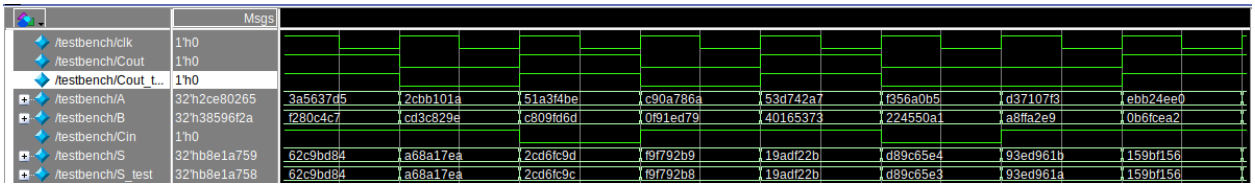
Lab 1 - FPGA Design Basics

Name: Jason Xie
NSID: xix277
Student Number: 11255431

Ripple Carry Adder

Simulation



As can be seen from the simulation, the Cout_test and Cout match, as well as the S and S_test outputs.

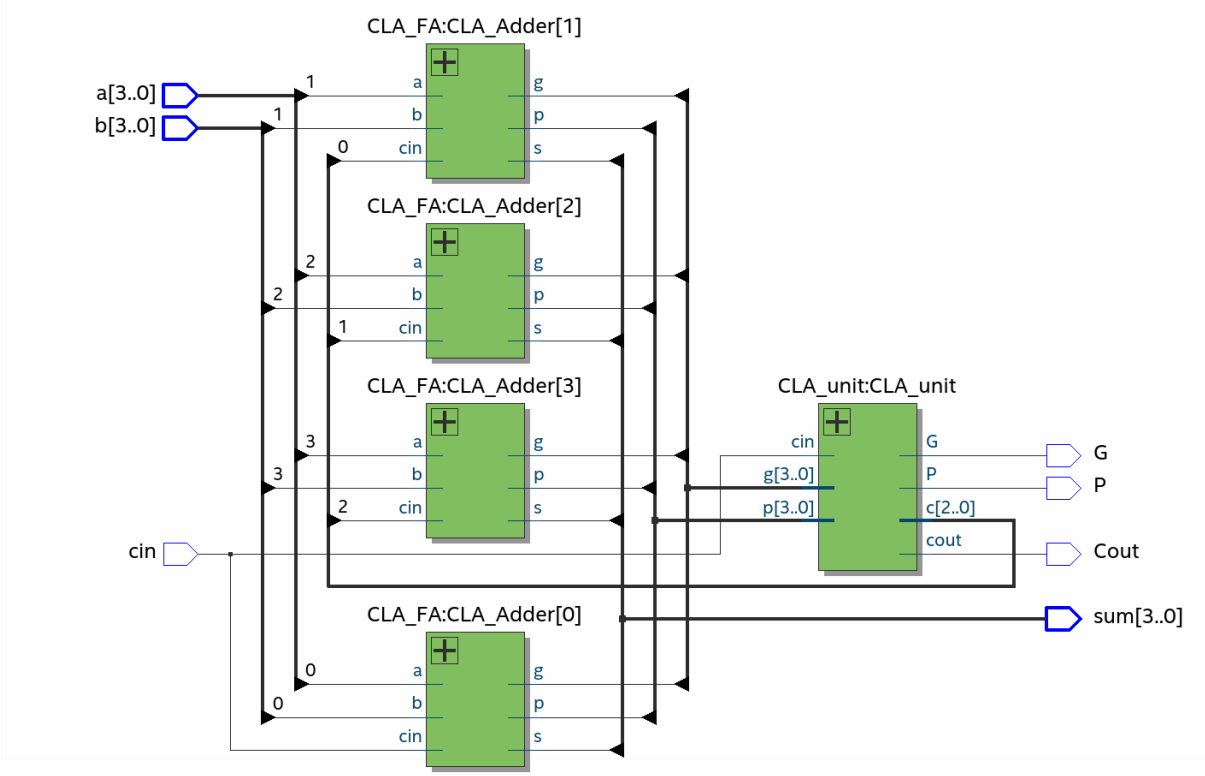
Hardware Resource utilization

	Resource	Usage
1	Estimated Total logic elements	148
2		
3	Total combinational functions	83
4	▼ Logic element usage by number of LUT inputs	
1	-- 4 input functions	36
2	-- 3 input functions	20
3	-- <=2 input functions	27
5		
6	▼ Logic elements by mode	
1	-- normal mode	83
2	-- arithmetic mode	0
7		
8	▼ Total registers	98
1	-- Dedicated logic registers	98
2	-- I/O registers	0
9		
10	I/O pins	99
11		
12	Embedded Multiplier 9-bit elements	0
13		
14	Maximum fan-out node	clk~input
15	Maximum fan-out	98
16	Total fan-out	586
17	Average fan-out	1.55

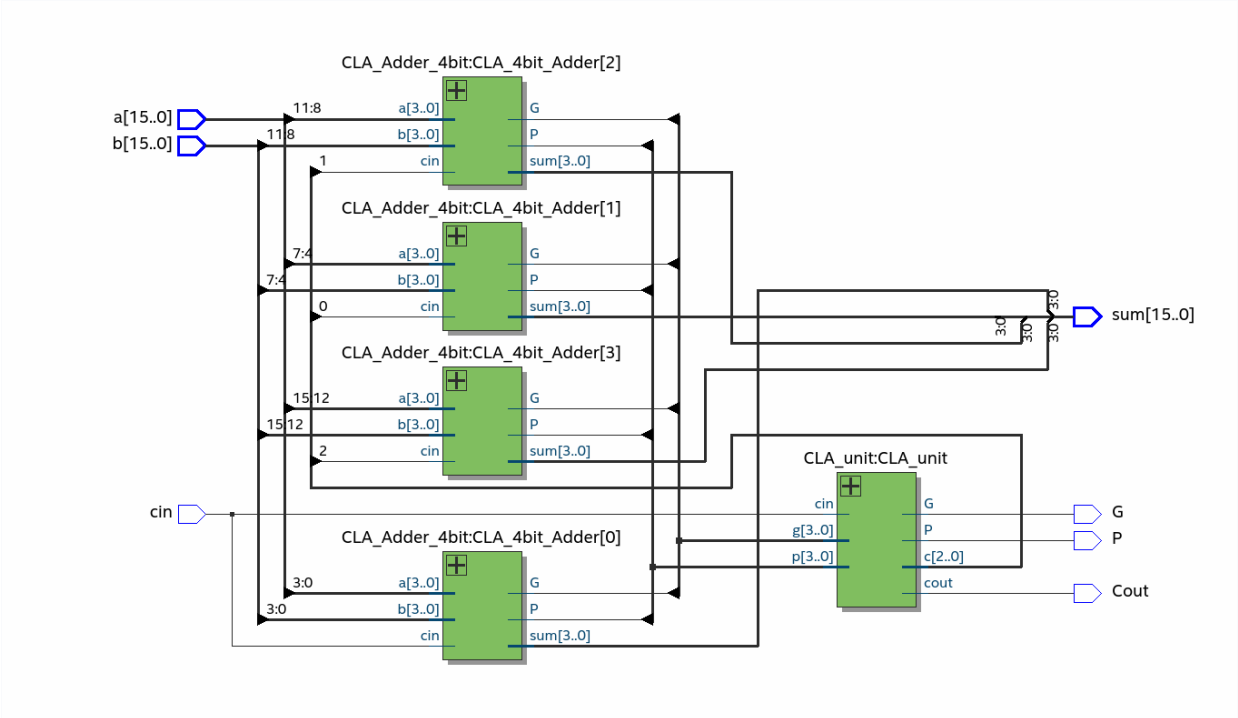
At 85C the max frequency is 99.81 MHz and at 0C the max frequency is 107.52 MHz. Therefore the max frequency will fall within these ranges from 0C-85C.

Carry-Lookahead Adder

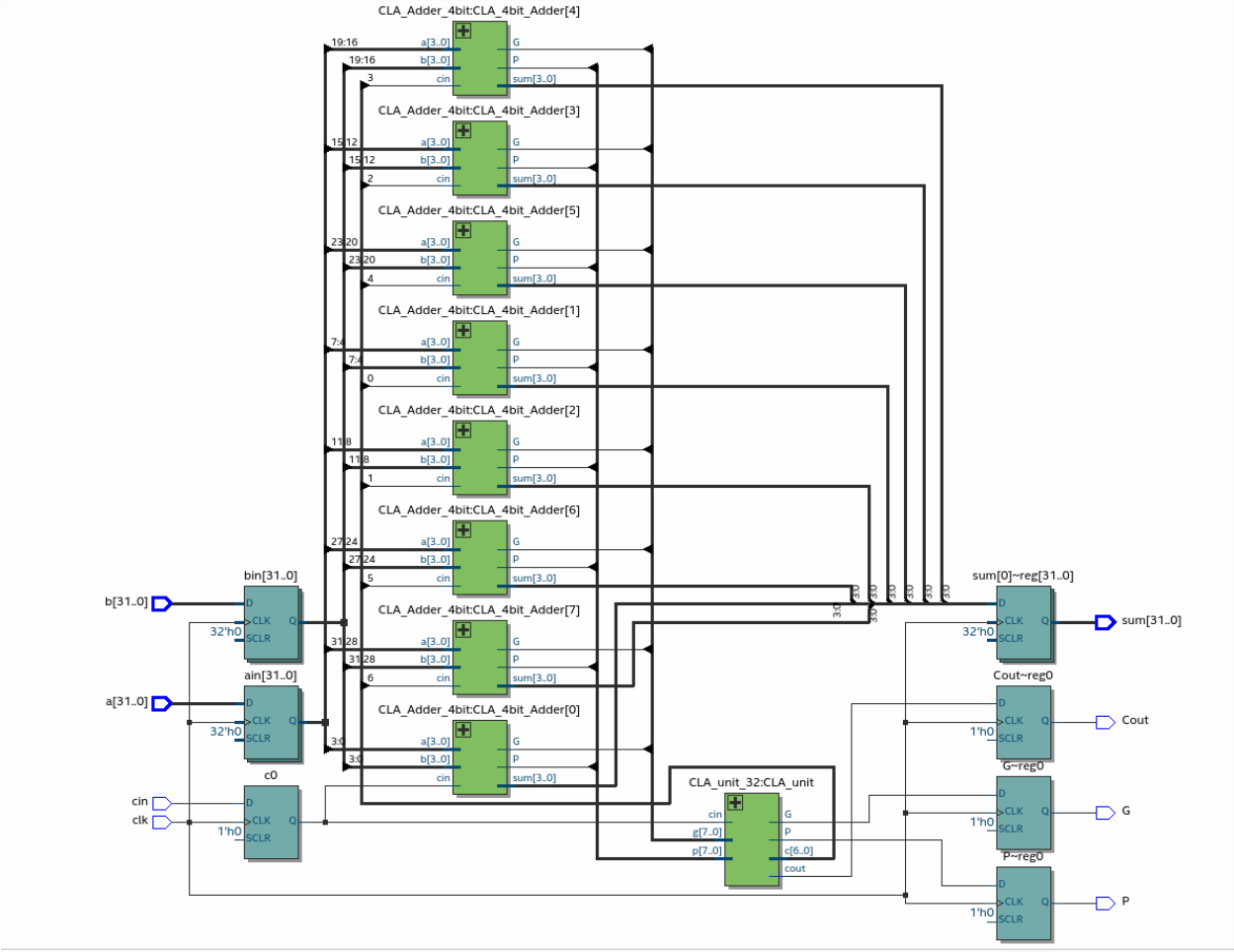
4-bit CLA



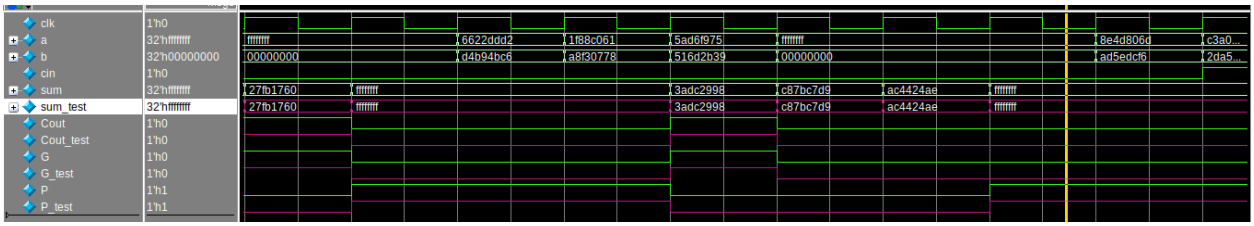
16-bit CLA



32-bit CLA



Simulation



As can be seen from the simulation, the 32 bit CLA matches the test results.

Hardware Resource utilization

	Resource	Usage
1	Estimated Total logic elements	193
2		
3	Total combinational functions	128
4	▾ Logic element usage by number of LUT inputs	
1	-- 4 input functions	79
2	-- 3 input functions	33
3	-- <=2 input functions	16
5		
6	▾ Logic elements by mode	
1	-- normal mode	128
2	-- arithmetic mode	0
7		
8	▾ Total registers	100
1	-- Dedicated logic registers	100
2	-- I/O registers	0
9		
10	I/O pins	101
11		
12	Embedded Multiplier 9-bit elements	0
13		
14	Maximum fan-out node	clk~input
15	Maximum fan-out	100
16	Total fan-out	783
17	Average fan-out	1.82

At 85C the max frequency is 163.93 MHz and at 0C the max frequency is 182.48 MHz. Therefore the max frequency will fall within these ranges from 0C-85C.

Comparison

Comparing the two, the ripple carry adder requires less resources whereas the carry lookahead adder (CLA) requires more. This means that the ripple carry adder requires less area. However, in terms of speed the CLA is faster. So depending on the requirements of design, if you need a smaller area and speed does not matter, choose the ripple carry adder. If the design requires speed and area does not matter, then choose the CLA.

Appendix

ripple_carry_adder.v

```
module ripple_carry_adder(
input wire clk, Cin,
input wire [31:0] A, B,
output reg [31:0] S,
output reg Cout);

reg C0;
wire C32;
reg [31:0] Ain, Bin;
wire [31:0] Sout;
wire [30:0] C;

always @ (posedge clk)
S <= Sout;

always @ (posedge clk)
Cout <= C32;
always @ (posedge clk)
C0 <= Cin;
always @ (posedge clk)
Ain <= A;
always @ (posedge clk)
Bin <= B;


full_adder rpp_crry [31:0] (.A(Ain), .B(Bin), .Cin({C, C0}),
.Cout({C32,C}), .S(Sout));

endmodule
```

full_adder.v

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

always @ *
S = (A ^ B) ^ Cin;

always @ *
Cout = (A & B) | (Cin & (A ^ B));
endmodule
```

tesbench_ripple.sv

```
`timescale 1us/1ns

module testbench_ripple();

reg clk;
wire Cout;
reg [31:0] A, B;
reg Cin;
wire [31:0] S;


reg Cout_test;
reg [32:0] S_inter;
reg [31:0] S_test;
reg [31:0] prev_A, prev_B;
reg prev_Cin;
initial #640 $stop;

initial clk = 1'b1;
always #0.5 clk = ~clk;

initial begin
Cin <= 0;
A <= 0;
B <= 0;
end

always @ (posedge clk)
begin
Cin <= $urandom();
A <= $urandom();
B <= $urandom();
end

ripple_carry_adder test(
.clk(clk),
.Cin(Cin),
.A(A),
.B(B),
```

```
.S(S),
.Cout(Cout)
);

always @ (posedge clk)
begin
prev_A <= A;
prev_B <= B;
prev_Cin <= Cin;
end


always @ (posedge clk)
S_inter = prev_A + prev_B;

always @ (posedge clk)
S_test = S_inter[31:0];

always @ (posedge clk)
Cout_test = S_inter[32];
endmodule
```

CLA_unit_32.v

```
module CLA_unit_32(
input wire [7:0] p, g,
input wire cin,
output wire [6:0] c,
output wire cout,
output wire G, P
);

// P logic
assign P = p[0] & p[1] & p[2] & p[3] & p[4] & p[5] & p[6] & p[7];

// G logic
assign G = g[7] |
(g[6] & p[7]) |
(g[5] & p[7] & p[6]) |
(g[4] & p[7] & p[6] & p[5]) |
(g[3] & p[7] & p[6] & p[5] & p[4]) |
(g[2] & p[7] & p[6] & p[5] & p[4] & p[3]) |
(g[1] & p[7] & p[6] & p[5] & p[4] & p[3] & p[2]) |
(g[0] & p[7] & p[6] & p[5] & p[4] & p[3] & p[2] & p[1]);

// cout logic
// c1
assign c[0] = g[0] | (p[0] & cin);
// c2
assign c[1] = g[1] | (p[1] & c[0]);
// c3
assign c[2] = g[2] | (p[2] & c[1]);
// c4
assign c[3] = g[3] | (p[3] & c[2]);
// c5
assign c[4] = g[4] | (p[4] & c[3]);
// c6
assign c[5] = g[5] | (p[5] & c[4]);
// c7
assign c[6] = g[6] | (p[6] & c[5]);
// cout
assign cout = g[7] | (p[7] & c[6]);
endmodule
```

CLA_Adder_32bit.v

```
module CLA_Adder_32bit(
input wire cin, clk,
input [31:0] a, b,
output reg G, P, Cout,
output reg [31:0] sum
);
reg [31:0] ain, bin;
reg c0;
wire cout, Gout, Pout;
wire [31:0] sumout;
wire [7:0] p, g;
wire [6:0] c;
// outputs
always @ (posedge clk)
G <= Gout;
always @ (posedge clk)
P <= Pout;

always @ (posedge clk)
Cout <= cout;
always @ (posedge clk)
sum <= sumout;

// inputs
always @ (posedge clk)
c0 <= cin;

always @ (posedge clk)
ain <= a;

always @ (posedge clk)
bin <= b;

CLA_unit_32 CLA_unit (.cin(c0), .p(p), .g(g), .cout(cout), .P(Pout),
.G(Gout), .c(c));

CLA_Adder_4bit CLA_4bit_Adder [7:0] ( .cin({c,c0}), .P(p), .G(g),
.sum(sumout), .a(ain), .b(bin));
endmodule
```

testbench_CLA.sv

```
`timescale 1us/1ns

module testbench_CLA();

reg clk;
wire Cout, G, P;
reg [31:0] a, b;
reg cin;
wire [31:0] sum;

// test logic
reg Cout_test, G_test, P_test, prev_p;
reg [31:0] temp_P, temp_G;
reg [32:0] sum_inter;
reg [31:0] sum_test;
reg [31:0] prev_a, prev_b;
reg prev_cin;
// initial #640 $stop;

initial clk = 1'b1;
always #0.5 clk = ~clk;

initial begin
cin <= 0;
a <= 0;
b <= 0;
repeat (100)begin
repeat(3)
begin
// repeat(3) @(posedge clk)
@(posedge clk)
cin <= $urandom;
a <= $urandom;
b <= $urandom;
// repeat(3) @(posedge clk)
// cin <= 0;
// a <= ~0;
// b <= 0;
end
```

```
repeat(3) begin
@(posedge clk)
cin <= 0;
a <= ~0;
b <= 0;
end
end

$finish;


end



always @ (posedge clk)
begin
prev_a <= a;
prev_b <= b;
prev_cin <= cin;
end


always @ (posedge clk)
sum_inter <= a + b;

always @ (posedge clk)
sum_test <= prev_a + prev_b;//sum_inter[31:0];
// always @ (posedge clk)
// sum_test = sum_inter[31:0];

always @ (posedge clk)
Cout_test = sum_inter[32];




always @ (posedge clk)
begin
temp_P = prev_a | prev_b;
```

```
P_test = &(temp_P);
temp_G = prev_a & prev_b;
G_test = temp_G[31];
prev_p = temp_P[31];
for(int i = 30;i >= 0; i--)
begin
G_test |= prev_p & (temp_G[i]);
prev_p &= temp_P[i];
end
end


CLA_Adder_32bit test(
.clk(clk),
.cin(cin),
.a(a),
.b(b),
.G(G),
.P(P),
.Cout(Cout),
.sum(sum)
);


endmodule
```