EHB436E

DIGITAL SYSTEM DESIGN APPLICATIONS

Muhammed Erkmen
040170049

EXPERIMENT-4 REPORT

Half Adder

Source Code of half adder:

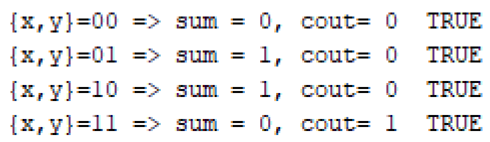
```
`timescale 1ns / 1ps
module arithmetic_circuits(
input x,
input y,
output cout,
output sum
    );

HA HA1 (.x(x),.y(y),.sum(sum),.cout(cout));

endmodule



module HA (
input x,
input y,
output cout,
output sum
);
assign sum = ((~x)&&y) || (x&&(~y));
assign cout = x&&y;
endmodule
```

Half adder adds 2 bit and outs sum and cout.

Testbench of half adder:

```
`timescale 1ns / 1ps

module arithmetic_circuits_tb();

reg x;
reg y;
wire sum;
wire cout;
parameter wait_time = 20;
integer i;
// true results

reg [1:0] correct_results_sum[3:0];
reg [1:0] correct_results_cout[3:0];

initial begin
correct_results_sum[0] =0;
correct_results_cout[0] =0 ;
correct_results_sum[1] =1;
correct_results_cout[1] =0 ;
correct_results_sum[2] =1;
correct_results_cout[2] = 0;
correct_results_sum[3] =0;
correct_results_cout[3] = 1;
end

arithmetic_circuits L1 (.x(x),.y(y),.sum(sum),.cout(cout));

initial begin
for (i=0;i<4;i=i+1)
begin
{x,y} = i;
#wait_time;
$write ("{x,y}=%d%d => sum = %d, cout= %d",x,y,sum,cout);
if(sum== correct_results_sum[i] && cout == correct_results_cout[i])
$display("  TRUE");
else
$display("  FALSE");
end
end
endmodule
```

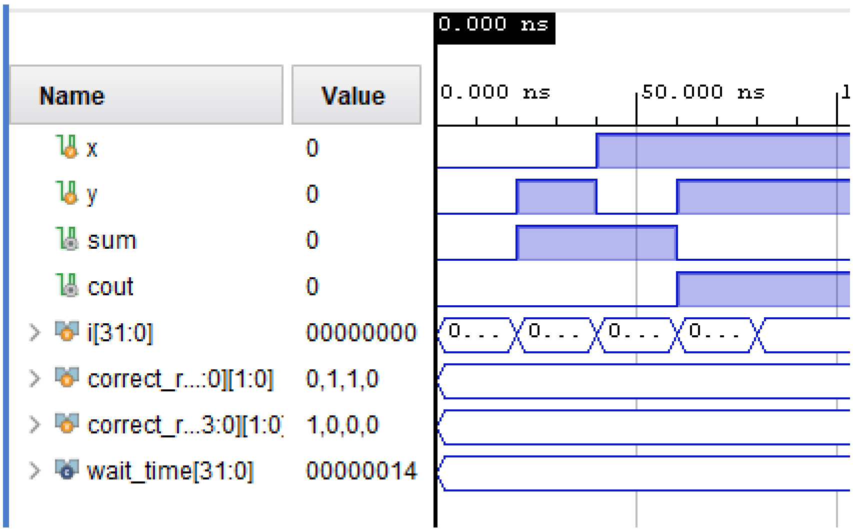
Testbench TCL console output:



```
{x,y}=00 => sum = 0, cout= 0  TRUE
{x,y}=01 => sum = 1, cout= 0  TRUE
{x,y}=10 => sum = 1, cout= 0  TRUE
{x,y}=11 => sum = 0, cout= 1  TRUE
```

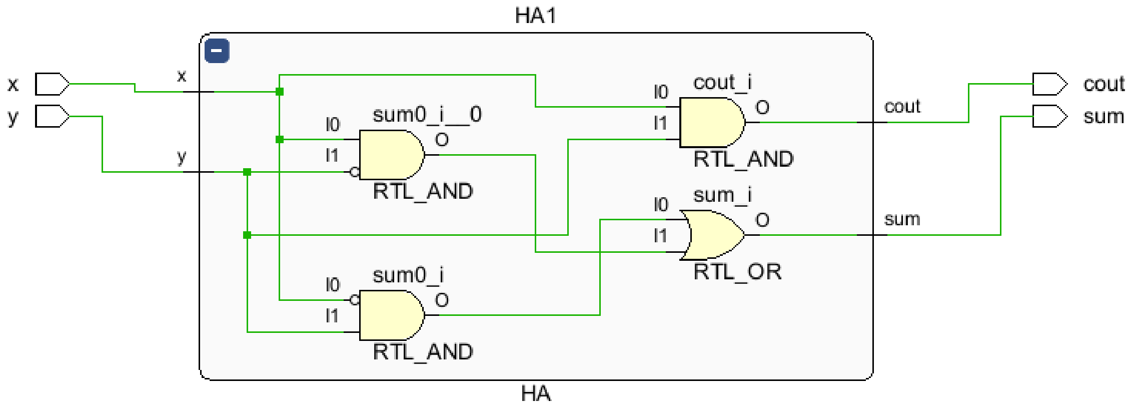
I coded true results in testbench, so the outputs show me that everything is okay.

Behavioral simulation of half adder

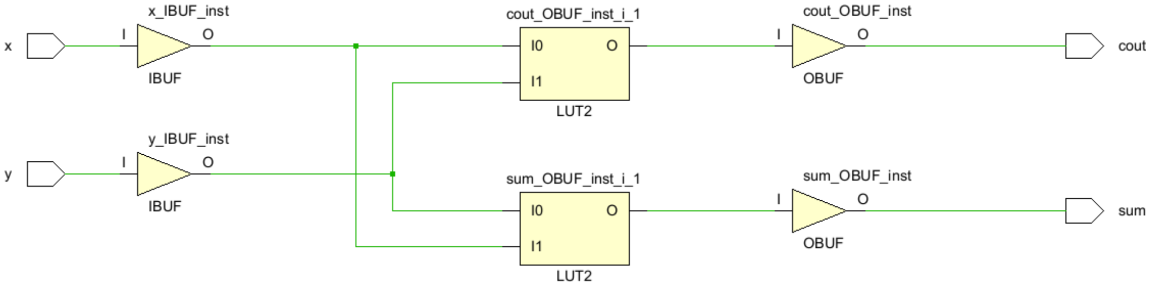


Name	Value
x	0
y	0
sum	0
cout	0
i[31:0]	00000000
correct_r...:0][1:0]	0,1,1,0
correct_r...3:0][1:0]	1,0,0,0
wait_time[31:0]	00000014

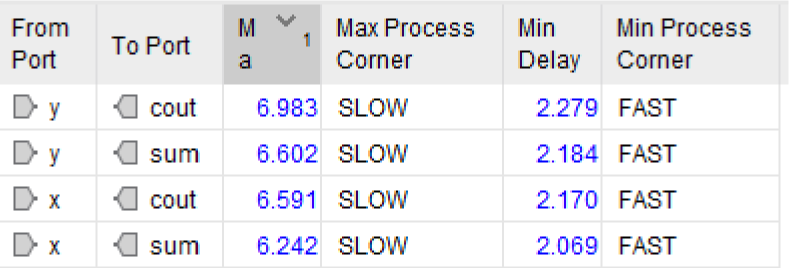
RTL Schematic of half adder



Technology Schematic of half adder

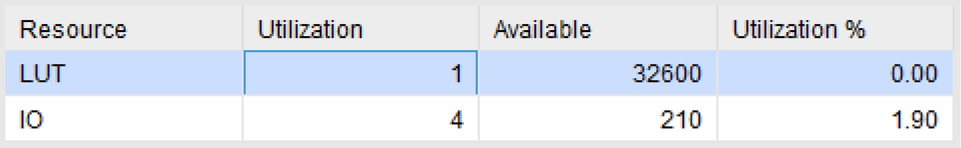


Timing Summary of half adder

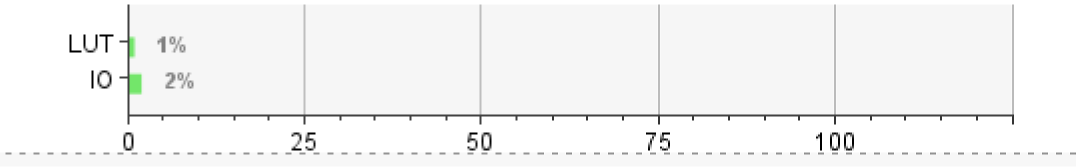


From Port	To Port	Ma	Max Process Corner	Min Delay	Min Process Corner
y	cout	6.983	SLOW	2.279	FAST
y	sum	6.602	SLOW	2.184	FAST
x	cout	6.591	SLOW	2.170	FAST
x	sum	6.242	SLOW	2.069	FAST

Utilization Report of half adder



Resource	Utilization	Available	Utilization %
LUT	1	32600	0.00
IO	4	210	1.90



Only 1 LUT is used in half adder application.

FULL ADDER

Source Code of full adder:

```
`timescale 1ns / 1ps
module arithmetic_circuits(
input x,
input y,
input cin,
output cout,
output sum
    );
FA FA1 (.x(x),.y(y),.cin(cin),.sum(sum),.cout(cout));
endmodule


module HA (
input x,
input y,
output cout,
output sum
);
assign sum = ((~x)&&y) || (x&&(~y));
assign cout = x&&y;
endmodule

module FA (
input x,
input y,
input cin,
output cout,
output sum);
wire sum1;
wire cout1;
wire sum2;
wire cout2;

HA HA1 (.x(x),.y(y),.sum(sum1),.cout(cout1));
HA HA2 (.x(sum1),.y(cin),.sum(sum2),.cout(cout2));

assign sum = sum2;
assign cout = cout2 || cout1;

endmodule
```

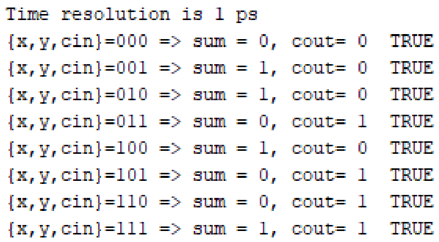
Full adder adds 2 1-bit number and a carry in.

Testbench of full adder:

```
`timescale 1ns / 1ps
module arithmetic_circuits_tb();
reg x;
reg y;
reg cin;
wire sum;
wire cout;
parameter wait_time = 20;
integer i;
// true results
reg [1:0] correct_results_sum[7:0];
reg [1:0] correct_results_cout[7:0];
initial begin
correct_results_sum[0] =0;
correct_results_cout[0] =0 ;
correct_results_sum[1] =1;
correct_results_cout[1] =0 ;
correct_results_sum[2] =1;
correct_results_cout[2] = 0;
correct_results_sum[3] =0;
correct_results_cout[3] = 1;
//
correct_results_sum[4] =1;
correct_results_cout[4] =0 ;
correct_results_sum[5] =0;
correct_results_cout[5] =1 ;
correct_results_sum[6] =0;
correct_results_cout[6] = 1;
correct_results_sum[7] =1;
correct_results_cout[7] = 1;
end
arithmetic_circuits L1 (.x(x),.y(y),.cin(cin),.sum(sum),.cout(cout));
initial begin
for (i=0;i<8;i=i+1)
begin
{x,y,cin} = i;
#wait_time;
$write ("{x,y,cin}=%d%d%d => sum = %d, cout= %d",x,y,cin,sum,cout);
if(sum== correct_results_sum[i] && cout == correct_results_cout[i])
$display("  TRUE");
else
$display("  FALSE");
end

end
endmodule
```

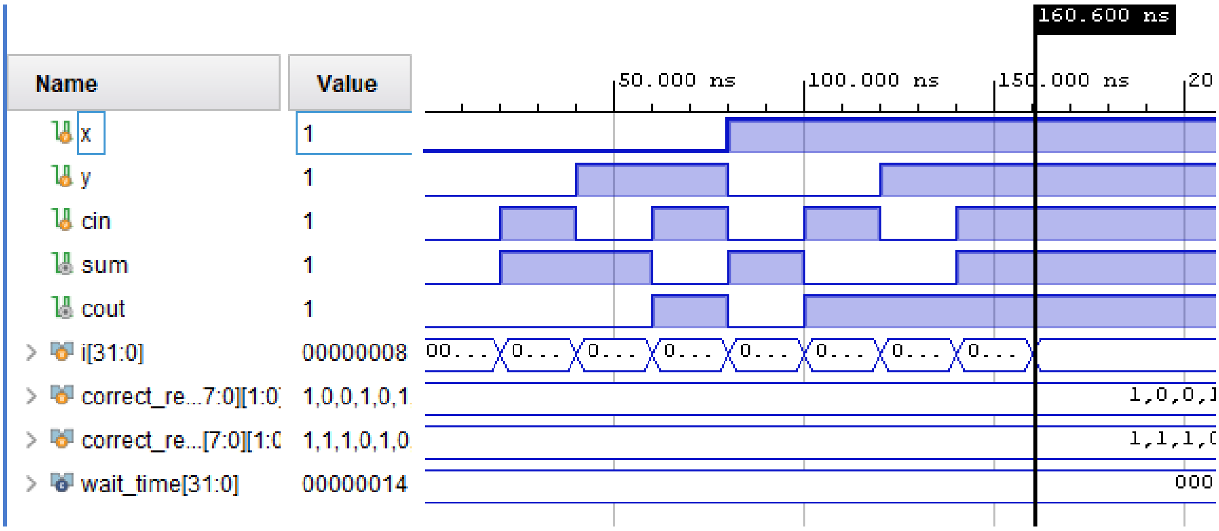
TCL Console output of full adder:

I coded correct results in testbench so output of testbench shows me everything works fine.



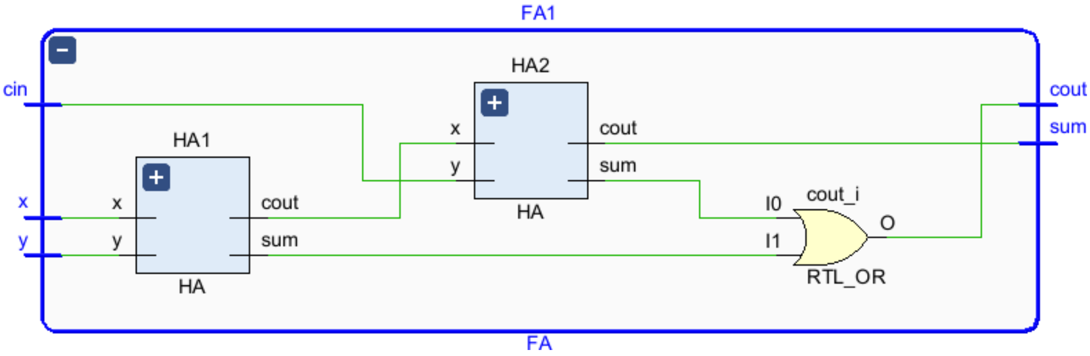
```
Time resolution is 1 ps
{x,y,cin}=000 => sum = 0, cout= 0  TRUE
{x,y,cin}=001 => sum = 1, cout= 0  TRUE
{x,y,cin}=010 => sum = 1, cout= 0  TRUE
{x,y,cin}=011 => sum = 0, cout= 1  TRUE
{x,y,cin}=100 => sum = 1, cout= 0  TRUE
{x,y,cin}=101 => sum = 0, cout= 1  TRUE
{x,y,cin}=110 => sum = 0, cout= 1  TRUE
{x,y,cin}=111 => sum = 1, cout= 1  TRUE
```

Behavioral Simulation of full adder:

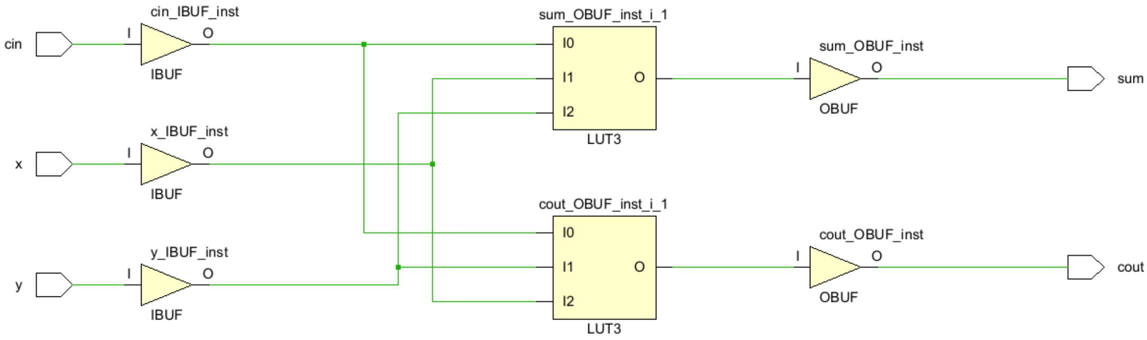


Name	Value
x	1
y	1
cin	1
sum	1
cout	1
i[31:0]	00000008
correct_re...7:0][1:0]	1,0,0,1,0,1
correct_re...[7:0][1:0]	1,1,1,0,1,0
wait_time[31:0]	00000014

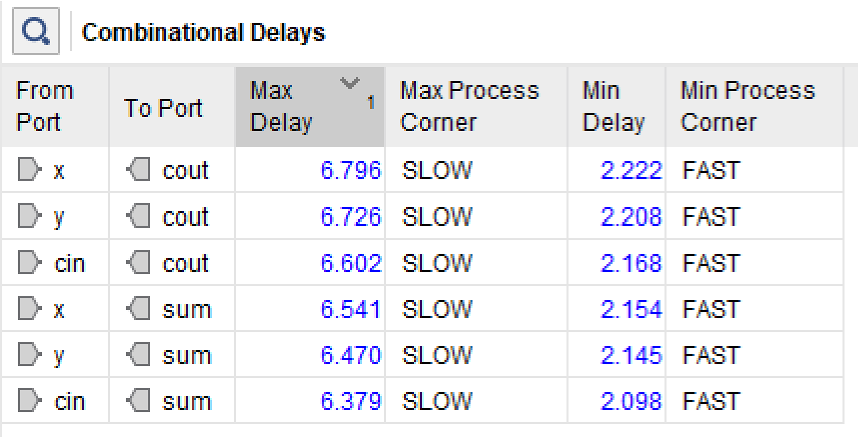
RTL schematic of full adder:



Technology Schematic of full adder:



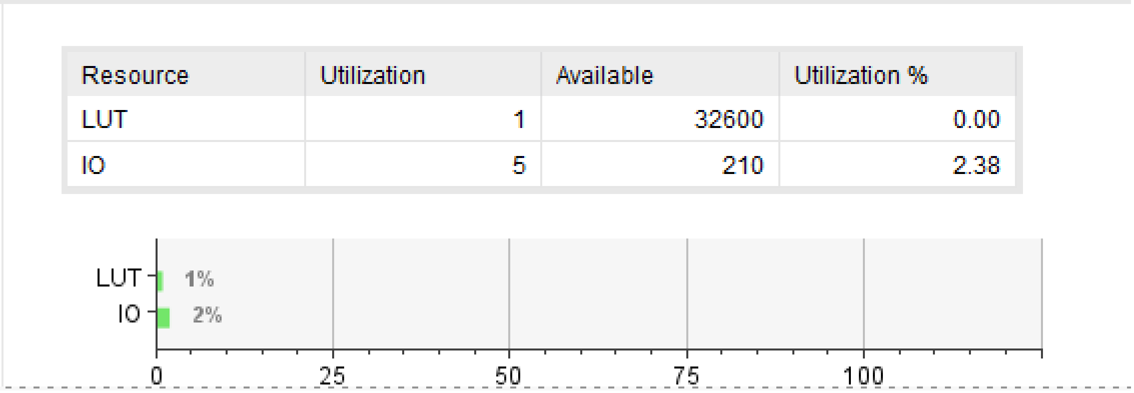
Timing report of full adder:



Combinational Delays

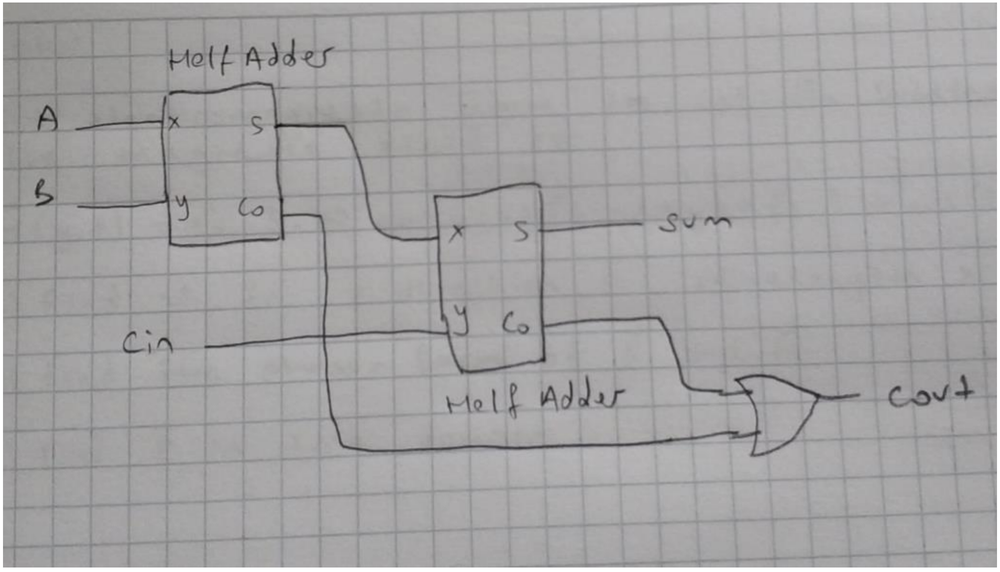
From Port	To Port	Max Delay	Max Process Corner	Min Delay	Min Process Corner
x	cout	6.796	SLOW	2.222	FAST
y	cout	6.726	SLOW	2.208	FAST
cin	cout	6.602	SLOW	2.168	FAST
x	sum	6.541	SLOW	2.154	FAST
y	sum	6.470	SLOW	2.145	FAST
cin	sum	6.379	SLOW	2.098	FAST

Utilization summary of full adder:



Resource	Utilization	Available	Utilization %
LUT	1	32600	0.00
IO	5	210	2.38

Hand drawing:



1 LUT used in full adder.

Ripple Carry Adder

This ripple carry adder adds 2 4-bit number with a carry input and outs 4 bit sum and 1 bit carry out.

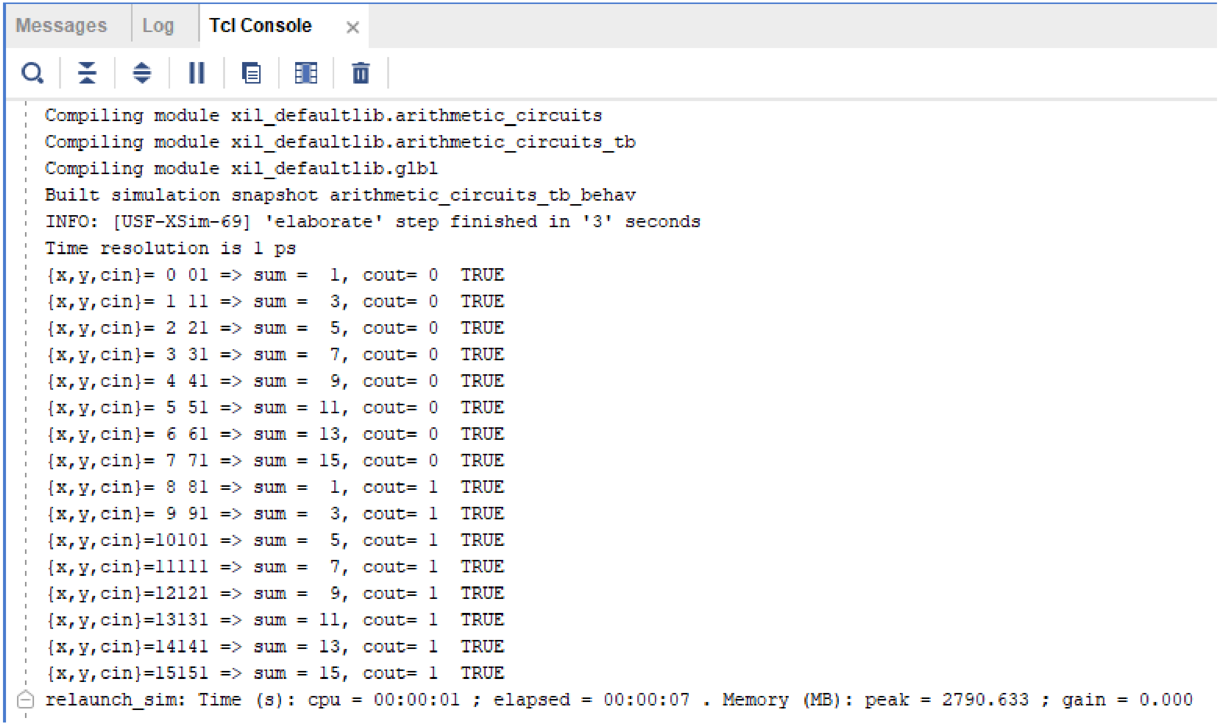
Source code of ripple carry adder:

```
`timescale 1ns / 1ps
module arithmetic_circuits(
input [3:0] x,
input [3:0] y,
input cin,
output cout,
output [3:0] sum
    );
RCA RCA1 (.x(x),.y(y),.cin(cin),.sum(sum),.cout(cout));
endmodule

module HA (
input x,
input y,
output cout,
output sum
);
assign sum = ((~x)&&y) || (x&&(~y));
assign cout = x&&y;
endmodule
module FA (
input x,
input y,
input cin,
output cout,
output sum);
wire sum1;
wire cout1;
wire sum2;
wire cout2;
HA HA1 (.x(x),.y(y),.sum(sum1),.cout(cout1));
HA HA2 (.x(sum1),.y(cin),.sum(sum2),.cout(cout2));
assign sum = sum2;
assign cout = cout2 || cout1;
endmodule
module RCA (
input [3:0] x,
input [3:0] y,
input cin,
output [3:0] sum,
output cout
);
wire cout1,cout2,cout3;
FA FA1 (.x(x[0]),.y(y[0]),.cin(cin),.sum(sum[0]),.cout(cout1));
FA FA2 (.x(x[1]),.y(y[1]),.cin(cout1),.sum(sum[1]),.cout(cout2));
FA FA3 (.x(x[2]),.y(y[2]),.cin(cout2),.sum(sum[2]),.cout(cout3));
FA FA4 (.x(x[3]),.y(y[3]),.cin(cout3),.sum(sum[3]),.cout(cout));
endmodule
```

Testbench of ripple carry adder:

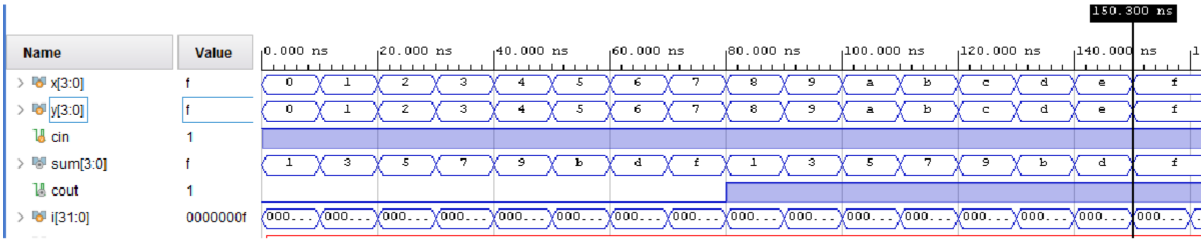
```
`timescale 1ns / 1ps

module arithmetic_circuits_tb();

reg [3:0] x;
reg [3:0] y;
reg cin;
wire [3:0] sum;
wire cout;
parameter wait_time = 10;
integer i;
integer a=0;
// true results

reg [4:0] correct_results[31:0];

initial begin
correct_results[0]= 5'd1 ;
correct_results[1]= 5'd3 ;correct_results[2]= 5'd5 ;
correct_results[3]= 5'd7 ;correct_results[4]= 5'd9 ;
correct_results[5]= 5'd11 ;correct_results[6]= 5'd13 ;
correct_results[7]= 5'd15 ;correct_results[8]= 5'd17 ;
correct_results[9]= 5'd19 ;correct_results[ 10]= 5'd21 ;
correct_results[ 11 ]= 5'd23 ;correct_results[ 12 ]= 5'd25 ;
correct_results[ 13 ]= 5'd27 ;correct_results[ 14 ]= 5'd29 ;
correct_results[ 15 ]= 5'd31 ;

end

arithmetic_circuits L1
(.x(x),.y(y),.cin(cin),.sum(sum),.cout(cout));

initial begin

for (i=0;i<16;i=i+1)
begin
x=i;
y=i;
cin=1;
#wait_time;
$write ("{x,y,cin}=%d%d%d => sum = %d, cout=
%d",x,y,cin,sum,cout);
if({cout,sum} == correct_results[i])
$display("  TRUE");
else
$display("  FALSE");
end

end
endmodule
```

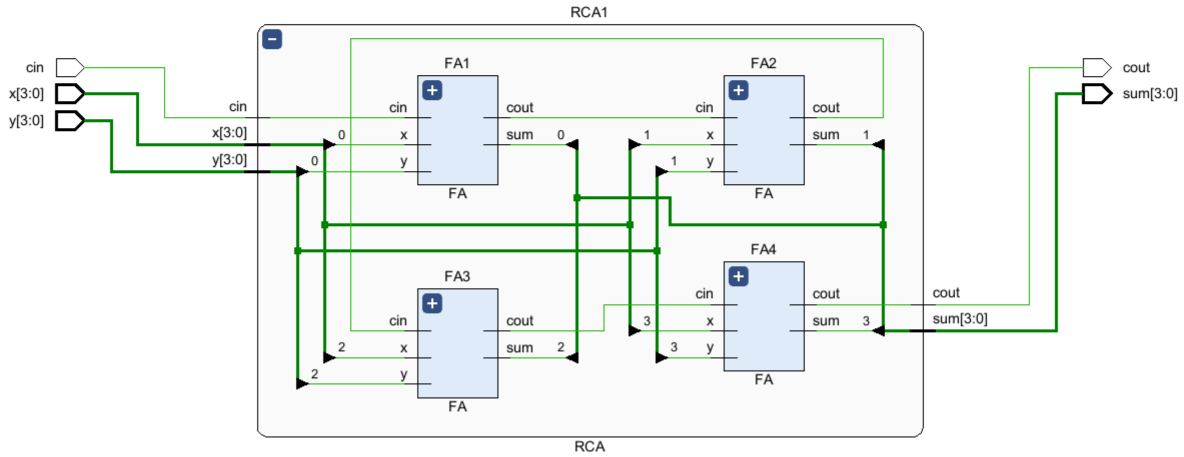
Tcl console output of ripple carry adder:



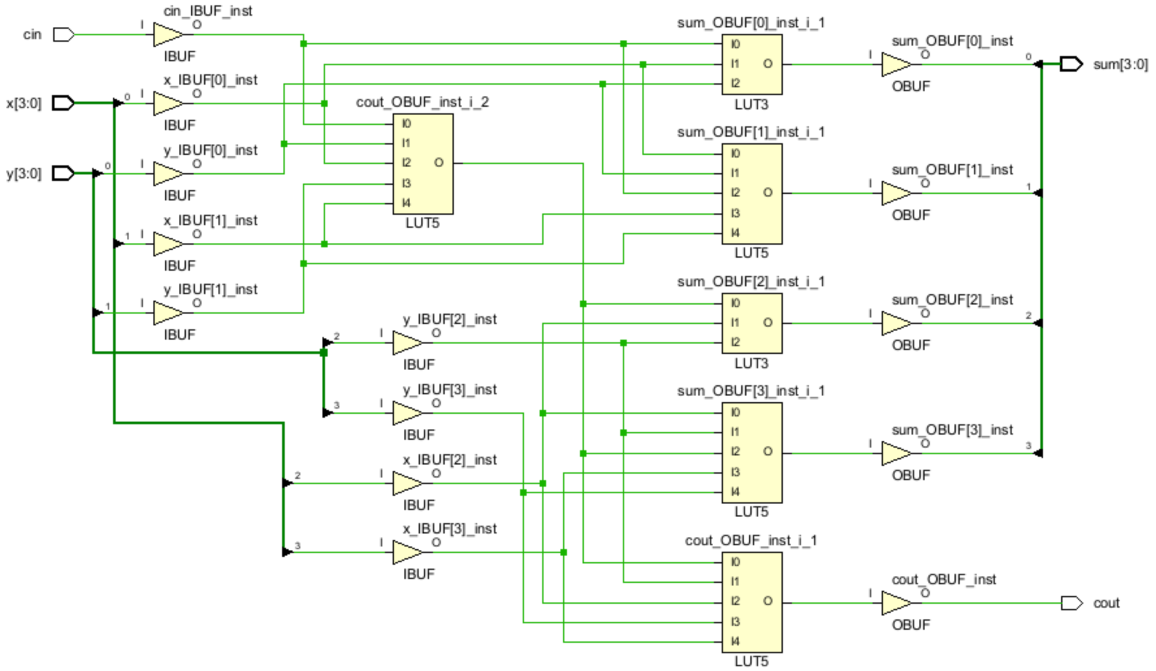
```
Compiling module xil_defaultlib.arithmetic_circuits
Compiling module xil_defaultlib.arithmetic_circuits_tb
Compiling module xil_defaultlib.glbl
Built simulation snapshot arithmetic_circuits_tb_behav
INFO: [USF-XSim-69] 'elaborate' step finished in '3' seconds
Time resolution is 1 ps
{x,y,cin}= 0 01 => sum =  1, cout= 0  TRUE
{x,y,cin}= 1 11 => sum =  3, cout= 0  TRUE
{x,y,cin}= 2 21 => sum =  5, cout= 0  TRUE
{x,y,cin}= 3 31 => sum =  7, cout= 0  TRUE
{x,y,cin}= 4 41 => sum =  9, cout= 0  TRUE
{x,y,cin}= 5 51 => sum = 11, cout= 0  TRUE
{x,y,cin}= 6 61 => sum = 13, cout= 0  TRUE
{x,y,cin}= 7 71 => sum = 15, cout= 0  TRUE
{x,y,cin}= 8 81 => sum =  1, cout= 1  TRUE
{x,y,cin}= 9 91 => sum =  3, cout= 1  TRUE
{x,y,cin}=10101 => sum =  5, cout= 1  TRUE
{x,y,cin}=11111 => sum =  7, cout= 1  TRUE
{x,y,cin}=12121 => sum =  9, cout= 1  TRUE
{x,y,cin}=13131 => sum = 11, cout= 1  TRUE
{x,y,cin}=14141 => sum = 13, cout= 1  TRUE
{x,y,cin}=15151 => sum = 15, cout= 1  TRUE
relaunch_sim: Time (s): cpu = 00:00:01 ; elapsed = 00:00:07 . Memory (MB): peak = 2790.633 ; gain = 0.000
```

Behavioral Simulation and RTL schematic of ripple carry adder:

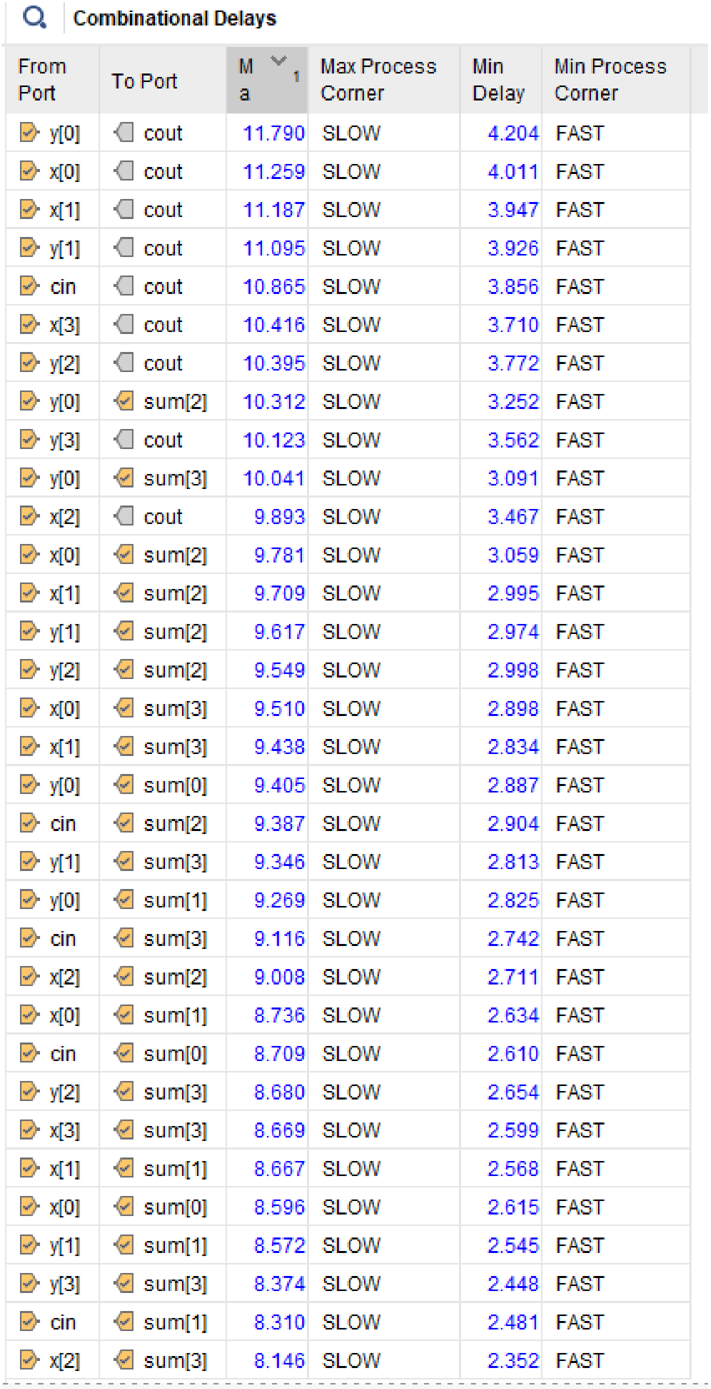




Technology schematic of ripple carry adder



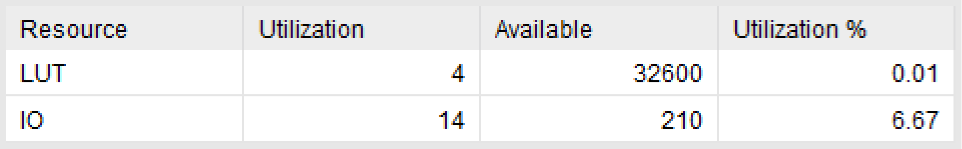
Timing report of ripple carry adder:



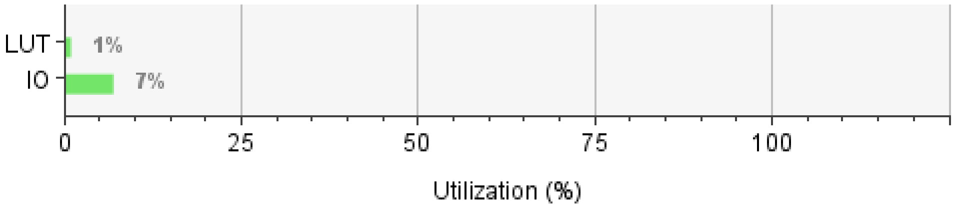
Combinational Delays

From Port	To Port	M a	Max Process Corner	Min Delay	Min Process Corner
y[0]	cout	11.790	SLOW	4.204	FAST
x[0]	cout	11.259	SLOW	4.011	FAST
x[1]	cout	11.187	SLOW	3.947	FAST
y[1]	cout	11.095	SLOW	3.926	FAST
cin	cout	10.865	SLOW	3.856	FAST
x[3]	cout	10.416	SLOW	3.710	FAST
y[2]	cout	10.395	SLOW	3.772	FAST
y[0]	sum[2]	10.312	SLOW	3.252	FAST
y[3]	cout	10.123	SLOW	3.562	FAST
y[0]	sum[3]	10.041	SLOW	3.091	FAST
x[2]	cout	9.893	SLOW	3.467	FAST
x[0]	sum[2]	9.781	SLOW	3.059	FAST
x[1]	sum[2]	9.709	SLOW	2.995	FAST
y[1]	sum[2]	9.617	SLOW	2.974	FAST
y[2]	sum[2]	9.549	SLOW	2.998	FAST
x[0]	sum[3]	9.510	SLOW	2.898	FAST
x[1]	sum[3]	9.438	SLOW	2.834	FAST
y[0]	sum[0]	9.405	SLOW	2.887	FAST
cin	sum[2]	9.387	SLOW	2.904	FAST
y[1]	sum[3]	9.346	SLOW	2.813	FAST
y[0]	sum[1]	9.269	SLOW	2.825	FAST
cin	sum[3]	9.116	SLOW	2.742	FAST
x[2]	sum[2]	9.008	SLOW	2.711	FAST
x[0]	sum[1]	8.736	SLOW	2.634	FAST
cin	sum[0]	8.709	SLOW	2.610	FAST
y[2]	sum[3]	8.680	SLOW	2.654	FAST
x[3]	sum[3]	8.669	SLOW	2.599	FAST
x[1]	sum[1]	8.667	SLOW	2.568	FAST
x[0]	sum[0]	8.596	SLOW	2.615	FAST
y[1]	sum[1]	8.572	SLOW	2.545	FAST
y[3]	sum[3]	8.374	SLOW	2.448	FAST
cin	sum[1]	8.310	SLOW	2.481	FAST
x[2]	sum[3]	8.146	SLOW	2.352	FAST

Utilization summary of ripple carry adder:



Resource	Utilization	Available	Utilization %
LUT	4	32600	0.01
IO	14	210	6.67



4 LUTs used in 4-bit RCA .

PARAMETRIZED RIPPLE CARRY ADDER

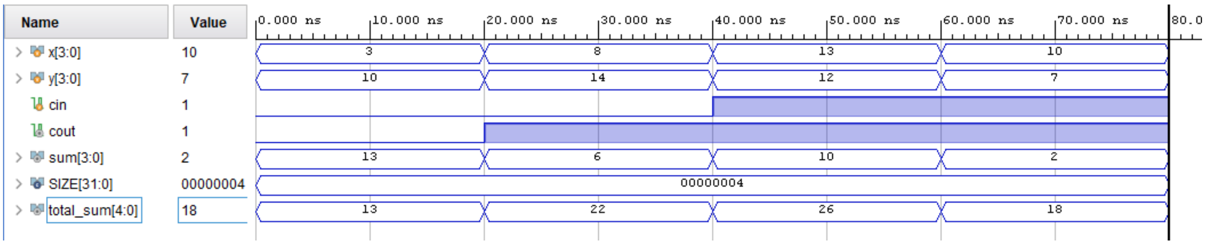
Source Code of parametrized ripple carry adder:

```
`timescale 1ns / 1ps
module parametric_RCA
#(parameter SIZE = 8)(
input [(SIZE-1):0] x,
input [(SIZE-1):0] y,
input cin,
output cout,
output [(SIZE-1):0] sum);

genvar i;
wire [(SIZE-1):0] cout_inside;
assign cout = cout_inside[SIZE-1];
generate
for (i=0;i<SIZE;i=i+1)
    begin
    if(i==0)
    FA u0
(.x(x[i]),.y(y[i]),.cin(cin),.sum(sum[i]),.cout(cout_inside[i]));
    else
    FA u1 (.x(x[i]),.y(y[i]),.cin(cout_inside[i-
1]),.sum(sum[i]),.cout(cout_inside[i]));
    end
endgenerate
endmodule

module HA (
input x,
input y,
output cout,
output sum
);
assign sum = ((~x)&&y) || (x&&(~y));
assign cout = x&&y;
endmodule

module FA (
input x,
input y,
input cin,
output cout,
output sum);
wire sum1;
wire cout1;
wire sum2;
wire cout2;
HA HA1 (.x(x),.y(y),.sum(sum1),.cout(cout1));
HA HA2 (.x(sum1),.y(cin),.sum(sum2),.cout(cout2));
assign sum = sum2;
assign cout = cout2 || cout1;
endmodule
```

Testbench Code of parametric ripple carry adder:

```
`timescale 1ns / 1ps

module parametric_RCA_tb;
localparam SIZE = 8;
reg [(SIZE-1):0] x;
reg [(SIZE-1):0] y;
reg cin;
wire cout;
wire [(SIZE-1):0] sum;
wire [SIZE:0] total_sum;

assign total_sum = {cout,sum};

parametric_RCA #(.SIZE(SIZE)) U0
(.x(x),.y(y),.cin(cin),.cout(cout),.sum(sum));



initial begin
x=8'd217;
y=8'd65;
cin=0;
#20;
x=8'd200;
y=8'd150;
cin=0;
#20;
x=8'd110;
y=8'd221;
cin=1;
#20;
x=8'd249;
y=8'd107;
cin=1;
#20;
$finish;
end
```

Simulation Results of parametric RCA:

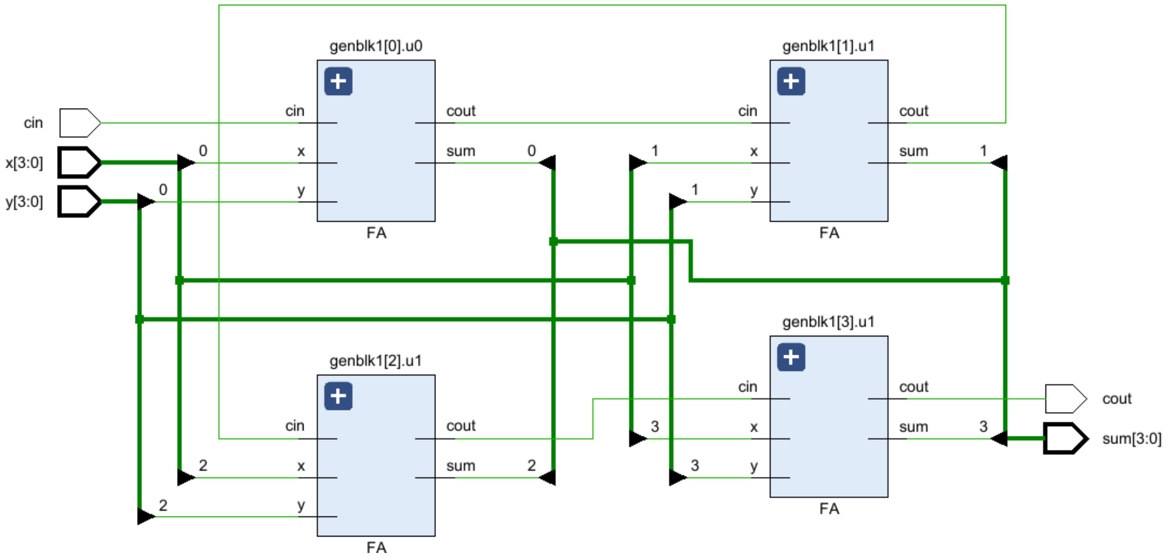

Name
Value
0.000 ns
10.000 ns
20.000 ns
30.000 ns
40.000 ns
50.000 ns
60.000 ns
70.000 ns
80.0
x[3:0] 10 | 3 | 8 | 13 | 10
y[3:0] 7 | 10 | 14 | 12 | 7
cin 1
cout 1
sum[3:0] 2 | 13 | 6 | 10 | 2
SIZE[31:0] 00000004 | 00000004
total_sum[4:0] 18 | 13 | 22 | 26 | 18

We expect 10+7+1 = 18 in total sum : {cout,sum} so this case is working true.

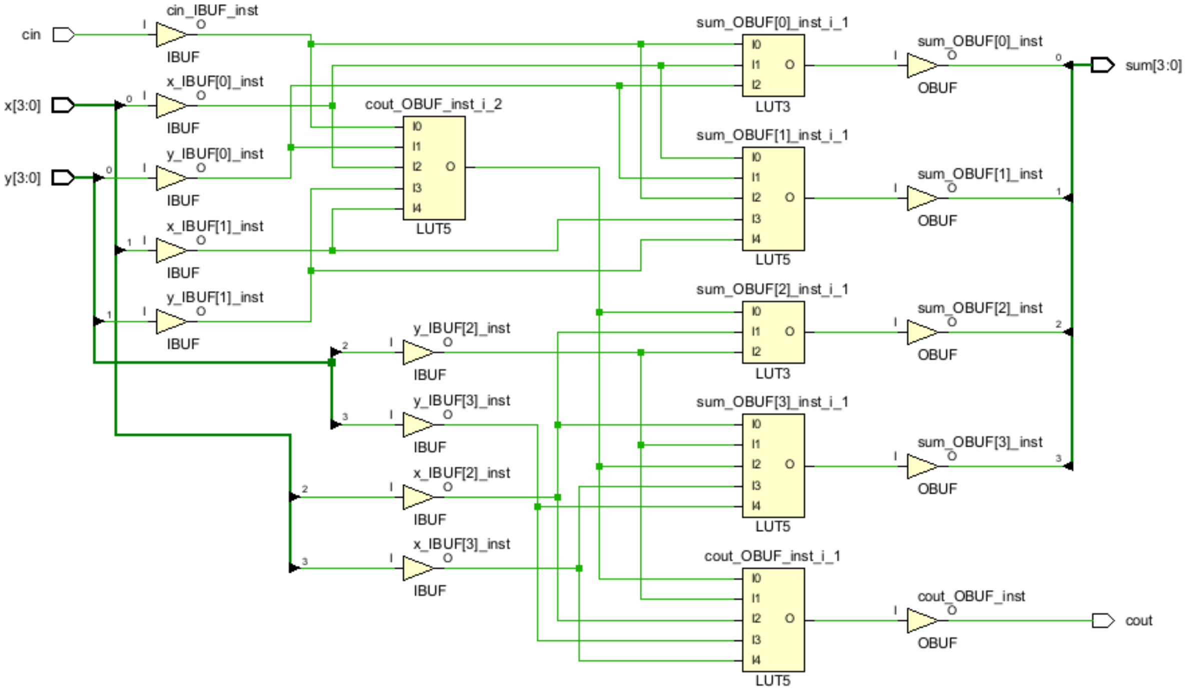
We expect 8+14+0=22 in total sum : {cout,sum} and this case is working true too.

We expect 3+10+0=0 in total sum {cout,sum} and this case is working true.

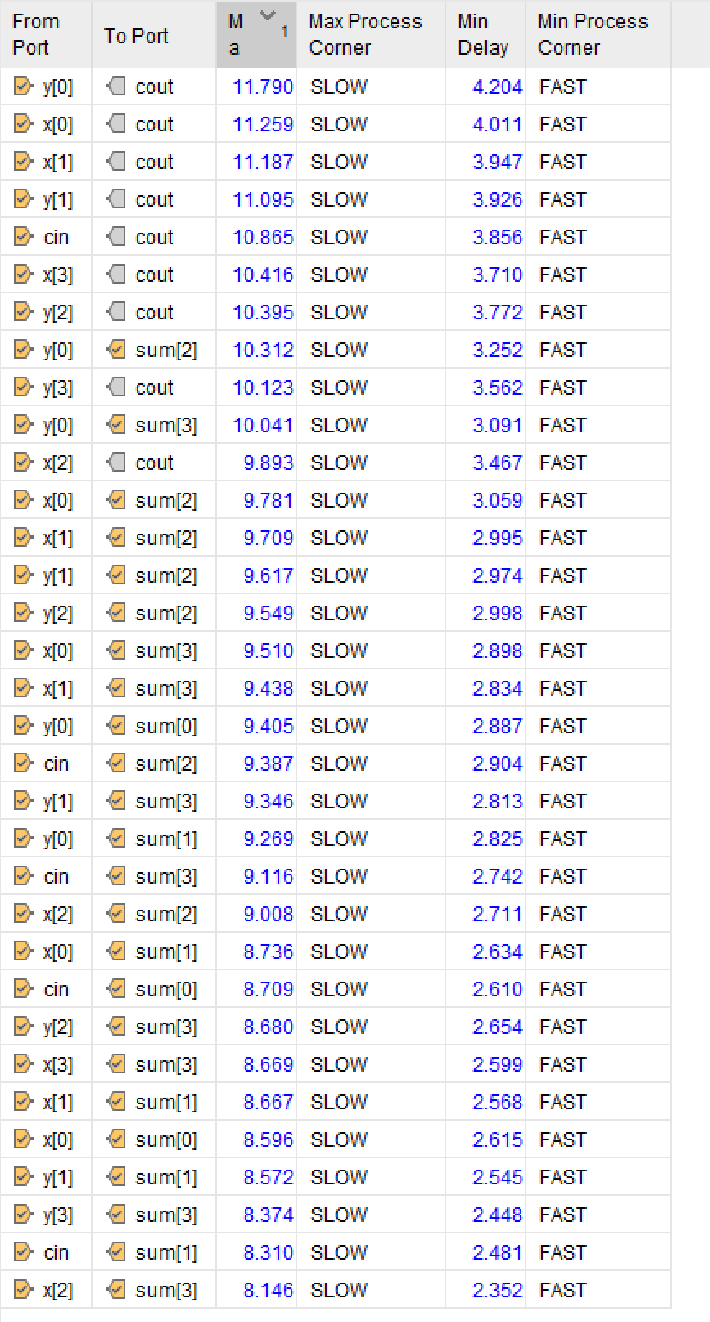
RTL Schematic of parametric RCA:



Technology Schematic of parametric RCA:

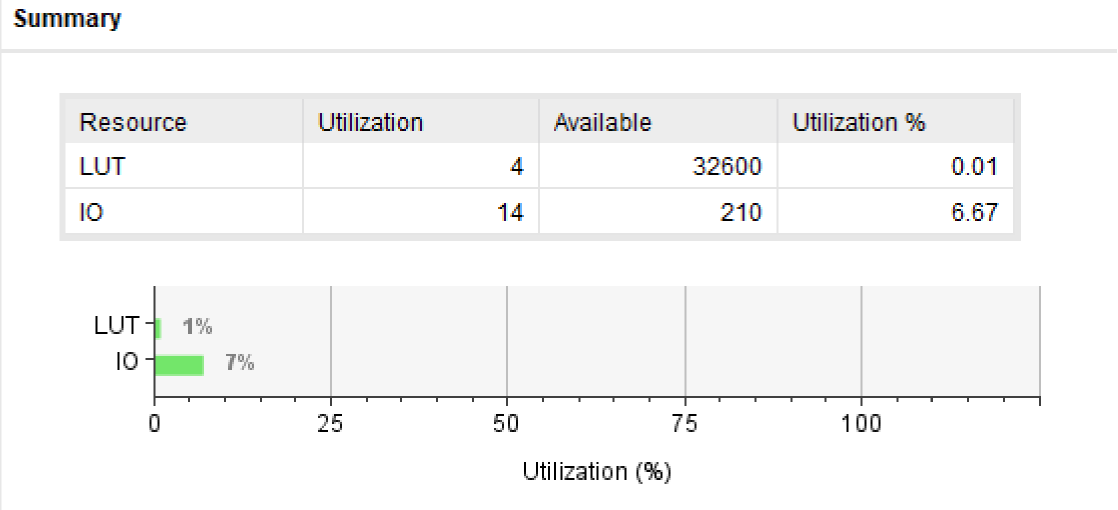


Timing summary of parametric RCA:



From Port	To Port	Ma	Max Process Corner	Min Delay	Min Process Corner
y[0]	cout	11.790	SLOW	4.204	FAST
x[0]	cout	11.259	SLOW	4.011	FAST
x[1]	cout	11.187	SLOW	3.947	FAST
y[1]	cout	11.095	SLOW	3.926	FAST
cin	cout	10.865	SLOW	3.856	FAST
x[3]	cout	10.416	SLOW	3.710	FAST
y[2]	cout	10.395	SLOW	3.772	FAST
y[0]	sum[2]	10.312	SLOW	3.252	FAST
y[3]	cout	10.123	SLOW	3.562	FAST
y[0]	sum[3]	10.041	SLOW	3.091	FAST
x[2]	cout	9.893	SLOW	3.467	FAST
x[0]	sum[2]	9.781	SLOW	3.059	FAST
x[1]	sum[2]	9.709	SLOW	2.995	FAST
y[1]	sum[2]	9.617	SLOW	2.974	FAST
y[2]	sum[2]	9.549	SLOW	2.998	FAST
x[0]	sum[3]	9.510	SLOW	2.898	FAST
x[1]	sum[3]	9.438	SLOW	2.834	FAST
y[0]	sum[0]	9.405	SLOW	2.887	FAST
cin	sum[2]	9.387	SLOW	2.904	FAST
y[1]	sum[3]	9.346	SLOW	2.813	FAST
y[0]	sum[1]	9.269	SLOW	2.825	FAST
cin	sum[3]	9.116	SLOW	2.742	FAST
x[2]	sum[2]	9.008	SLOW	2.711	FAST
x[0]	sum[1]	8.736	SLOW	2.634	FAST
cin	sum[0]	8.709	SLOW	2.610	FAST
y[2]	sum[3]	8.680	SLOW	2.654	FAST
x[3]	sum[3]	8.669	SLOW	2.599	FAST
x[1]	sum[1]	8.667	SLOW	2.568	FAST
x[0]	sum[0]	8.596	SLOW	2.615	FAST
y[1]	sum[1]	8.572	SLOW	2.545	FAST
y[3]	sum[3]	8.374	SLOW	2.448	FAST
cin	sum[1]	8.310	SLOW	2.481	FAST
x[2]	sum[3]	8.146	SLOW	2.352	FAST

Utilization report of parametric RCA:

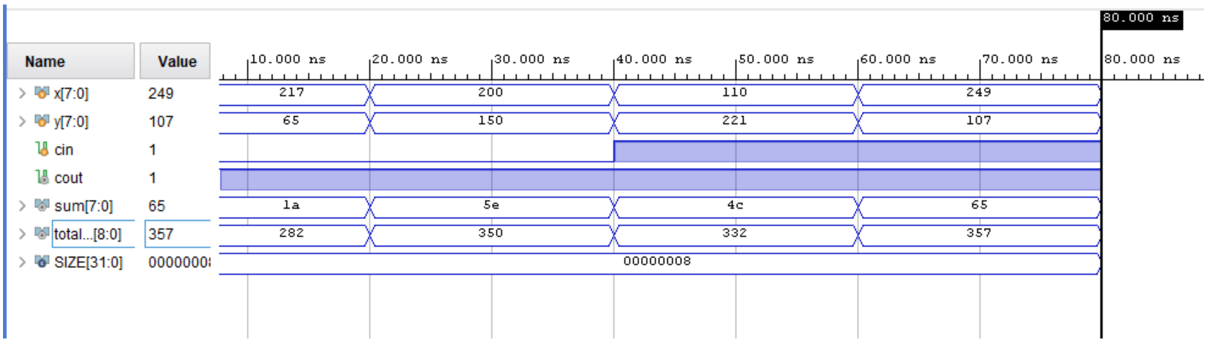


Summary

Resource	Utilization	Available	Utilization %
LUT	4	32600	0.01
IO	14	210	6.67

4 LUTs used in 4 bit parametric RCA.

SIZE=8



X=249 y=65 and cin=0 we expect 282 in total sum, and that’s correct.
x=200 y=150 and cin= 0 , we expect 350 in total sum, and that’s correct.
X=110 y=221 and cin=1 = we expect 332 in total sum and that’s correct.

Carry lookahead adder:

This is 4-bit inputs and a carry input, carry lookahead adder module.

Source code of carry lookahead adder:

```
`timescale 1ns / 1ps
module CLA(
input [3:0] x,
input [3:0] y,
input cin,
output cout,
output [3:0] sum
    );

// P = x^y
// G = x&y
// Digital Design book expressions:
// Si = Pi ^^ Ci
// Ci+1 = Gi + PiCi

// C1 = G0 + P0Cin
// C2 = G1 + P1(G0+P0Cin)
// C3 = G2 + P2(G1+(P1(G0+P0Cin)))
// C4 = G3 + P3(G2 + P2(G1+(P1(G0+P0Cin))))
wire [3:0] p,g,c;
assign p = x^y;
assign g = x&y;
assign c[0] = cin;
assign c[1] = g[0] | (p[0]&cin);
assign c[2] = g[1] | (p[1]&(g[0] | (p[0]&cin)));
assign c[3] = g[2] | (p[2]&(g[1] | (p[1]&(g[0] | (p[0]&cin)))));
assign cout = g[3] | (p[3]& (g[2] | (p[2]&(g[1] | (p[1]&(g[0] |
(p[0]&cin)))))));
assign sum = p^c;

endmodule
```

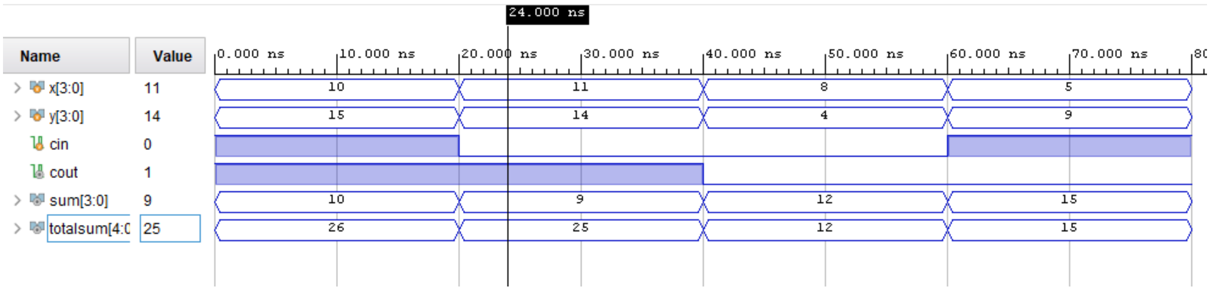
Here pi = xi ^ yi and gi = xi.yi.

And we get carry input values from the Ci+1 = Gi+PiCi formula so that should be faster than ripple carry adder

Testbench Code of carry lookahead adder:

```
`timescale 1ns / 1ps

module CLA_tb;


reg [3:0] x;
reg [3:0] y;
reg cin;
wire cout;
wire [3:0] sum;
wire [4:0] totalsum;
assign totalsum = {cout,sum};

CLA Y1 (.x(x),.y(y),.cin(cin),.cout(cout),.sum(sum));

initial begin

x= 4'd10;
y = 4'd15;
cin = 1;
#20;
x= 4'd11;
y = 4'd14;
cin = 0;
#20;
x= 4'd8;
y = 4'd4;
cin = 0;
#20;
x= 4'd5;
y = 4'd9;
cin = 1;
#20;
$finish;
end
endmodule
```

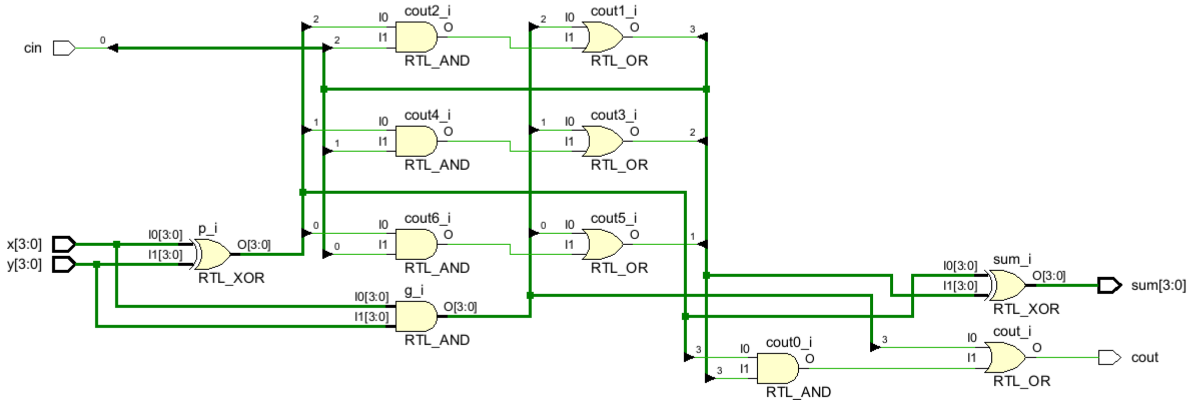
Simulation Results of CLA:



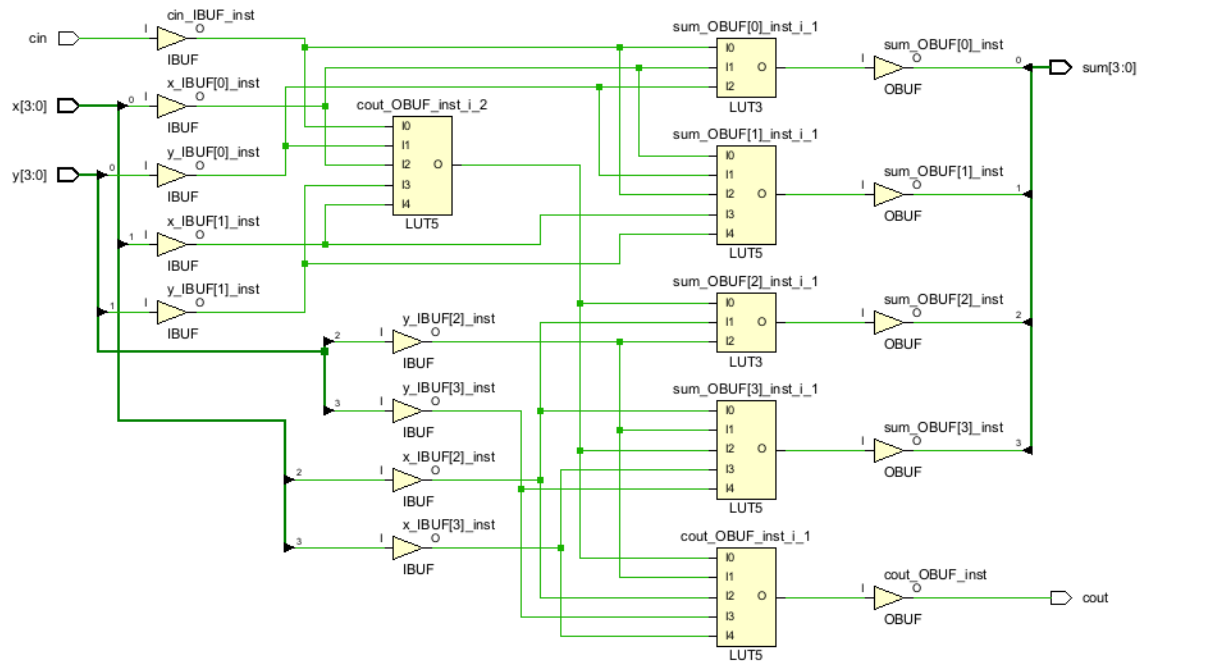
Name	Value
x[3:0]	11
y[3:0]	14
cin	0
cout	1
sum[3:0]	9
totalsum[4:0	25

Here is x = 11 , y=14 and cin=0 so sum should be 25 but it doesn't fit in 4 bits so with cout it becomes 25 as other comes true that way too.

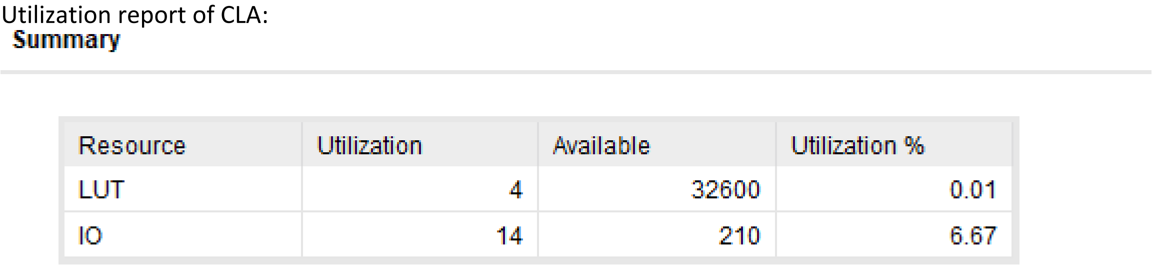
RTL Schematic of CLA:



Technology Schematic of CLA:

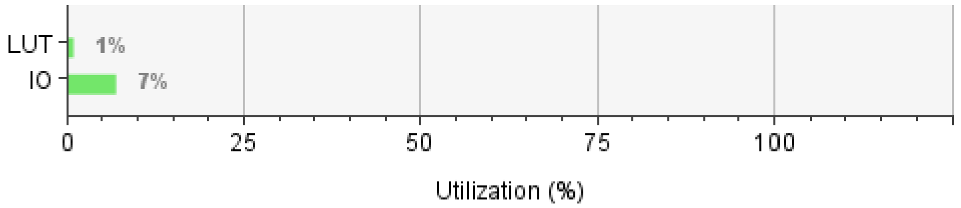


Utilization report of CLA:

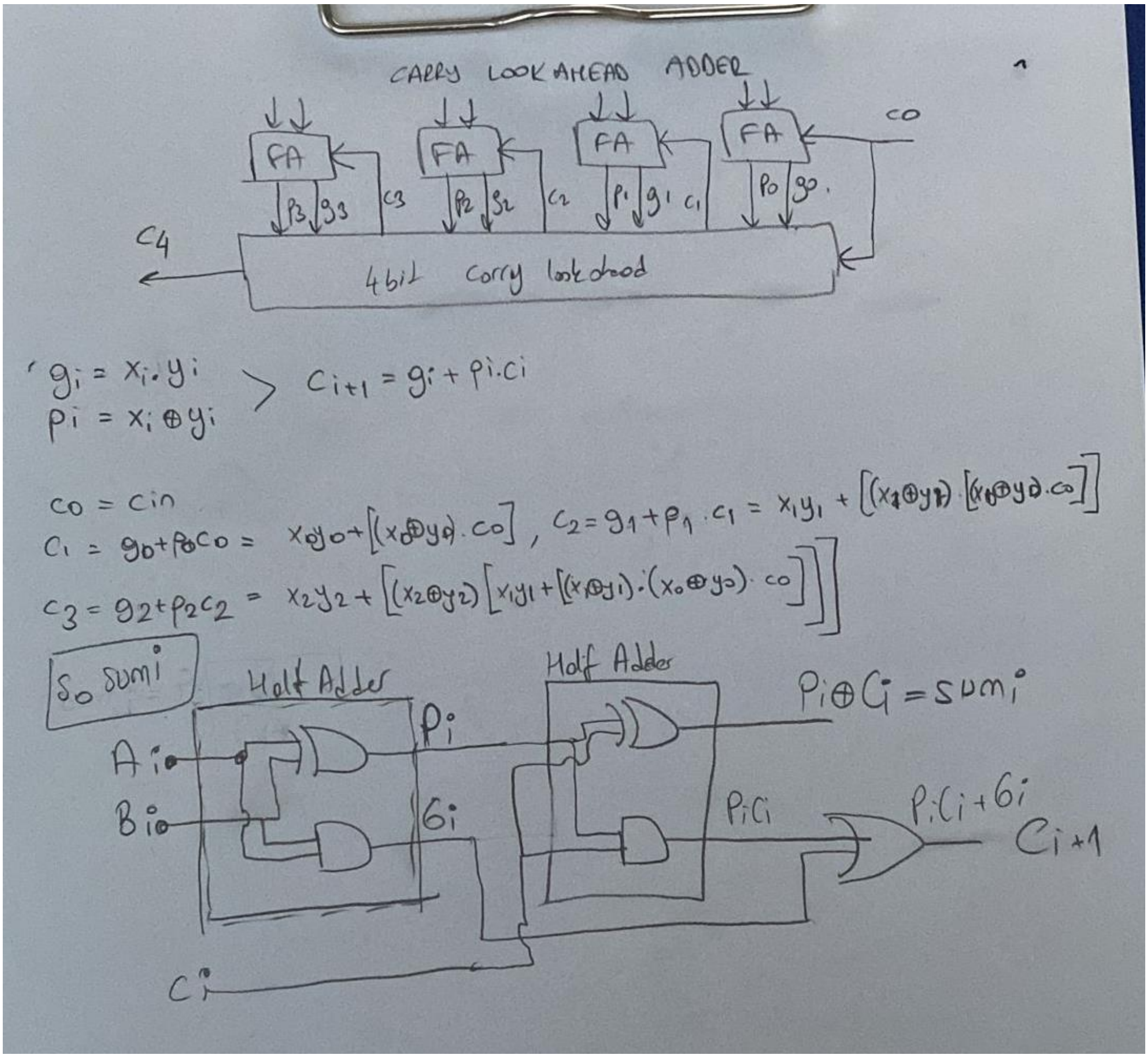


Summary

Resource	Utilization	Available	Utilization %
LUT	4	32600	0.01
IO	14	210	6.67



Sheet calculations of CLA:



$g_i = x_i \cdot y_i$

$p_i = x_i \oplus y_i$ $>$ $c_{i+1} = g_i + p_i \cdot c_i$

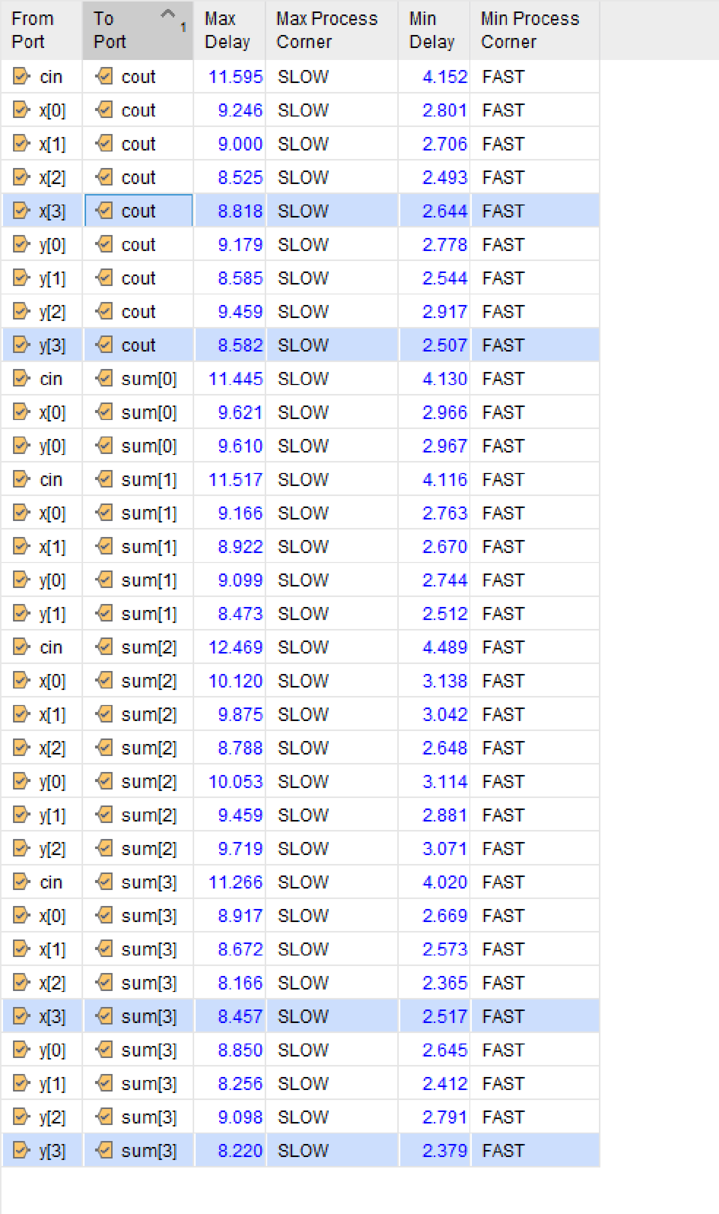
$c_0 = c_{in}$

$c_1 = g_0 + p_0 c_0 = x_0 y_0 + [(x_0 \oplus y_0) \cdot c_0]$, $c_2 = g_1 + p_1 \cdot c_1 = x_1 y_1 + [(x_1 \oplus y_1) \cdot [(x_0 \oplus y_0) \cdot c_0]]$

$c_3 = g_2 + p_2 c_2 = x_2 y_2 + [(x_2 \oplus y_2) [x_1 y_1 + [(x_1 \oplus y_1) \cdot (x_0 \oplus y_0) \cdot c_0]]]$

$P_i \oplus C_i = sum_i$

Timing report of CLA:



From Port	To Port	Max Delay	Max Process Corner	Min Delay	Min Process Corner
cin	cout	11.595	SLOW	4.152	FAST
x[0]	cout	9.246	SLOW	2.801	FAST
x[1]	cout	9.000	SLOW	2.706	FAST
x[2]	cout	8.525	SLOW	2.493	FAST
x[3]	cout	8.818	SLOW	2.644	FAST
y[0]	cout	9.179	SLOW	2.778	FAST
y[1]	cout	8.585	SLOW	2.544	FAST
y[2]	cout	9.459	SLOW	2.917	FAST
y[3]	cout	8.582	SLOW	2.507	FAST
cin	sum[0]	11.445	SLOW	4.130	FAST
x[0]	sum[0]	9.621	SLOW	2.966	FAST
y[0]	sum[0]	9.610	SLOW	2.967	FAST
cin	sum[1]	11.517	SLOW	4.116	FAST
x[0]	sum[1]	9.166	SLOW	2.763	FAST
x[1]	sum[1]	8.922	SLOW	2.670	FAST
y[0]	sum[1]	9.099	SLOW	2.744	FAST
y[1]	sum[1]	8.473	SLOW	2.512	FAST
cin	sum[2]	12.469	SLOW	4.489	FAST
x[0]	sum[2]	10.120	SLOW	3.138	FAST
x[1]	sum[2]	9.875	SLOW	3.042	FAST
x[2]	sum[2]	8.788	SLOW	2.648	FAST
y[0]	sum[2]	10.053	SLOW	3.114	FAST
y[1]	sum[2]	9.459	SLOW	2.881	FAST
y[2]	sum[2]	9.719	SLOW	3.071	FAST
cin	sum[3]	11.266	SLOW	4.020	FAST
x[0]	sum[3]	8.917	SLOW	2.669	FAST
x[1]	sum[3]	8.672	SLOW	2.573	FAST
x[2]	sum[3]	8.166	SLOW	2.365	FAST
x[3]	sum[3]	8.457	SLOW	2.517	FAST
y[0]	sum[3]	8.850	SLOW	2.645	FAST
y[1]	sum[3]	8.256	SLOW	2.412	FAST
y[2]	sum[3]	9.098	SLOW	2.791	FAST
y[3]	sum[3]	8.220	SLOW	2.379	FAST

As we see last full adder in CLA
x[3] to cout delay is = 8.818 ns
y[3] to cout delay is = 8.582 ns

But in RCA

x[3] to cout delay is = 10.4 ns
y[3] to cout delay is = 10.123 ns

Add-Sub with overflow module

The blank operation is **XOR.**

When cin = 1, it is substracting.

When cin=0 it is adding.

When output V = 1, the 4 bit sum is not enough. If we put cout as the MSB and get that 5 bit value as the total sum, this value will be correct value of operation. When output V=0, total sum could be wrong so it is enough to take 4 bit sum as a result value of the operation.

Source Code of Add-Sub with overflow:

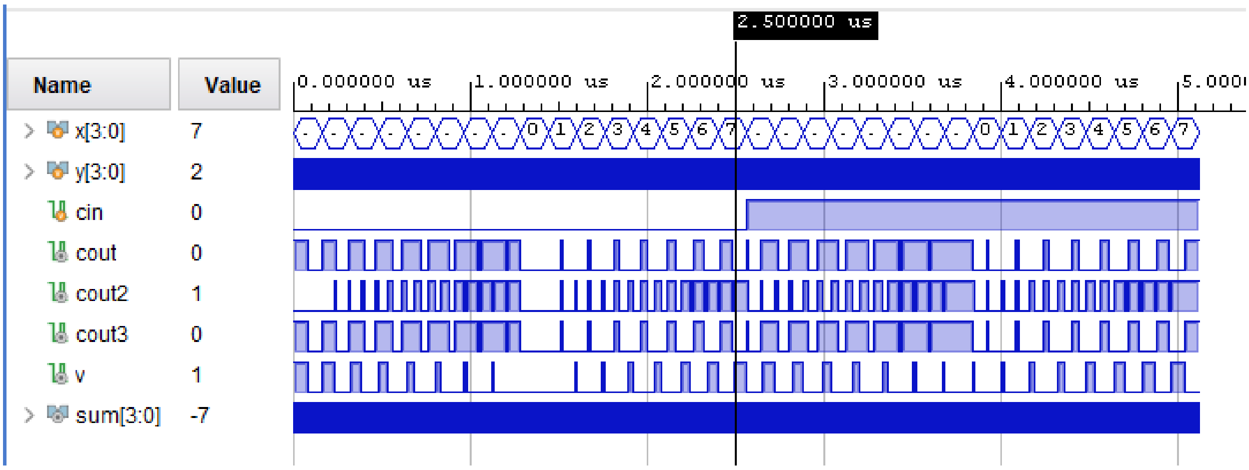
```
`timescale 1ns / 1ps

module Add_Sub(
input [3:0] x,
input [3:0] y,
input cin,
output cout,
output v,
output [3:0] sum
    );

wire [3:0] y_n;
wire [3:0] c_inside;
assign y_n = y^{4{cin}};
assign cout = c_inside[3];
assign v = c_inside[3]^c_inside[2];

FA FA1 (.x(x[0]),.y(y_n[0]),.cin(cin),.cout(c_inside[0]),.sum(sum[0]));
FA FA2 (.x(x[1]),.y(y_n[1]),.cin(c_inside[0]),.cout(c_inside[1]),.sum(sum[1]));
FA FA3 (.x(x[2]),.y(y_n[2]),.cin(c_inside[1]),.cout(c_inside[2]),.sum(sum[2]));
FA FA4 (.x(x[3]),.y(y_n[3]),.cin(c_inside[2]),.cout(c_inside[3]),.sum(sum[3]));

endmodule



module HA (
input x,
input y,
output cout,
output sum
);
assign sum = ((~x)&&y) || (x&&(~y));
assign cout = x&&y;
endmodule

module FA (
input x,
input y,
input cin,
output cout,
output sum);
wire sum1;
wire cout1;
wire sum2;
wire cout2;

HA HA1 (.x(x),.y(y),.sum(sum1),.cout(cout1));
HA HA2 (.x(sum1),.y(cin),.sum(sum2),.cout(cout2));

assign sum = sum2;
assign cout = cout2 || cout1;

endmodule
```

Testbench Code:

```
`timescale 1ns / 1ps
module Add_Sub_tb;

reg [3:0] x=0;
reg [3:0] y=0;
reg cin=0;
wire cout;
wire [3:0] sum;
wire v;
wire [4:0] totalsum;
assign totalsum = {cout,sum};
Add_Sub AS1 (.x(x),.y(y),.cin(cin),.cout(cout),.sum(sum),.v(v));

integer i;
integer z;
initial begin
cin = 0;
for(i=-8;i<8;i=i+1)
begin
            x=i;
            for(z=-8;z<8;z=z+1)
            begin
            y=z;
            #10;
            end

end
cin=1;
for(i=-8;i<8;i=i+1)
begin
            x=i;
            for(z=-8;z<8;z=z+1)
            begin
            y=z;
            #10;
            end

end
$finish;
end
endmodule
```

Simulation results of Add_Sub:

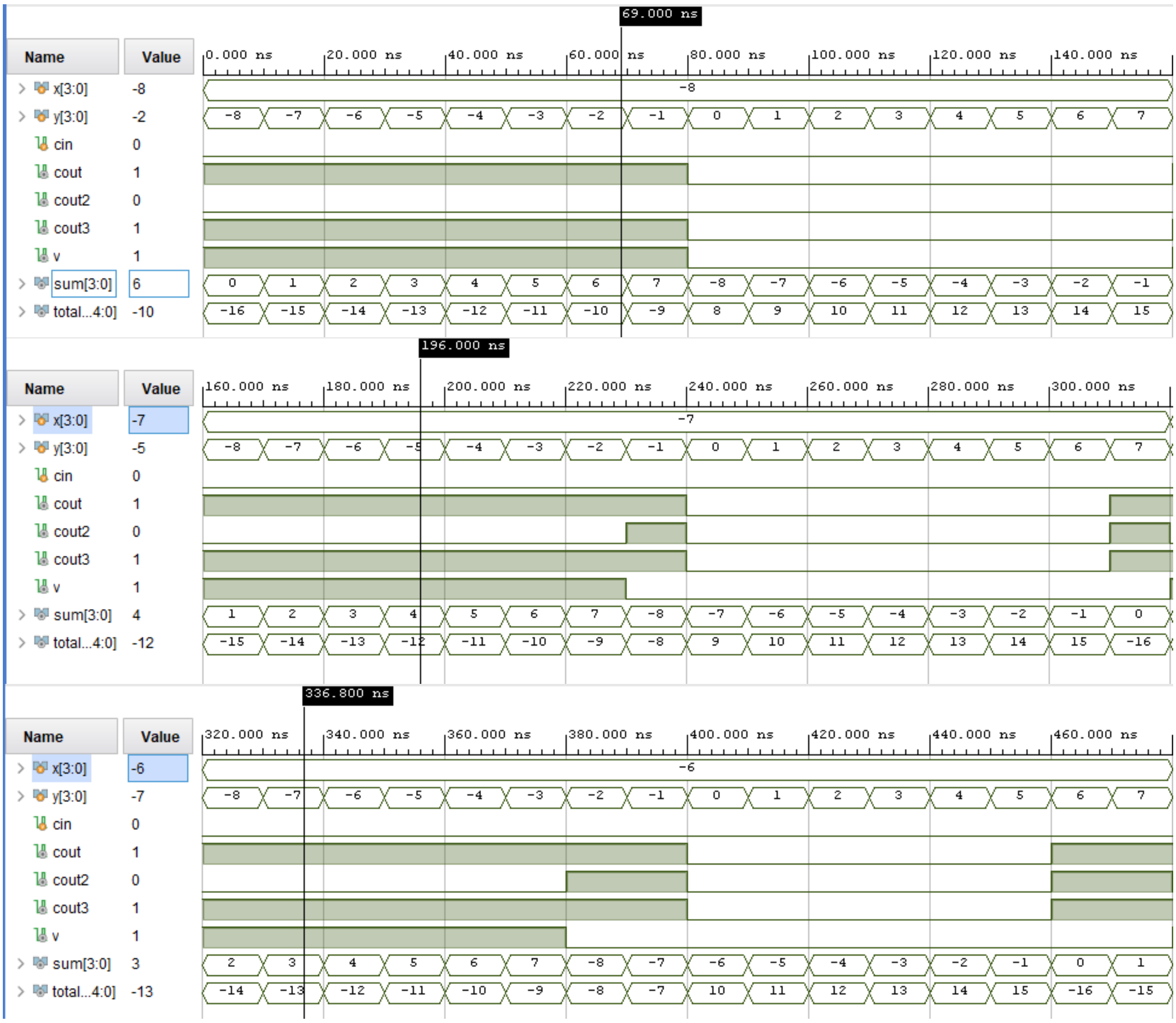
General Look to Add-Sub simulation:



Name	Value
x[3:0]	7
y[3:0]	2
cin	0
cout	0
cout2	1
cout3	0
v	1
sum[3:0]	-7

Spesific Looks to Add-Sub simulation:

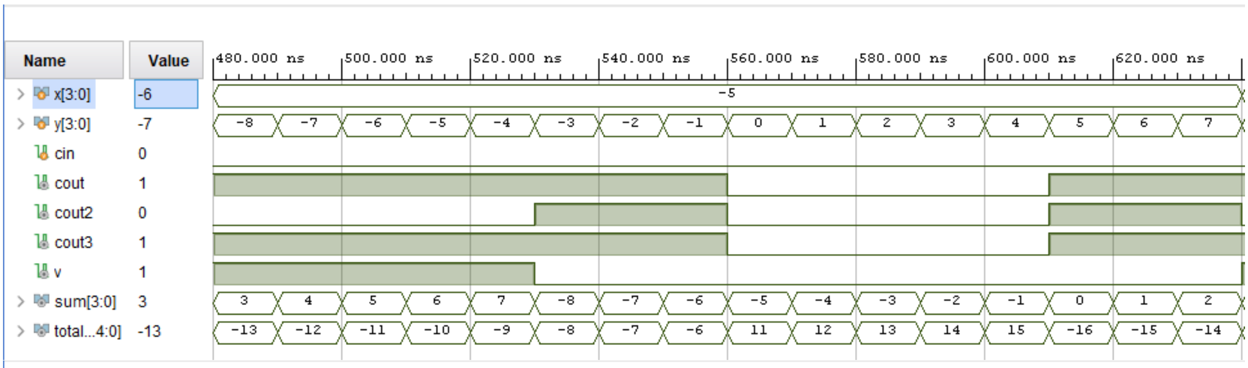
cin=0

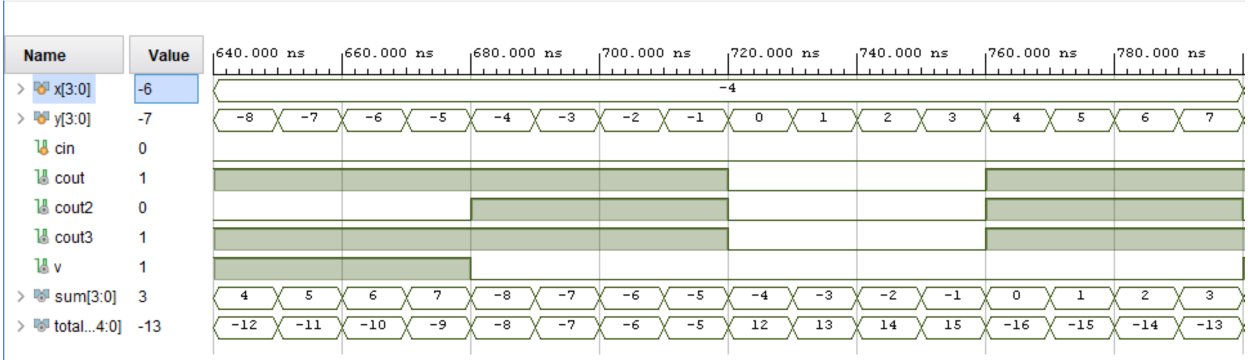


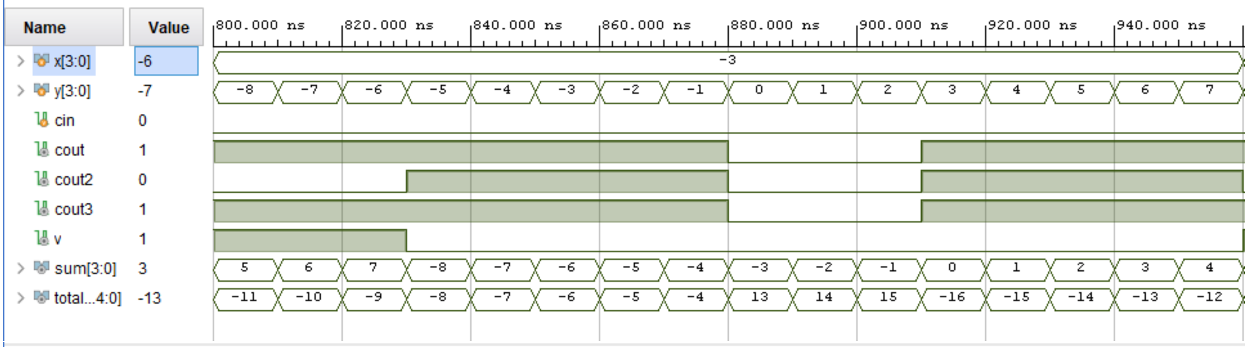
Name	Value
x[3:0]	-8
y[3:0]	-2
cin	0
cout	1
cout2	0
cout3	1
v	1
sum[3:0]	6
total...4:0]	-10

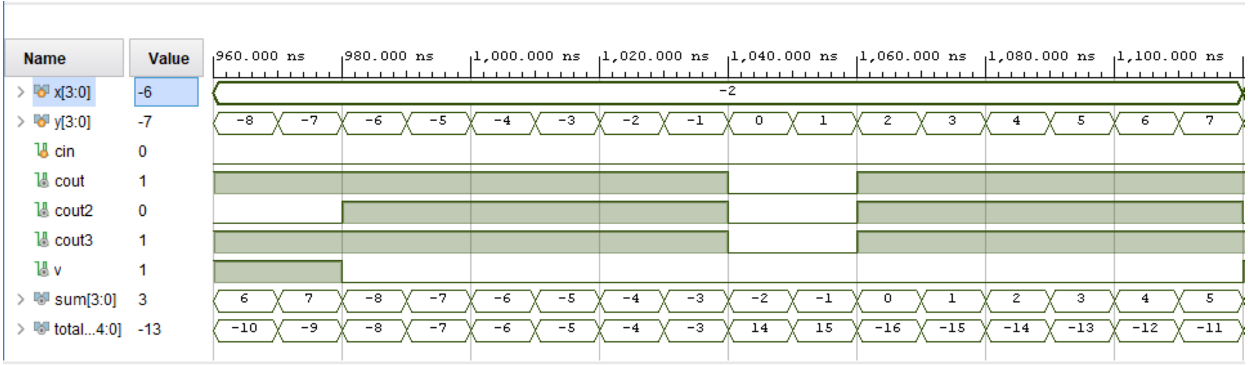
Name	Value
x[3:0]	-7
y[3:0]	-5
cin	0
cout	1
cout2	0
cout3	1
v	1
sum[3:0]	4
total...4:0]	-12

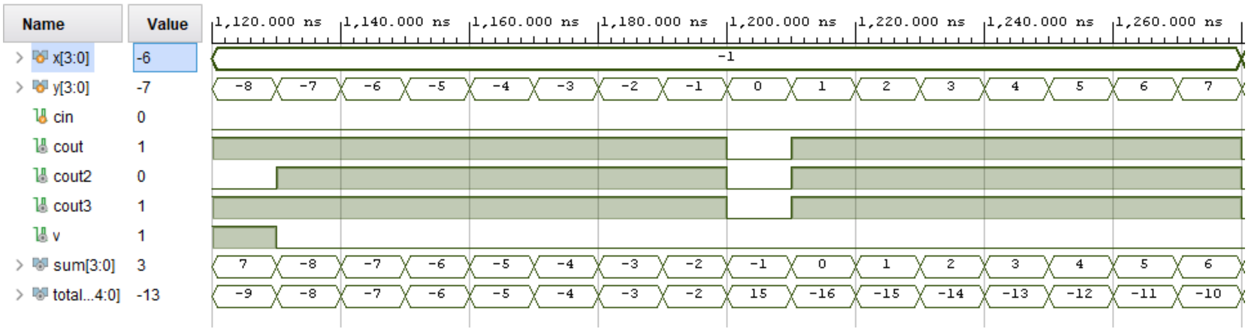
Name	Value
x[3:0]	-6
y[3:0]	-7
cin	0
cout	1
cout2	0
cout3	1
v	1
sum[3:0]	3
total...4:0]	-13

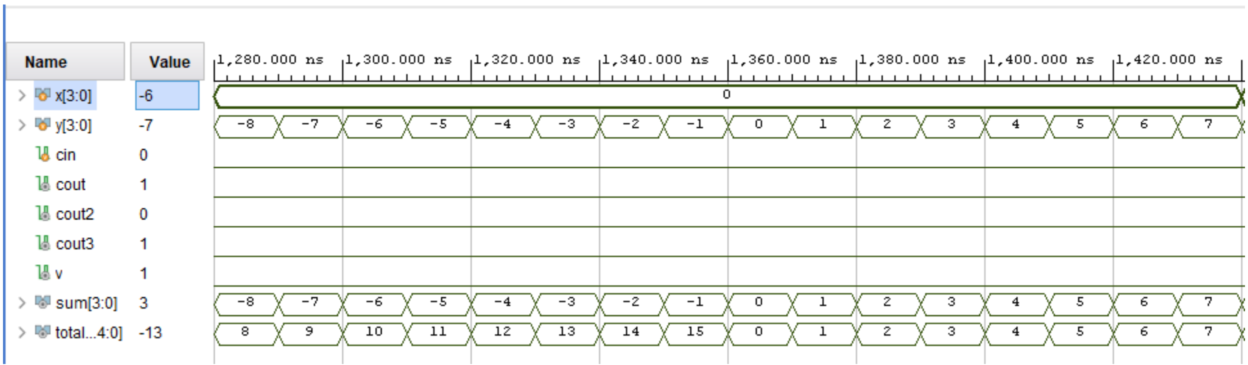

Name
Value
480.000 ns
500.000 ns
520.000 ns
540.000 ns
560.000 ns
580.000 ns
600.000 ns
620.000 ns
x[3:0] -6
y[3:0] -7
cin 0
cout 1
cout2 0
cout3 1
v 1
sum[3:0] 3
total...4:0] -13
-5
-8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7
3 4 5 6 7 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2
-13 -12 -11 -10 -9 -8 -7 -6 11 12 13 14 15 -16 -15 -14

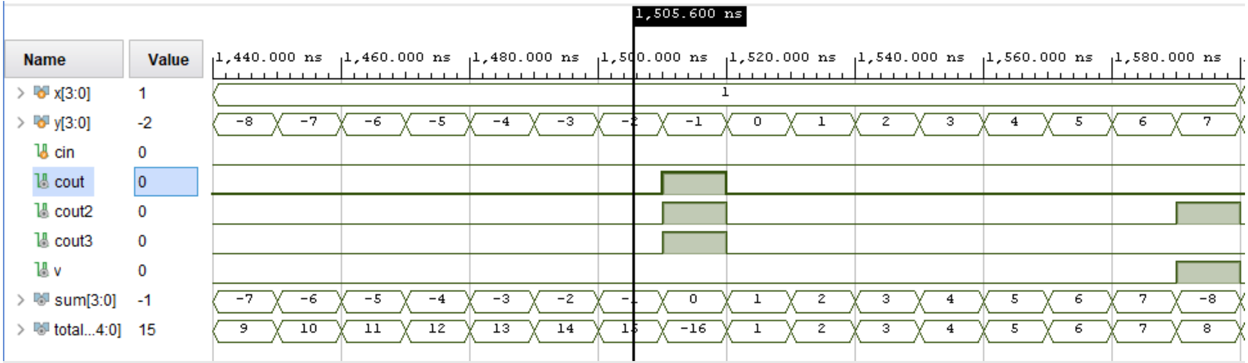

Name
Value
640.000 ns
660.000 ns
680.000 ns
700.000 ns
720.000 ns
740.000 ns
760.000 ns
780.000 ns
x[3:0] -6
y[3:0] -7
cin 0
cout 1
cout2 0
cout3 1
v 1
sum[3:0] 3
total...4:0] -13
-4
-8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7
4 5 6 7 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3
-12 -11 -10 -9 -8 -7 -6 -5 12 13 14 15 -16 -15 -14 -13

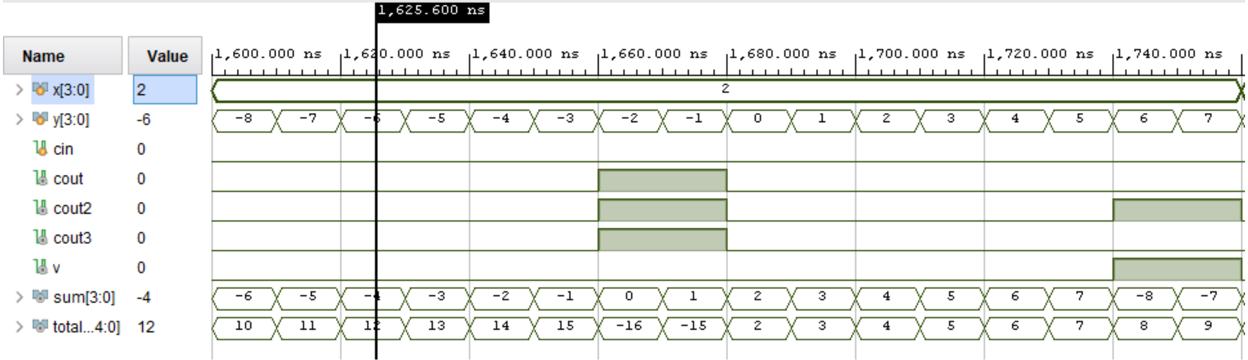

Name
Value
800.000 ns
820.000 ns
840.000 ns
860.000 ns
880.000 ns
900.000 ns
920.000 ns
940.000 ns
x[3:0] -6
y[3:0] -7
cin 0
cout 1
cout2 0
cout3 1
v 1
sum[3:0] 3
total...4:0] -13
-3
-8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7
5 6 7 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4
-11 -10 -9 -8 -7 -6 -5 -4 13 14 15 -16 -15 -14 -13 -12

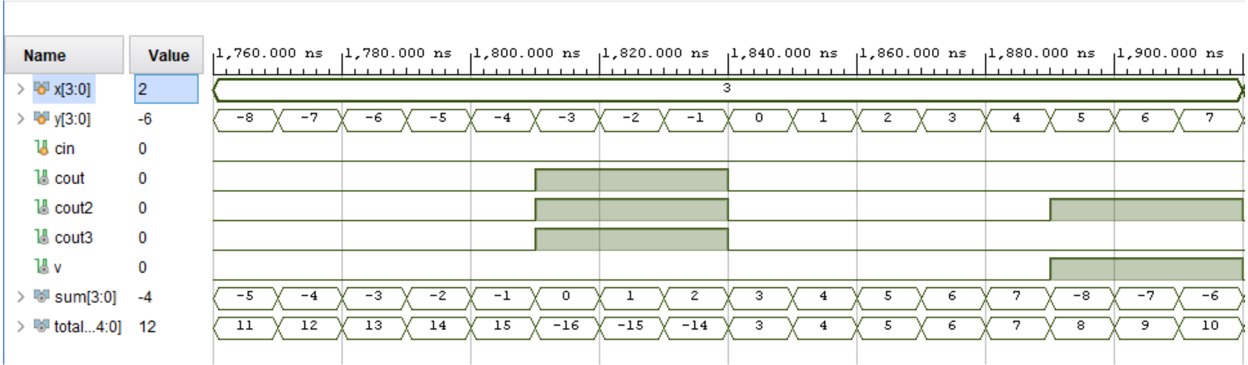

Name
Value
960.000 ns
980.000 ns
1,000.000 ns
1,020.000 ns
1,040.000 ns
1,060.000 ns
1,080.000 ns
1,100.000 ns
x[3:0] -6
y[3:0] -7
cin 0
cout 1
cout2 0
cout3 1
v 1
sum[3:0] 3
total...4:0] -13
-2
-8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7
6 7 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5
-10 -9 -8 -7 -6 -5 -4 -3 14 15 -16 -15 -14 -13 -12 -11

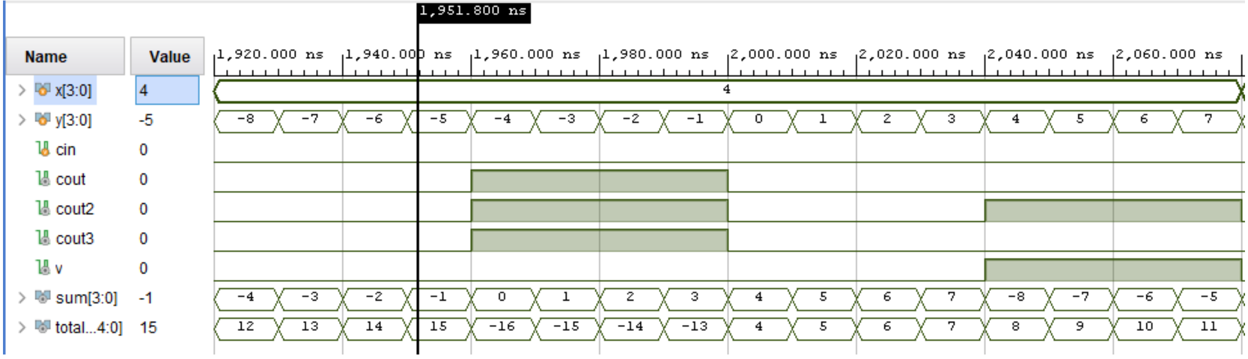

Name
Value
1,120.000 ns
1,140.000 ns
1,160.000 ns
1,180.000 ns
1,200.000 ns
1,220.000 ns
1,240.000 ns
1,260.000 ns
x[3:0] -6
y[3:0] -7
cin 0
cout 1
cout2 0
cout3 1
v 1
sum[3:0] 3
total...4:0] -13
-1
-8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7
7 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6
-9 -8 -7 -6 -5 -4 -3 -2 15 -16 -15 -14 -13 -12 -11 -10

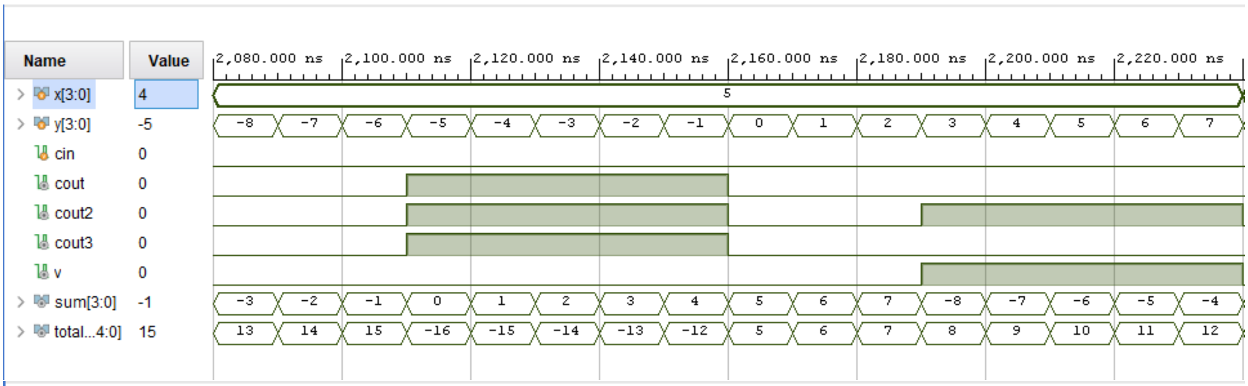

Name
Value
x[3:0] -6
y[3:0] -7
cin 0
cout 1
cout2 0
cout3 1
v 1
sum[3:0] 3
total...4:0] -13
1,280.000 ns
1,300.000 ns
1,320.000 ns
1,340.000 ns
1,360.000 ns
1,380.000 ns
1,400.000 ns
1,420.000 ns


1,505.600 ns
Name
Value
x[3:0] 1
y[3:0] -2
cin 0
cout 0
cout2 0
cout3 0
v 0
sum[3:0] -1
total...4:0] 15
1,440.000 ns
1,460.000 ns
1,480.000 ns
1,500.000 ns
1,520.000 ns
1,540.000 ns
1,560.000 ns
1,580.000 ns

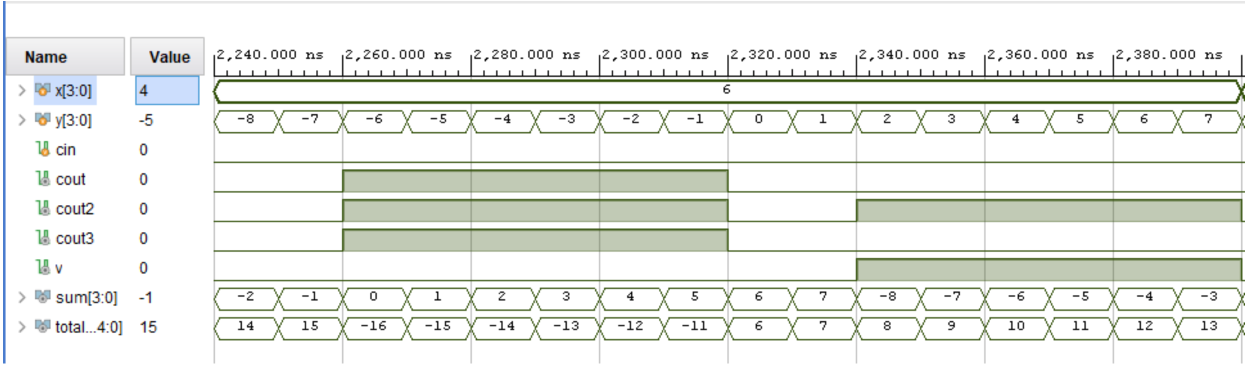

1,625.600 ns
Name
Value
x[3:0] 2
y[3:0] -6
cin 0
cout 0
cout2 0
cout3 0
v 0
sum[3:0] -4
total...4:0] 12
1,600.000 ns
1,620.000 ns
1,640.000 ns
1,660.000 ns
1,680.000 ns
1,700.000 ns
1,720.000 ns
1,740.000 ns


Name
Value
x[3:0] 2
y[3:0] -6
cin 0
cout 0
cout2 0
cout3 0
v 0
sum[3:0] -4
total...4:0] 12
1,760.000 ns
1,780.000 ns
1,800.000 ns
1,820.000 ns
1,840.000 ns
1,860.000 ns
1,880.000 ns
1,900.000 ns

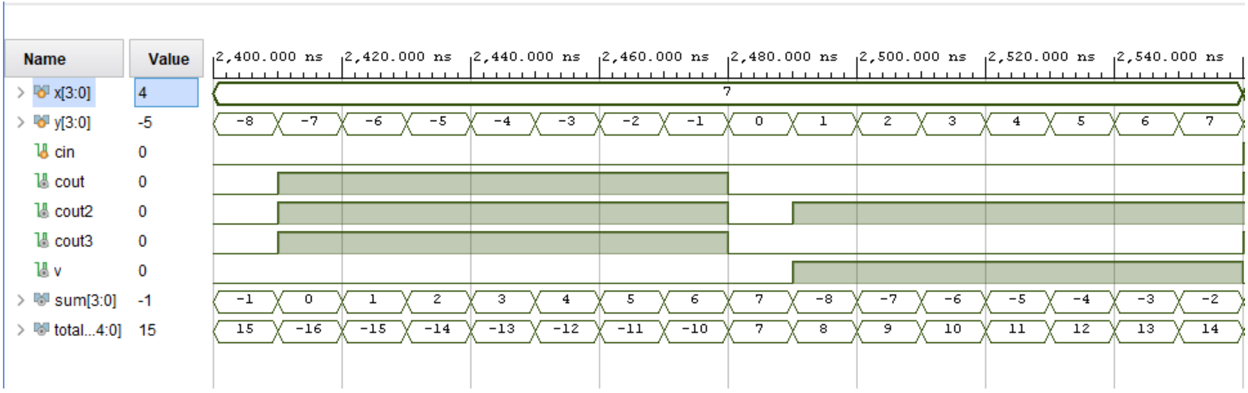

1,951.800 ns
Name
Value
x[3:0] 4
y[3:0] -5
cin 0
cout 0
cout2 0
cout3 0
v 0
sum[3:0] -1
total...4:0] 15
1,920.000 ns
1,940.000 ns
1,960.000 ns
1,980.000 ns
2,000.000 ns
2,020.000 ns
2,040.000 ns
2,060.000 ns



Name	Value
x[3:0]	4
y[3:0]	-5
cin	0
cout	0
cout2	0
cout3	0
v	0
sum[3:0]	-1
total...4:0]	15

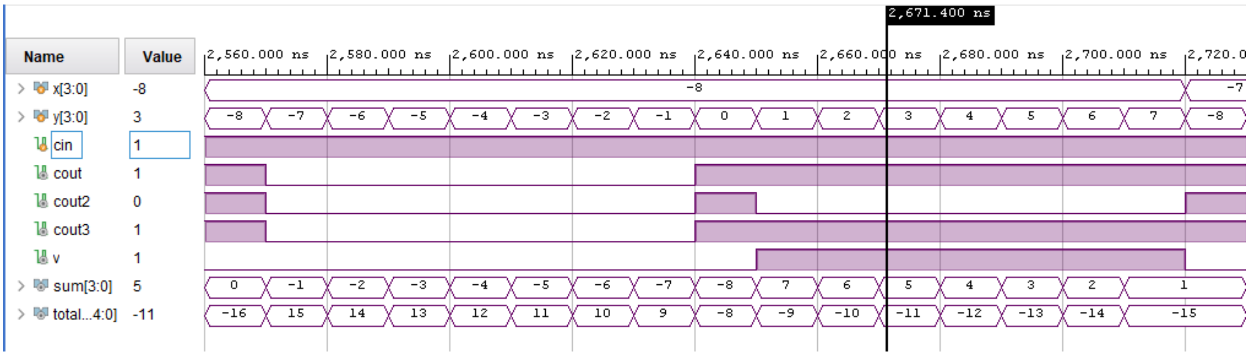


Name	Value
x[3:0]	4
y[3:0]	-5
cin	0
cout	0
cout2	0
cout3	0
v	0
sum[3:0]	-1
total...4:0]	15

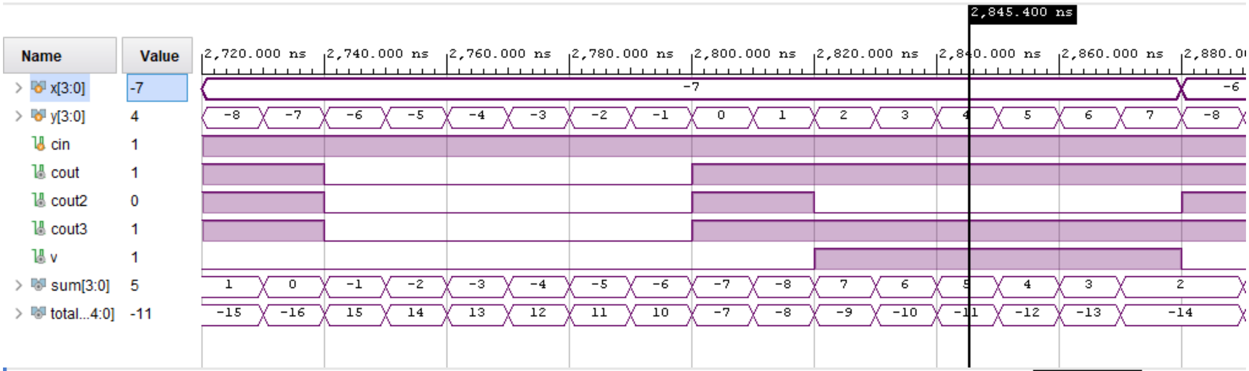


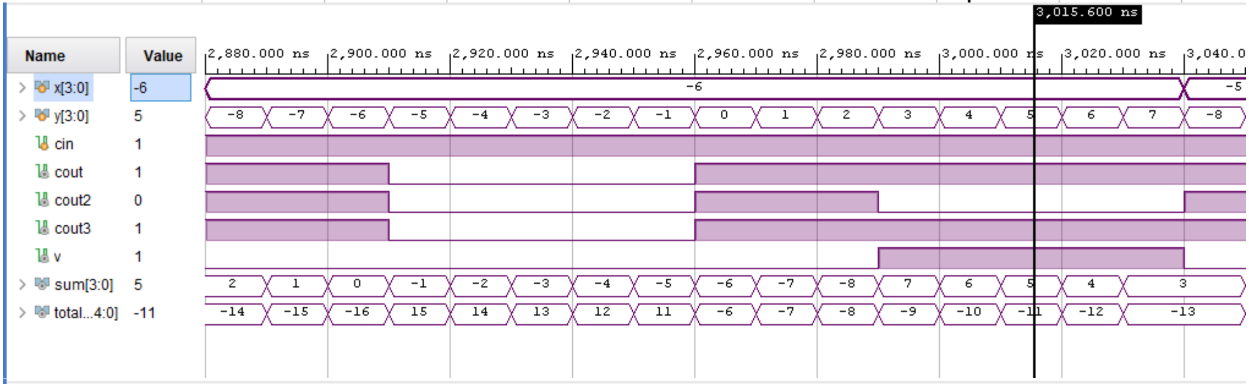
Name	Value
x[3:0]	4
y[3:0]	-5
cin	0
cout	0
cout2	0
cout3	0
v	0
sum[3:0]	-1
total...4:0]	15

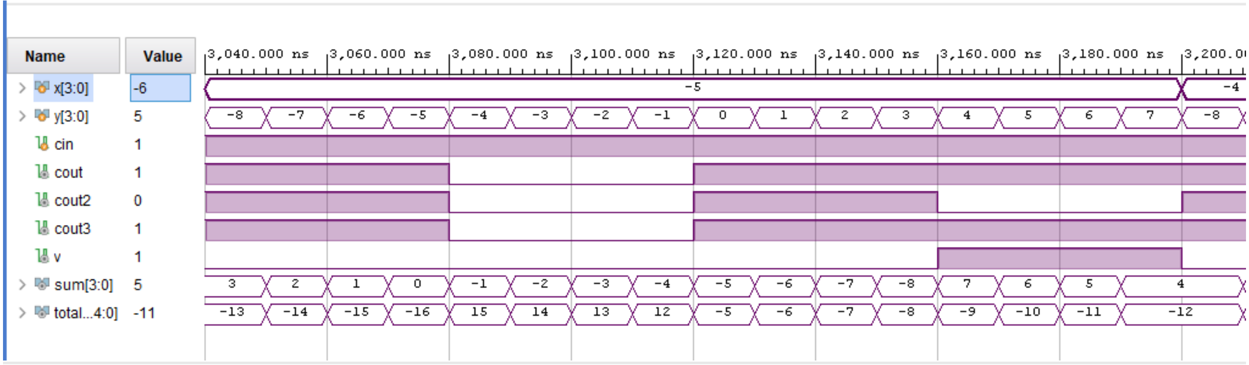
Now Cin=1 so i changed colors to read should realize something changed.

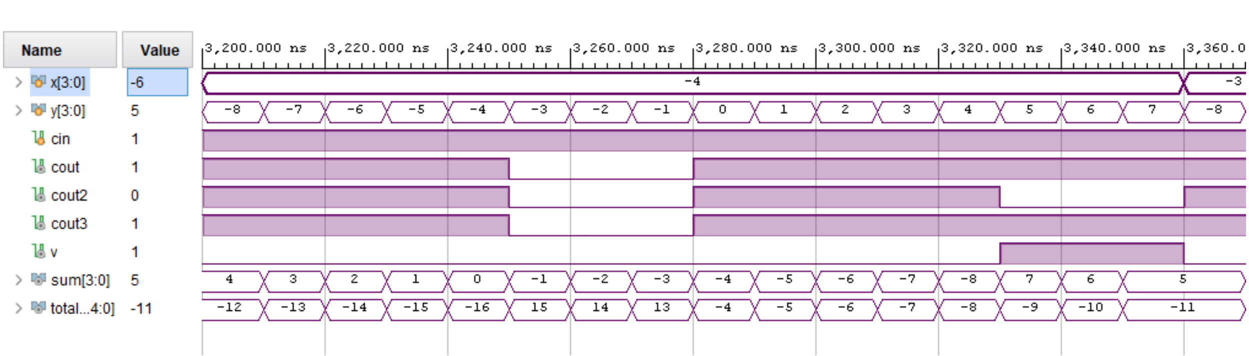


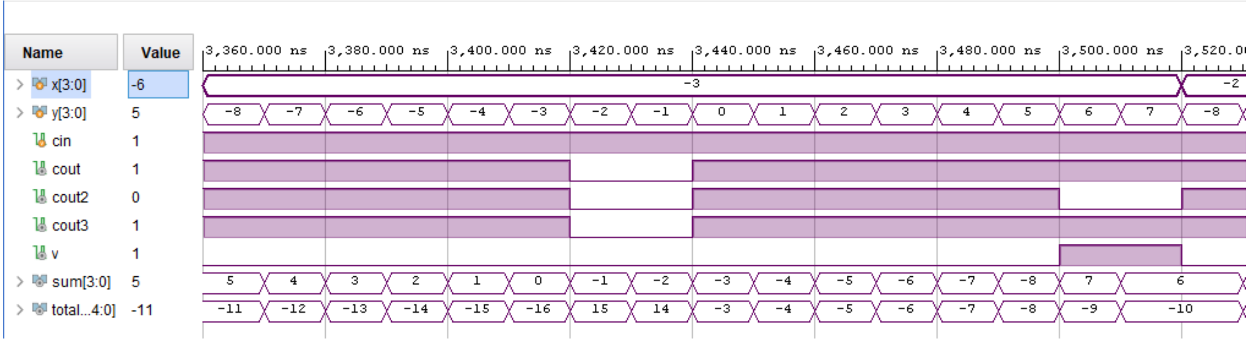
Name	Value
x[3:0]	-8
y[3:0]	3
cin	1
cout	1
cout2	0
cout3	1
v	1
sum[3:0]	5
total...4:0]	-11

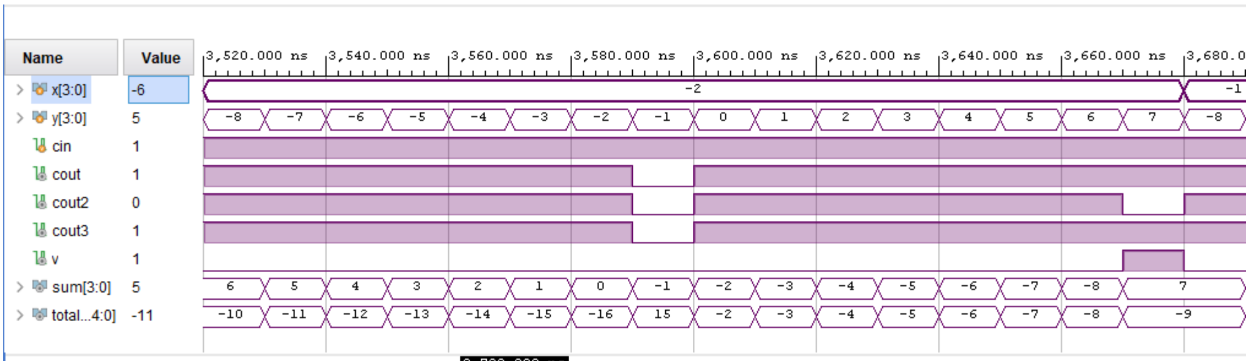

2,845.400 ns
Name
Value
x[3:0]
-7
y[3:0]
4
cin
1
cout
1
cout2
0
cout3
1
v
1
sum[3:0]
5
total...4:0]
-11
2,720.000 ns
2,740.000 ns
2,760.000 ns
2,780.000 ns
2,800.000 ns
2,820.000 ns
2,840.000 ns
2,860.000 ns
2,880.0
-7 -6
-8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 -8
1 0 -1 -2 -3 -4 -5 -6 -7 -8 7 6 5 4 3 2
-15 -16 15 14 13 12 11 10 -7 -8 -9 -10 -11 -12 -13 -14


3,015.600 ns
Name
Value
x[3:0]
-6
y[3:0]
5
cin
1
cout
1
cout2
0
cout3
1
v
1
sum[3:0]
5
total...4:0]
-11
2,880.000 ns
2,900.000 ns
2,920.000 ns
2,940.000 ns
2,960.000 ns
2,980.000 ns
3,000.000 ns
3,020.000 ns
3,040.0
-6 -5
-8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 -8
2 1 0 -1 -2 -3 -4 -5 -6 -7 -8 7 6 5 4 3
-14 -15 -16 15 14 13 12 11 -6 -7 -8 -9 -10 -11 -12 -13

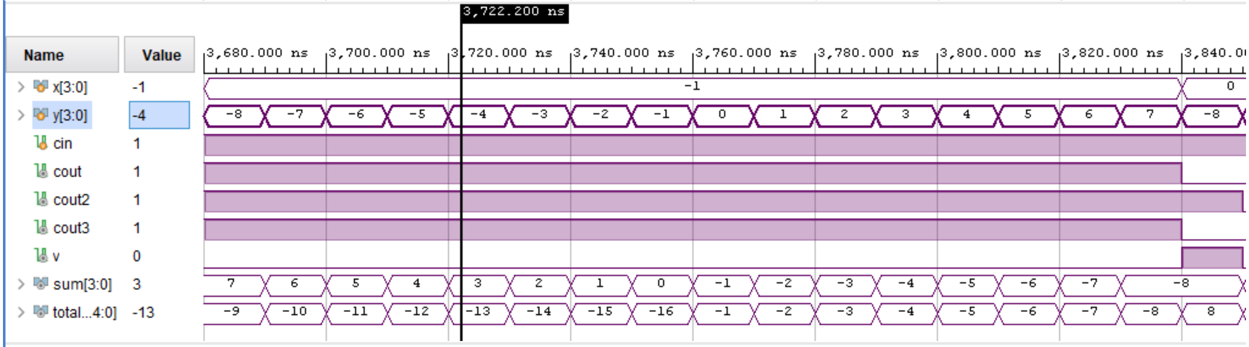

Name
Value
x[3:0]
-6
y[3:0]
5
cin
1
cout
1
cout2
0
cout3
1
v
1
sum[3:0]
5
total...4:0]
-11
3,040.000 ns
3,060.000 ns
3,080.000 ns
3,100.000 ns
3,120.000 ns
3,140.000 ns
3,160.000 ns
3,180.000 ns
3,200.0
-5 -4
-8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 -8
3 2 1 0 -1 -2 -3 -4 -5 -6 -7 -8 7 6 5 4
-13 -14 -15 -16 15 14 13 12 -5 -6 -7 -8 -9 -10 -11 -12


Name
Value
x[3:0]
-6
y[3:0]
5
cin
1
cout
1
cout2
0
cout3
1
v
1
sum[3:0]
5
total...4:0]
-11
3,200.000 ns
3,220.000 ns
3,240.000 ns
3,260.000 ns
3,280.000 ns
3,300.000 ns
3,320.000 ns
3,340.000 ns
3,360.0
-4 -3
-8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 -8
4 3 2 1 0 -1 -2 -3 -4 -5 -6 -7 -8 7 6 5
-12 -13 -14 -15 -16 15 14 13 -4 -5 -6 -7 -8 -9 -10 -11

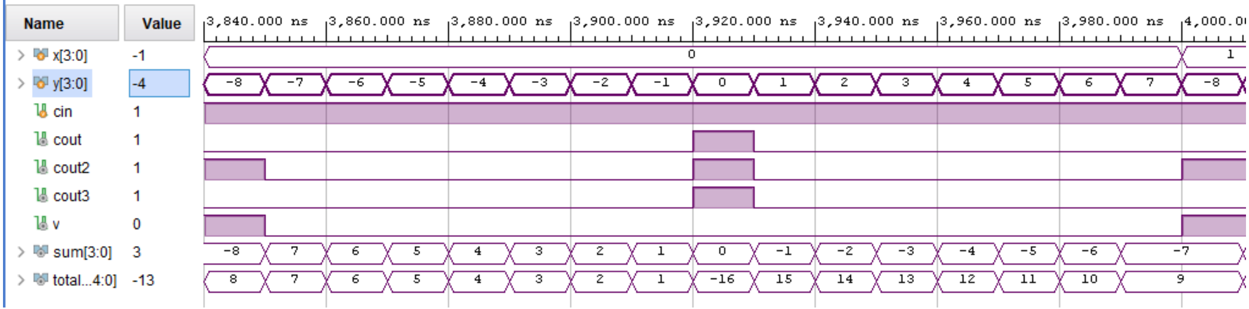

Name
Value
x[3:0]
-6
y[3:0]
5
cin
1
cout
1
cout2
0
cout3
1
v
1
sum[3:0]
5
total...4:0]
-11
3,360.000 ns
3,380.000 ns
3,400.000 ns
3,420.000 ns
3,440.000 ns
3,460.000 ns
3,480.000 ns
3,500.000 ns
3,520.0
-3 -2
-8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 -8
5 4 3 2 1 0 -1 -2 -3 -4 -5 -6 -7 -8 7 6
-11 -12 -13 -14 -15 -16 15 14 -3 -4 -5 -6 -7 -8 -9 -10



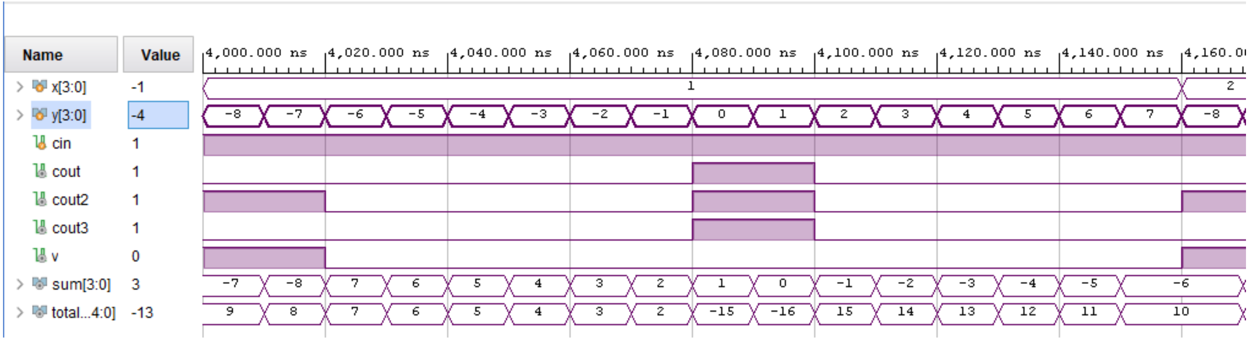
Name	Value
x[3:0]	-6
y[3:0]	5
cin	1
cout	1
cout2	0
cout3	1
v	1
sum[3:0]	5
total...4:0]	-11



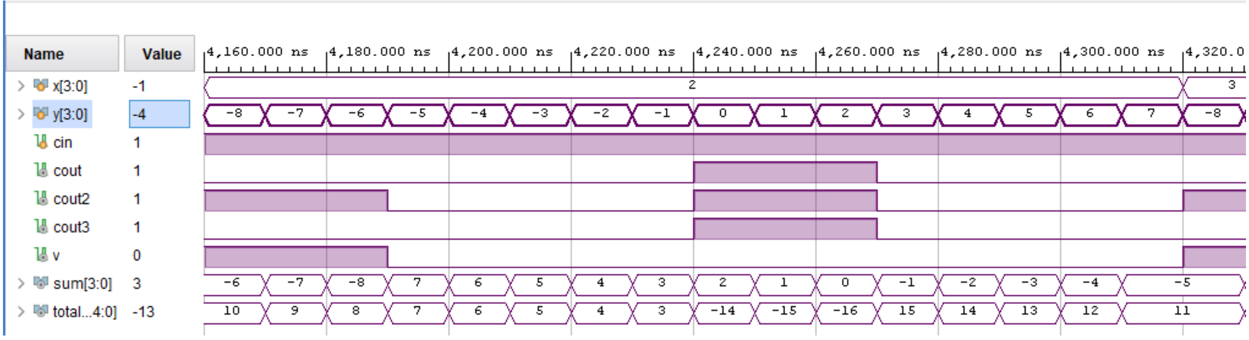
Name	Value
x[3:0]	-1
y[3:0]	-4
cin	1
cout	1
cout2	1
cout3	1
v	0
sum[3:0]	3
total...4:0]	-13



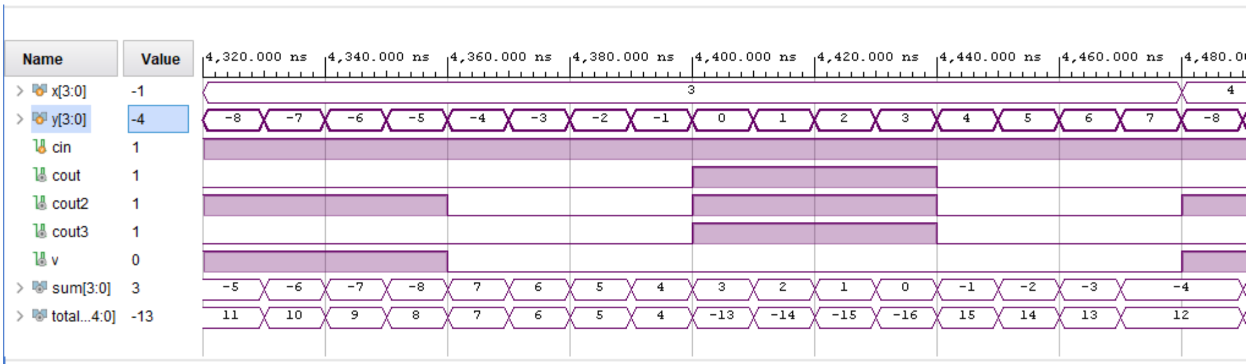
Name	Value
x[3:0]	-1
y[3:0]	-4
cin	1
cout	1
cout2	1
cout3	1
v	0
sum[3:0]	3
total...4:0]	-13

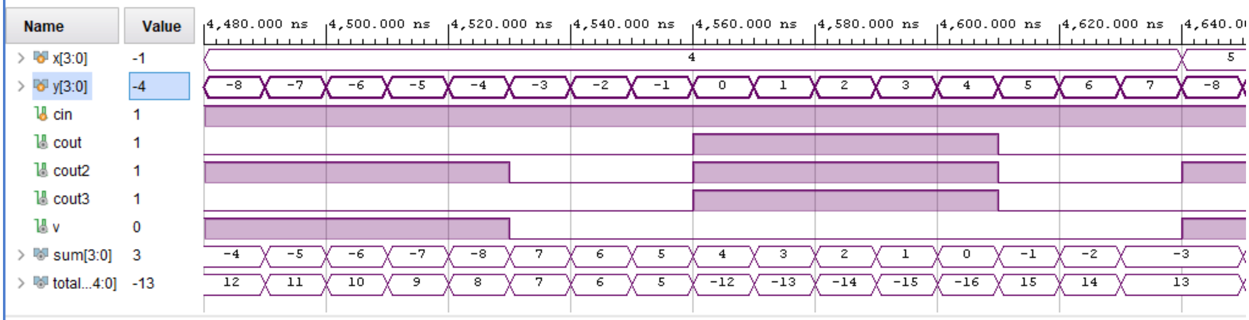


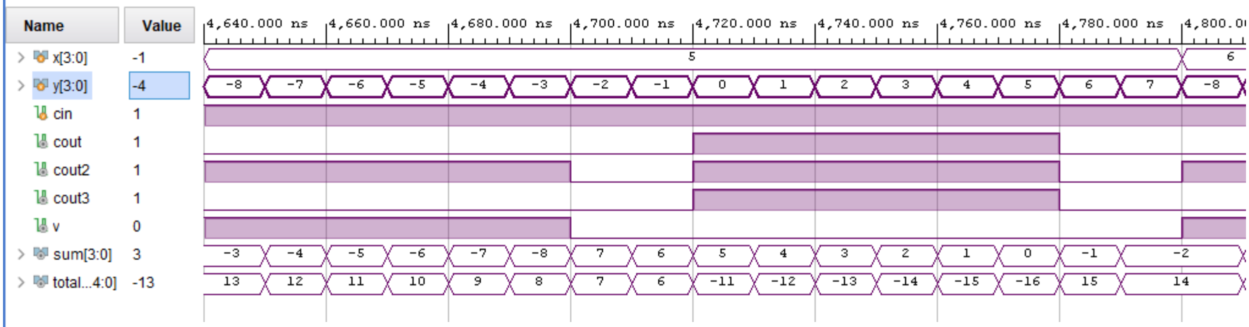
Name	Value
x[3:0]	-1
y[3:0]	-4
cin	1
cout	1
cout2	1
cout3	1
v	0
sum[3:0]	3
total...4:0]	-13

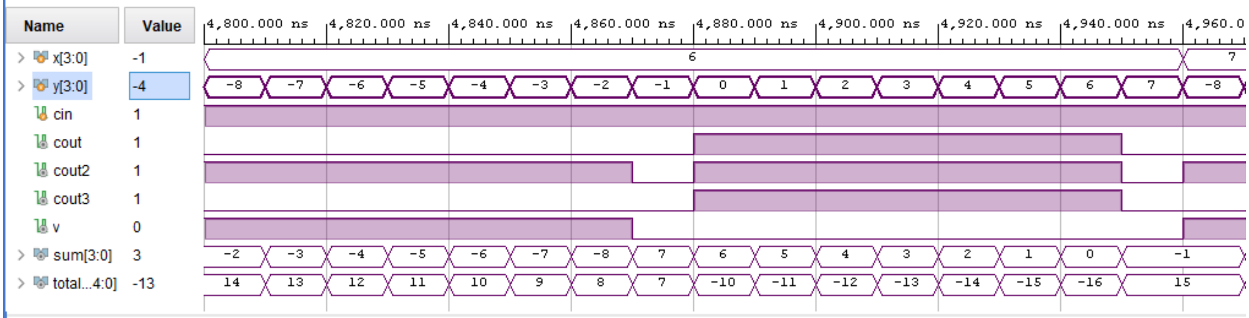


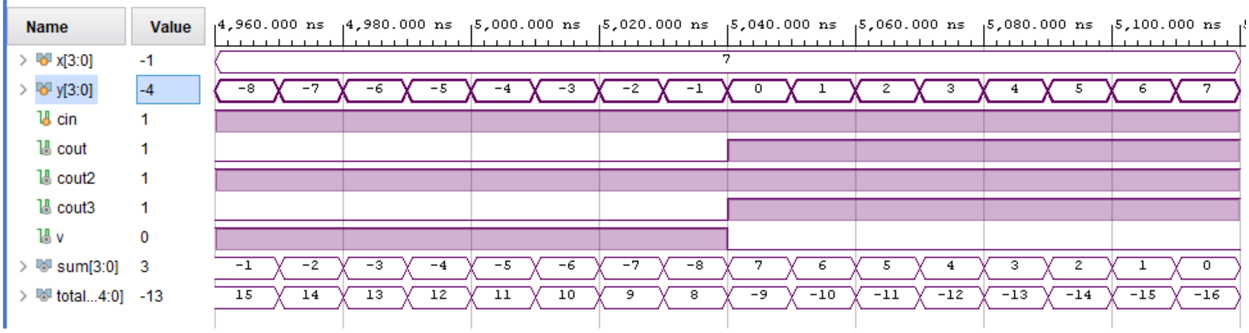
Name	Value
x[3:0]	-1
y[3:0]	-4
cin	1
cout	1
cout2	1
cout3	1
v	0
sum[3:0]	3
total...4:0]	-13


Name
Value
4,320.000 ns
4,340.000 ns
4,360.000 ns
4,380.000 ns
4,400.000 ns
4,420.000 ns
4,440.000 ns
4,460.000 ns
x[3:0] -1
3
4
y[3:0] -4
-8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 -8
cin 1
cout 1
cout2 1
cout3 1
v 0
sum[3:0] 3
-5 -6 -7 -8 7 6 5 4 3 2 1 0 -1 -2 -3 -4
total...4:0] -13
11 10 9 8 7 6 5 4 -13 -14 -15 -16 15 14 13 12

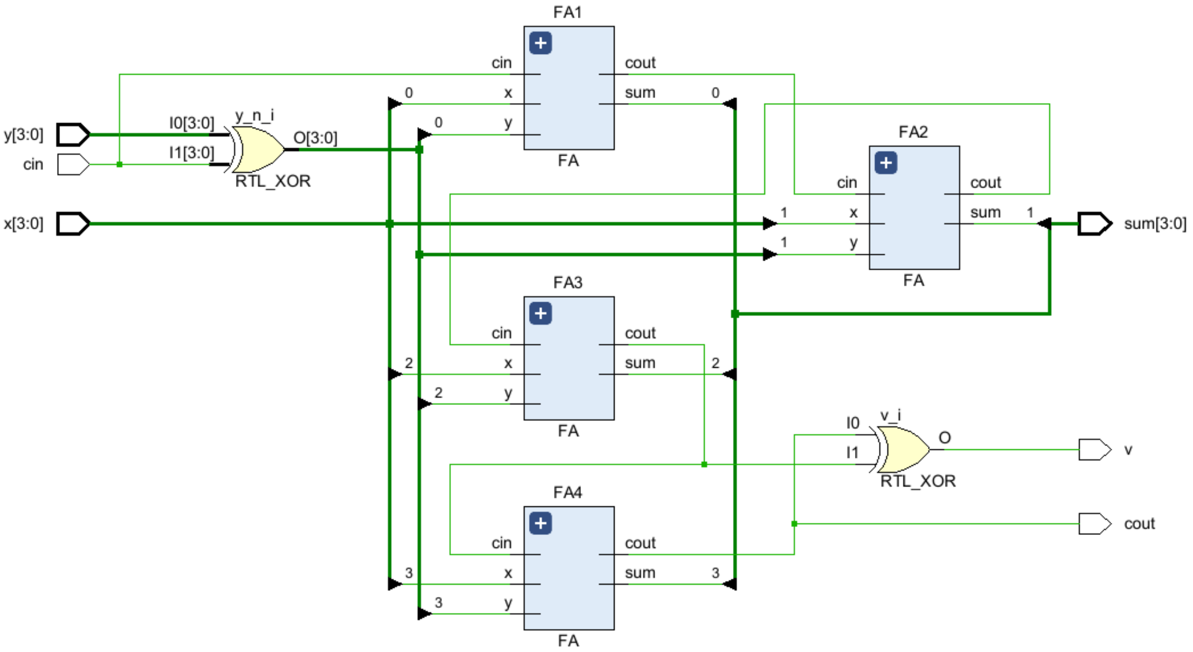

Name
Value
4,480.000 ns
4,500.000 ns
4,520.000 ns
4,540.000 ns
4,560.000 ns
4,580.000 ns
4,600.000 ns
4,620.000 ns
x[3:0] -1
4
5
y[3:0] -4
-8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 -8
cin 1
cout 1
cout2 1
cout3 1
v 0
sum[3:0] 3
-4 -5 -6 -7 -8 7 6 5 4 3 2 1 0 -1 -2 -3
total...4:0] -13
12 11 10 9 8 7 6 5 -12 -13 -14 -15 -16 15 14 13


Name
Value
4,640.000 ns
4,660.000 ns
4,680.000 ns
4,700.000 ns
4,720.000 ns
4,740.000 ns
4,760.000 ns
4,780.000 ns
x[3:0] -1
5
6
y[3:0] -4
-8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 -8
cin 1
cout 1
cout2 1
cout3 1
v 0
sum[3:0] 3
-3 -4 -5 -6 -7 -8 7 6 5 4 3 2 1 0 -1 -2
total...4:0] -13
13 12 11 10 9 8 7 6 -11 -12 -13 -14 -15 -16 15 14

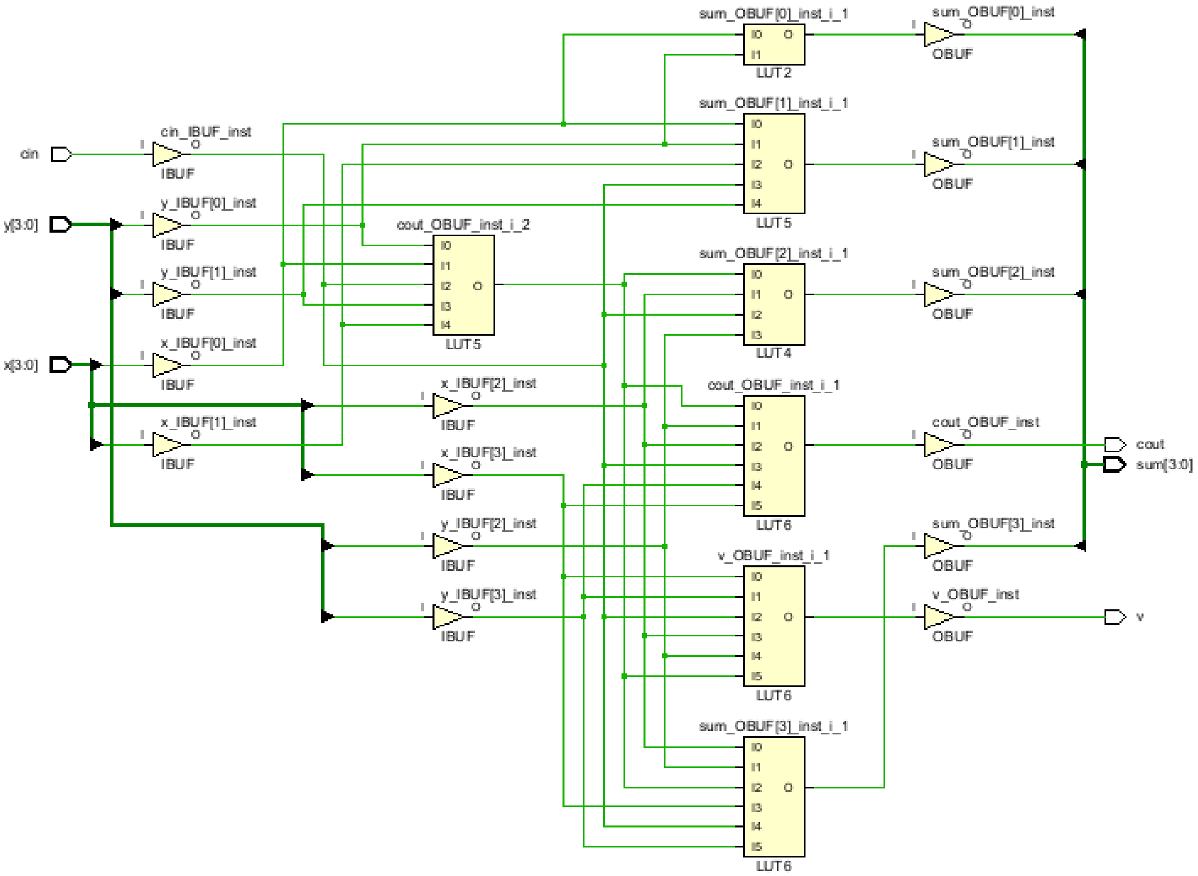

Name
Value
4,800.000 ns
4,820.000 ns
4,840.000 ns
4,860.000 ns
4,880.000 ns
4,900.000 ns
4,920.000 ns
4,940.000 ns
x[3:0] -1
6
7
y[3:0] -4
-8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 -8
cin 1
cout 1
cout2 1
cout3 1
v 0
sum[3:0] 3
-2 -3 -4 -5 -6 -7 -8 7 6 5 4 3 2 1 0 -1
total...4:0] -13
14 13 12 11 10 9 8 7 -10 -11 -12 -13 -14 -15 -16 15


Name
Value
4,960.000 ns
4,980.000 ns
5,000.000 ns
5,020.000 ns
5,040.000 ns
5,060.000 ns
5,080.000 ns
5,100.000 ns
x[3:0] -1
7
y[3:0] -4
-8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7
cin 1
cout 1
cout2 1
cout3 1
v 0
sum[3:0] 3
-1 -2 -3 -4 -5 -6 -7 -8 7 6 5 4 3 2 1 0
total...4:0] -13
15 14 13 12 11 10 9 8 -9 -10 -11 -12 -13 -14 -15 -16

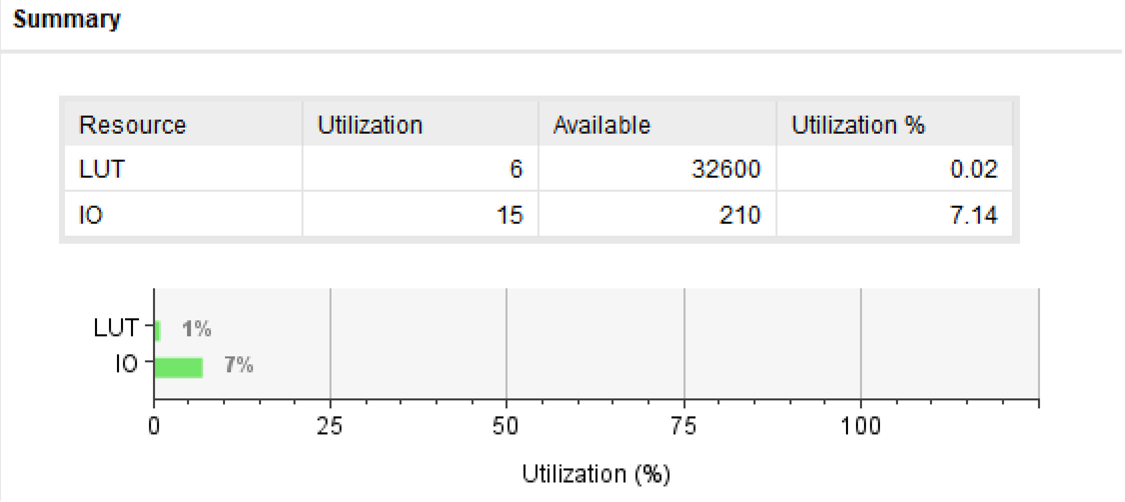
RTL Schematic of Add_Sub:



Technology Schematic of Add_Sub:



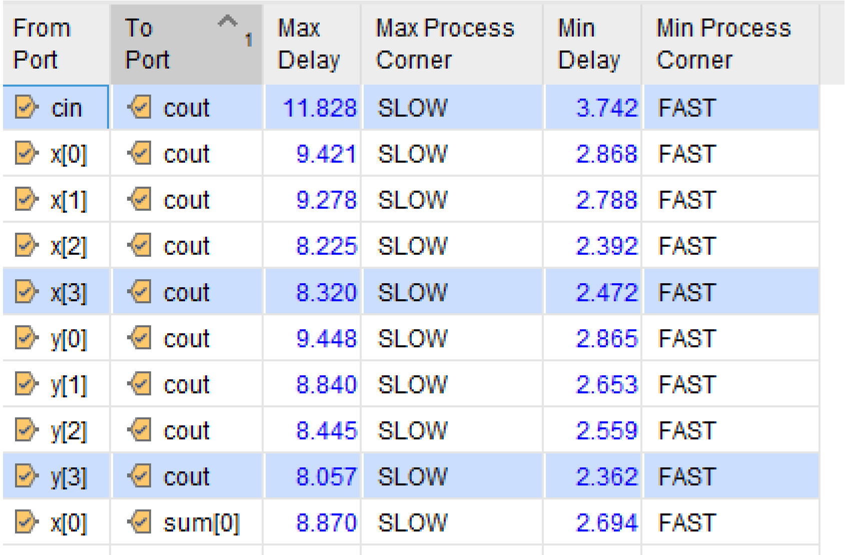
Utilization Report:



Summary

Resource	Utilization	Available	Utilization %
LUT	6	32600	0.02
IO	15	210	7.14

Timing Summary:



From Port	To Port	Max Delay	Max Process Corner	Min Delay	Min Process Corner
cin	cout	11.828	SLOW	3.742	FAST
x[0]	cout	9.421	SLOW	2.868	FAST
x[1]	cout	9.278	SLOW	2.788	FAST
x[2]	cout	8.225	SLOW	2.392	FAST
x[3]	cout	8.320	SLOW	2.472	FAST
y[0]	cout	9.448	SLOW	2.865	FAST
y[1]	cout	8.840	SLOW	2.653	FAST
y[2]	cout	8.445	SLOW	2.559	FAST
y[3]	cout	8.057	SLOW	2.362	FAST
x[0]	sum[0]	8.870	SLOW	2.694	FAST

in CLA
x[3] to cout delay is = 8.818 ns
y[3] to cout delay is = 8.582 ns

in RCA

x[3] to cout delay is = 10.4 ns
y[3] to cout delay is = 10.123 ns

in Add_Subb_with_V

x[3] to cout delay is = 8.328
y[3] to cout delay is = 8.057