
Full_Adder f2 (.A(A[1]), .B(B[1]), .Cin(c0), .S(s1), .Cout(c1));
Full_Adder f3 (.A(A[2]), .B(B[2]), .Cin(c1), .S(s2), .Cout(c2));
Full_Adder f4 (.A(A[3]), .B(B[3]), .Cin(c2), .S(s3), .Cout(c3));

always @(*) begin total = {c3, s3, s2, s1, s0}; end

register_logic f5 (.clk(clk), .enable(enable), .Data(total), .Q(Q));
endmodule
```

ii. Test-bench

```
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'b0001;
B = 4'b0101;
Cin = 1'b0;

#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
```

iii. Complete Table 1 from the simulation

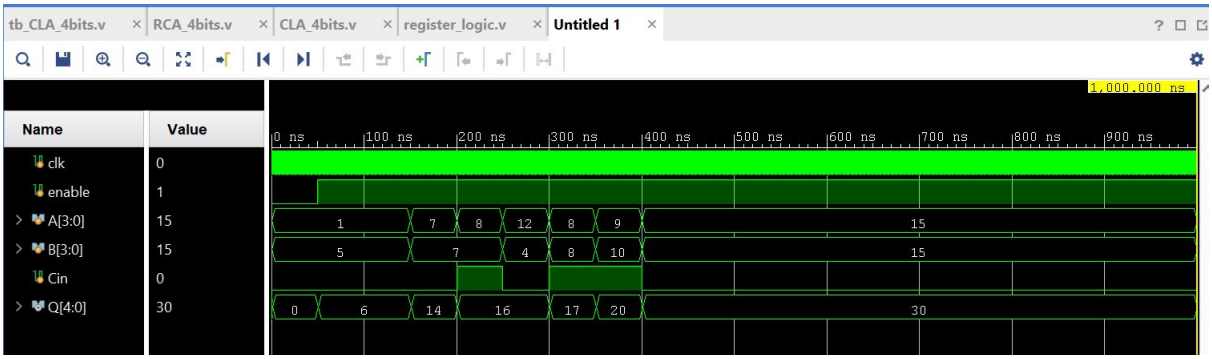
A[3:0]	B[3:0]	Cin	Sum[3:0]	Cout
0001	0101	0	0110	0
0111	0111	0	1110	0
1000	0111	1	0000	1
1100	0100	0	0000	1
1000	1000	1	0001	1
1001	1010	1	0100	1
1111	1111	0	1110	1

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]}]
```

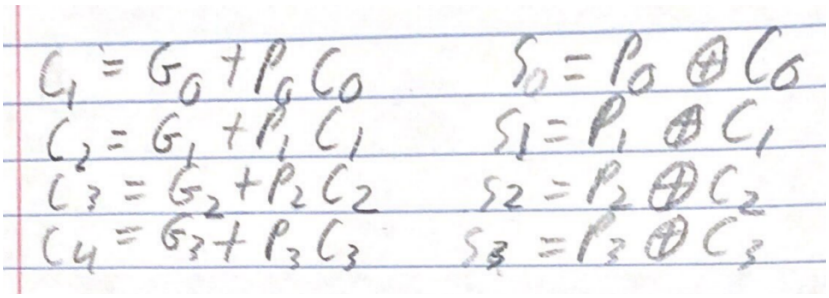
```
set_property PACKAGE_PIN V2 [get_ports {Cin}]

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



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

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

   wire [3:0] G, P, S, Sum;
   wire [4:0] C;
   wire Cout;
   assign C[0]=Cin;
//generate
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];
//propagate
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];
//carry bits
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]);
//sum results
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, enable,
   input [4:0] Data,
   output reg [4:0] Q
   );

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

endmodule
```

viii. Test-bench

```
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'b1110;
B = 4'b1111;
Cin = 1'b0;

#50;

end

always
#1 clk = ~clk;

endmodule
```

ix. Complete Table 2 from the simulation

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	1
1000	1000	1	0001	1
1101	1010	1	1000	1
1110	1111	0	1110	1

x. Constraints File (Just the uncommented portion)
xi. ## Clock signal
xii. set_property PACKAGE_PIN W5 [get_ports clk]

xiii. set_property IOSTANDARD LVCMOS33 [get_ports clk]
xiv. create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]
xv.
xvi. ## Switches
xvii. set_property PACKAGE_PIN V17 [get_ports {A[0]}]

xviii. set_property IOSTANDARD LVCMOS33 [get_ports {A[0]}]
xix. set_property PACKAGE_PIN V16 [get_ports {A[1]}]

xx. set_property IOSTANDARD LVCMOS33 [get_ports {A[1]}]
xxi. set_property PACKAGE_PIN W16 [get_ports {A[2]}]

xxii. set_property IOSTANDARD LVCMOS33 [get_ports {A[2]}]
xxiii. set_property PACKAGE_PIN W17 [get_ports {A[3]}]

xxiv. set_property IOSTANDARD LVCMOS33 [get_ports {A[3]}]
xxv. set_property PACKAGE_PIN W15 [get_ports {B[0]}]

xxvi. set_property IOSTANDARD LVCMOS33 [get_ports {B[0]}]
xxvii. set_property PACKAGE_PIN V15 [get_ports {B[1]}]

xxviii. set_property IOSTANDARD LVCMOS33 [get_ports {B[1]}]
xxix. set_property PACKAGE_PIN W14 [get_ports {B[2]}]

xxx. set_property IOSTANDARD LVCMOS33 [get_ports {B[2]}]
xxxi. set_property PACKAGE_PIN W13 [get_ports {B[3]}]

xxxii. set_property IOSTANDARD LVCMOS33 [get_ports {B[3]}]
xxxiii. set_property PACKAGE_PIN V2 [get_ports {Cin}]

xxxiv. set_property IOSTANDARD LVCMOS33 [get_ports {Cin}]
xxxv. ## LEDs
xxxvi. set_property PACKAGE_PIN U16 [get_ports {Q[0]}]

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

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

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

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

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

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

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

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

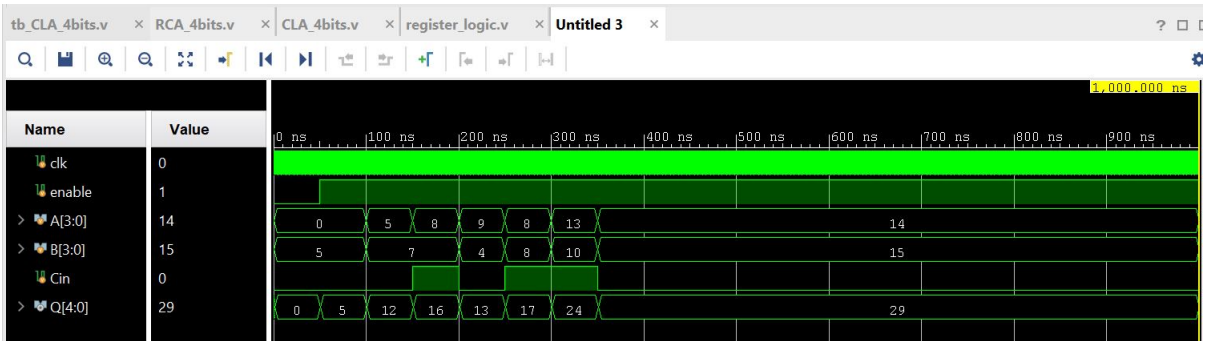
xlv. set_property IOSTANDARD LVCMOS33 [get_ports {Q[4]}]

xlvi. ##Buttons

xlvii. set_property PACKAGE_PIN U18 [get_ports {enable}]

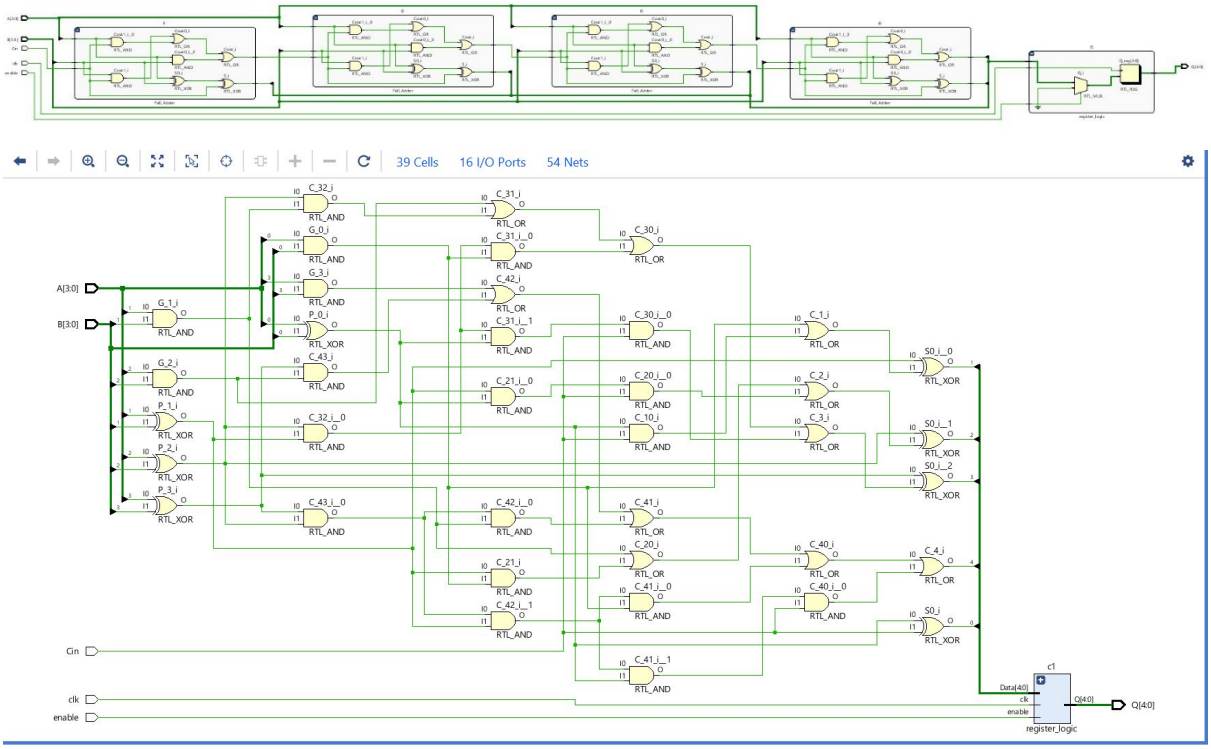
xlviii. set_property IOSTANDARD LVCMOS33 [get_ports {enable}]

xlix. Simulation waveform for the above test-cases

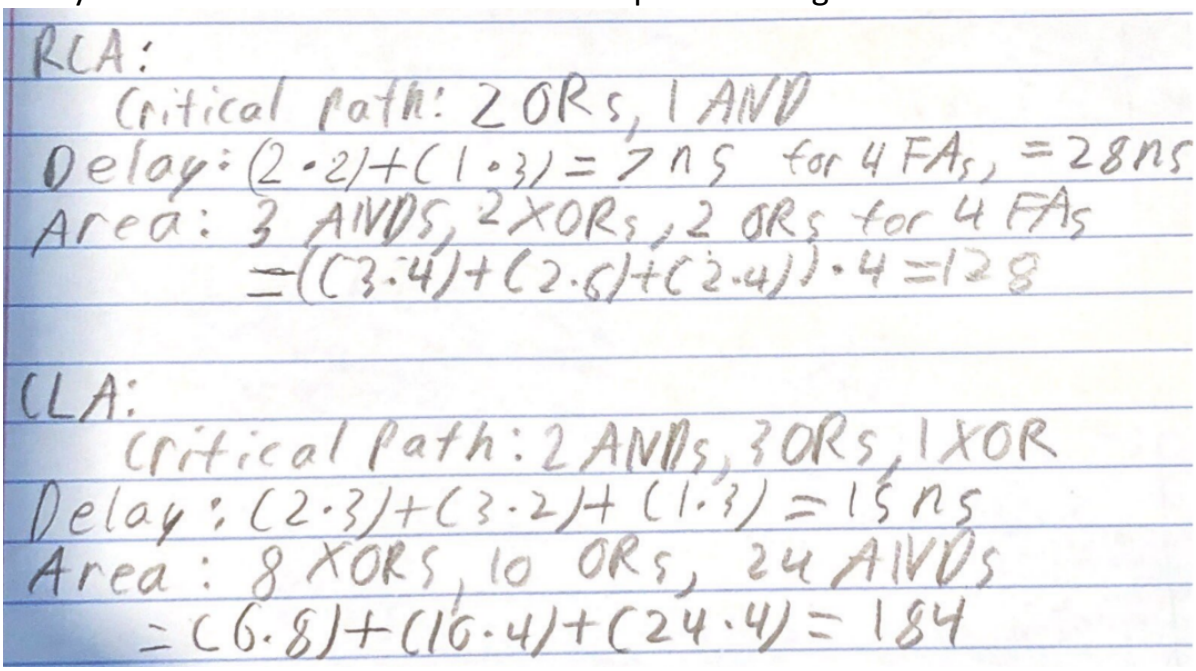


Part 3 –

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



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



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

The Carry Lookahead adder is faster than the Ripple Carry adder, but at the cost of taking up more space. Furthermore, the Carry Lookahead adder will become more efficient the more inputs are added, as opposed to the Ripple Carry adder. However, the gate logic for the CLA is much more complicated than the 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.*