Half Adder

I wrote the half adder module with two logic gates with the help of truth table in the experiment document.

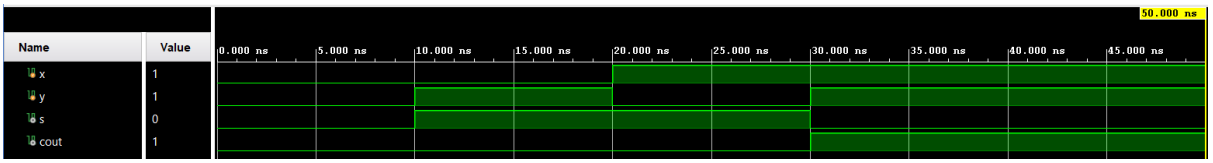
- Verilog Code:

```
module HA(
    input x,
    input y,
    output cout,
    output s
    );

 assign cout = x && y;
 assign s = x ^ y;

endmodule
```

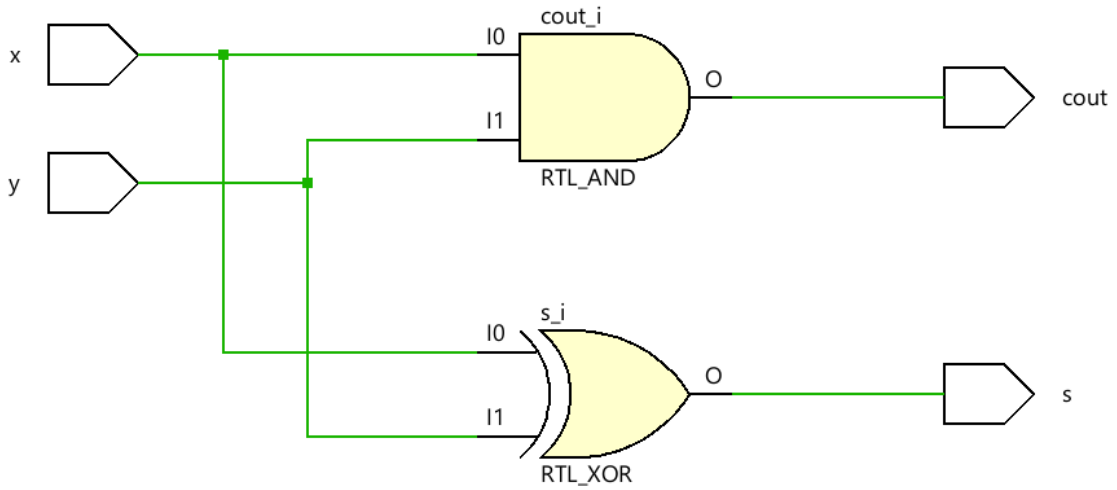
- Testbench Code:

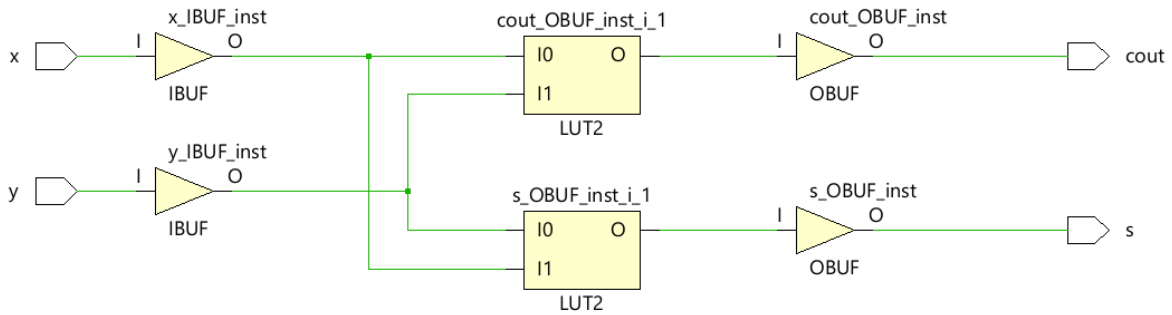
```
module HA_tb;

reg x;
reg y;
wire s;
wire cout;

HA uut(x, y, cout, s);
  initial begin
    for (integer i = 0; i < 4; i = i + 1) begin
      case (i)
        0: {x, y} = 2'b00;
        1: {x, y} = 2'b01;
        2: {x, y} = 2'b10;
        3: {x, y} = 2'b11;
      endcase
      #10;
    end
    #10 $finish;
  end
endmodule
```

- Simulation results for all inputs:



- RTL and Technology Schematics:



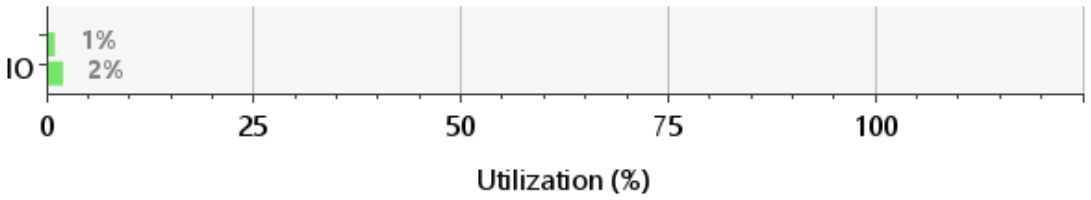


- Time and Utilization Reports:

Maximum combinational delay is 7.007ns and 1 LUT is used.

From Port	To Port	Max Delay	Max Process Corner	Min Delay	Min Process Corner
x	cout	6.625	SLOW	2.238	FAST
x	s	6.308	SLOW	2.135	FAST
y	cout	7.007	SLOW	2.337	FAST
y	s	6.658	SLOW	2.240	FAST

Resource	Utilization	Available	Utilization %
LUT	1	63400	0.00
IO	4	210	1.90



FULL ADDER

I write my full adder module with using the half adder modules

- Verilog Code:

```
module FA(
    input x,
    input y,
    input ci,
    output cout,
    output s
    );

    (*DONT_TOUCH = "TRUE"*) wire sum0, cout0, cout1;

    HA ha0(x, y, cout0, sum0);
    HA ha1(sum0, ci, cout1, s);
    assign cout = cout0 || cout1;

endmodule
```

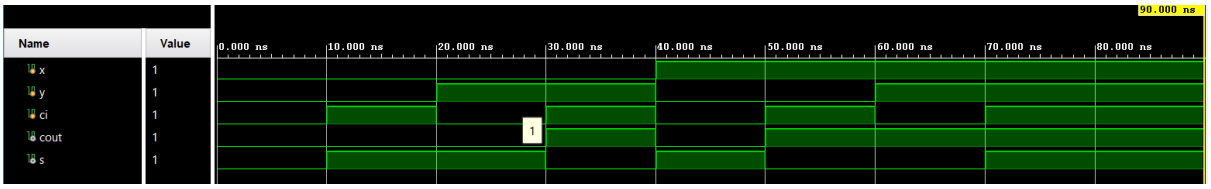
- Testbench Code:

```
module FA_tb;

  reg x, y, ci;
  wire cout, s;

  FA uut (x, y, ci, cout, s);

  initial begin
```

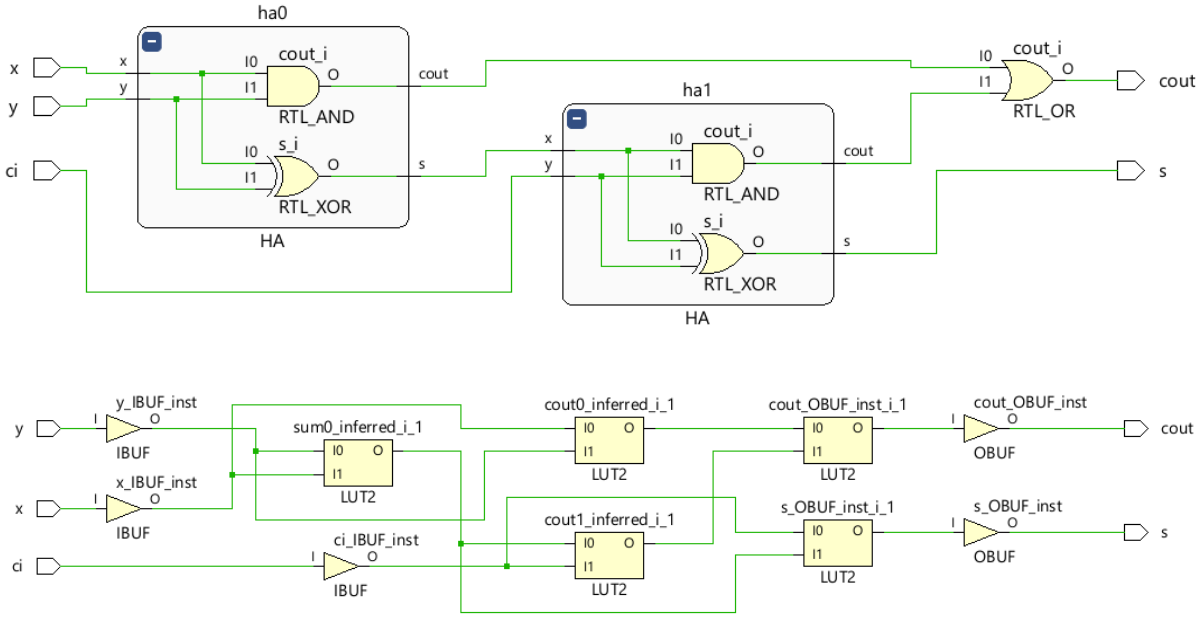
```
    for (integer i = 0; i < 8; i = i + 1) begin
      {x, y, ci} = i;
      #10;
    end
    #10 $finish;
  end

endmodule
```

- **Simulation results:**

all input combinations are used.



- RTL and Technology Schematics:

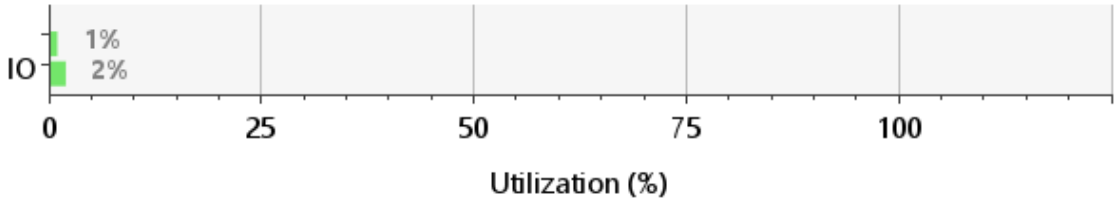


- Timing and Utilization Reports:

Since we used DONT TOUCH constraint there are 5 luts. 2+2 from half adders and 1 for or gate. and maximum combinational delay is increased to 8.178

From Port	To Port	Max Delay	Max Process Corner	Min Delay	Min Process Corner
ci	cout	7.054	SLOW	2.366	FAST
ci	s	6.955	SLOW	2.350	FAST
x	cout	8.178	SLOW	2.431	FAST
x	s	7.520	SLOW	2.570	FAST
y	cout	7.909	SLOW	2.301	FAST
y	s	7.250	SLOW	2.463	FAST

Resource	Utilization	Available	Utilization %
LUT	5	63400	0.01
IO	5	210	2.38



RIPPLE CARRY ADDER

I write my code with using the full adder module.

- Verilog Code

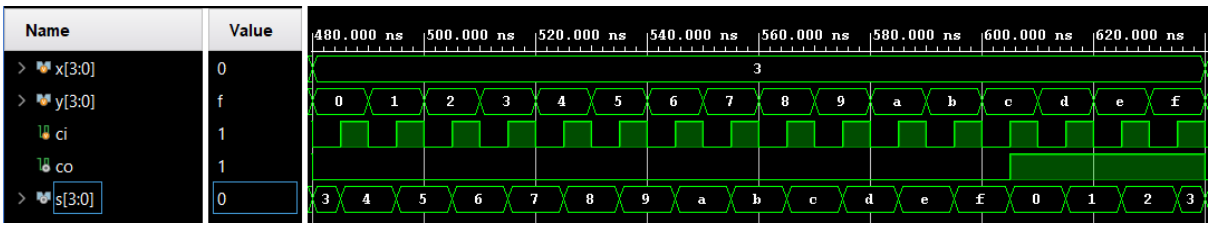
```
module RCA(
    input [3:0] x,
    input [3:0] y,
    input ci,
    output cout,
    output [3:0] s
    );

    (*DONT_TOUCH = "TRUE"*) wire [2:0] faout;
    FA fa0(x[0],y[0],ci,faout[0],s[0]);
    FA fa1(x[1],y[1],faout[0],faout[1],s[1]);
    FA fa2(x[2],y[2],faout[1],faout[2],s[2]);
```

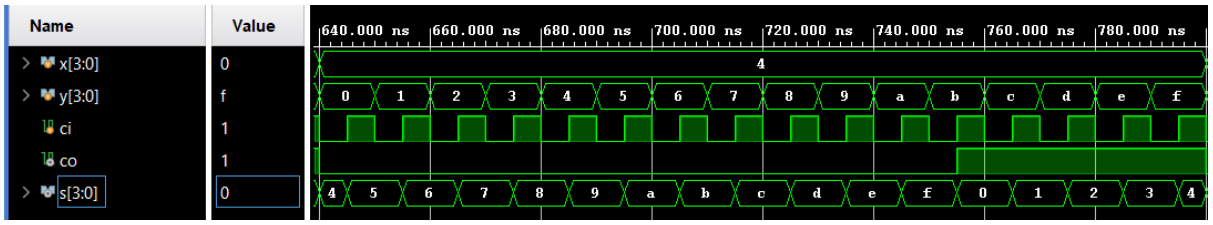
```
    FA fa3(x[3],y[3],faout[2],cout,s[3]);

endmodule
```

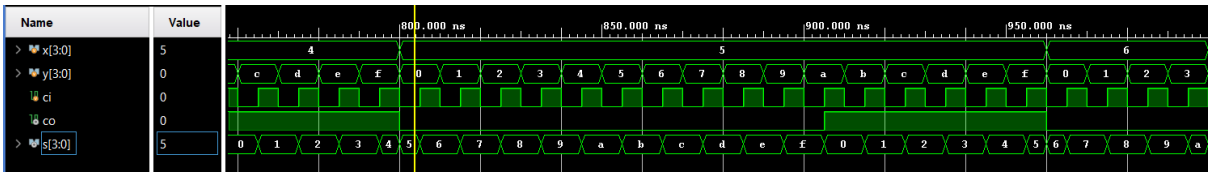
- Testbench Code:

```
module RCA_tb;
  reg [3:0] x, y;
  reg ci;
  wire co;
  wire [3:0] s;
  RCA uut (x, y, ci, co, s);
  initial begin
    for (integer i = 0; i < 511; i = i + 1) begin
      {x, y, ci} = i;
      #5;
    end
    #10 $finish;
  end

endmodule
```

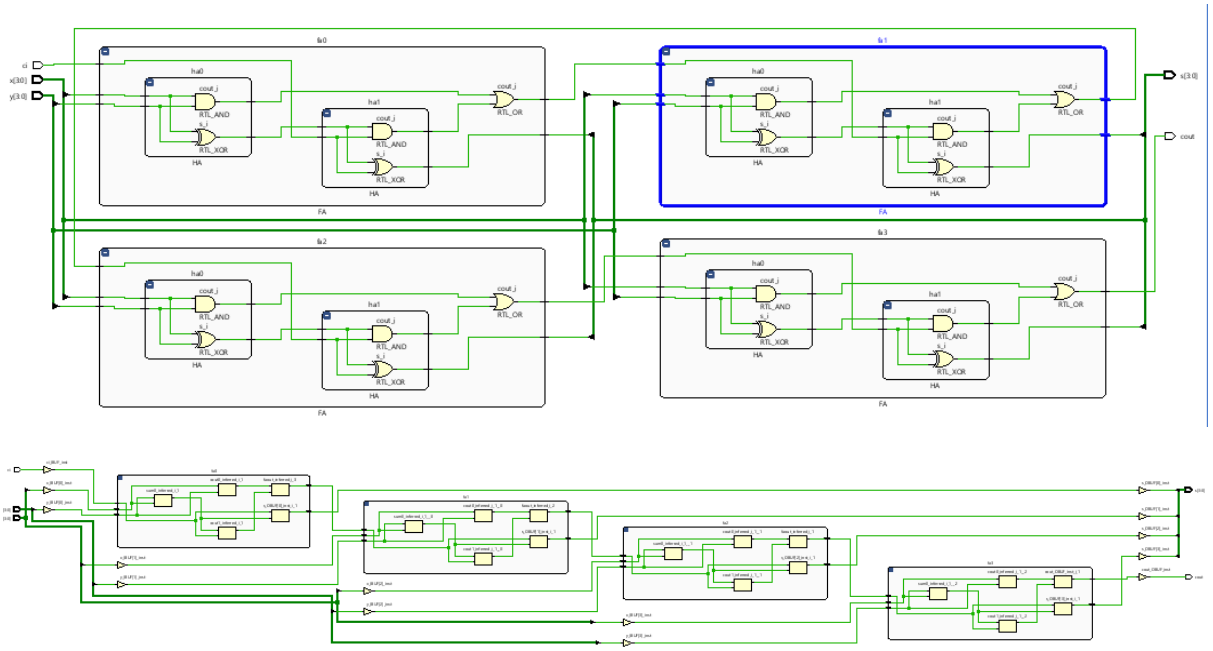
- Simulation results for all input combinations:







- RTL and technology Schematics:



- Timing and utilization reports:

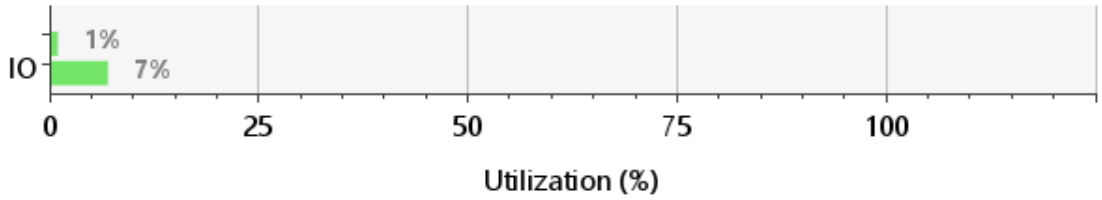
Maximum combinational delay is 15.792 and 19 LUTs are used in this circuit.

From Port	To Port	Ma 1	Max Process Corner	Min Delay	Min Process Corner
ci	cout	15.792	SLOW	5.747	FAST
ci	s[3]	15.097	SLOW	5.517	FAST
ci	s[2]	14.469	SLOW	5.314	FAST
y[0]	cout	13.314	SLOW	4.213	FAST
x[0]	cout	13.262	SLOW	4.072	FAST
ci	s[1]	12.774	SLOW	4.686	FAST
y[0]	s[3]	12.619	SLOW	3.982	FAST
x[0]	s[3]	12.567	SLOW	3.841	FAST
y[1]	cout	12.438	SLOW	3.674	FAST
x[1]	cout	12.429	SLOW	3.542	FAST
y[0]	s[2]	11.991	SLOW	3.780	FAST
x[0]	s[2]	11.939	SLOW	3.638	FAST
ci	s[0]	11.891	SLOW	4.400	FAST
y[1]	s[3]	11.744	SLOW	3.444	FAST
x[1]	s[3]	11.735	SLOW	3.312	FAST

From Port	To Port	Ma 1	Max Process Corner	Min Delay	Min Process Corner
y[2]	cout	11.242	SLOW	3.348	FAST
y[1]	s[2]	11.115	SLOW	3.241	FAST
x[1]	s[2]	11.106	SLOW	3.109	FAST
y[2]	s[2]	10.692	SLOW	3.412	FAST
y[2]	s[3]	10.547	SLOW	3.117	FAST
x[2]	cout	10.410	SLOW	2.839	FAST
y[0]	s[1]	10.296	SLOW	3.152	FAST
x[0]	s[1]	10.244	SLOW	3.010	FAST
x[0]	s[0]	10.115	SLOW	3.112	FAST
y[0]	s[0]	10.065	SLOW	3.127	FAST
x[2]	s[2]	9.860	SLOW	3.024	FAST
x[2]	s[3]	9.715	SLOW	2.609	FAST
x[3]	cout	9.707	SLOW	2.670	FAST
y[1]	s[1]	9.422	SLOW	2.844	FAST
x[1]	s[1]	9.413	SLOW	2.804	FAST
y[3]	cout	9.010	SLOW	2.660	FAST

x[3]	s[3]	8.903	SLOW	2.689	FAST
y[3]	s[3]	8.207	SLOW	2.452	FAST

Resource	Utilization	Available	Utilization %
LUT	19	63400	0.03
IO	14	210	6.67



Parametric RIPPLE CARRY ADDER

this code is similar to the normal RCA. the difference is we can modify the input bit longs easily and used generate for loop here.

- Verilog Code:

```
module parametric_RCA #(parameter SIZE = 4)(x, y, ci, cout, s);

    input [SIZE-1:0] x;
    input [SIZE-1:0] y;
    input ci;
    output cout;
    output [SIZE-1:0] s;

    (*DONT_TOUCH = "TRUE"*) wire [SIZE:0] faout;
    assign faout[0] = ci;
    assign cout = faout[SIZE];

    genvar i;
    generate
        for(i=0;i<SIZE;i=i+1)begin

            FA fax(x[i],y[i],faout[i],faout[i+1],s[i]);

        end
    endgenerate
endmodule
```

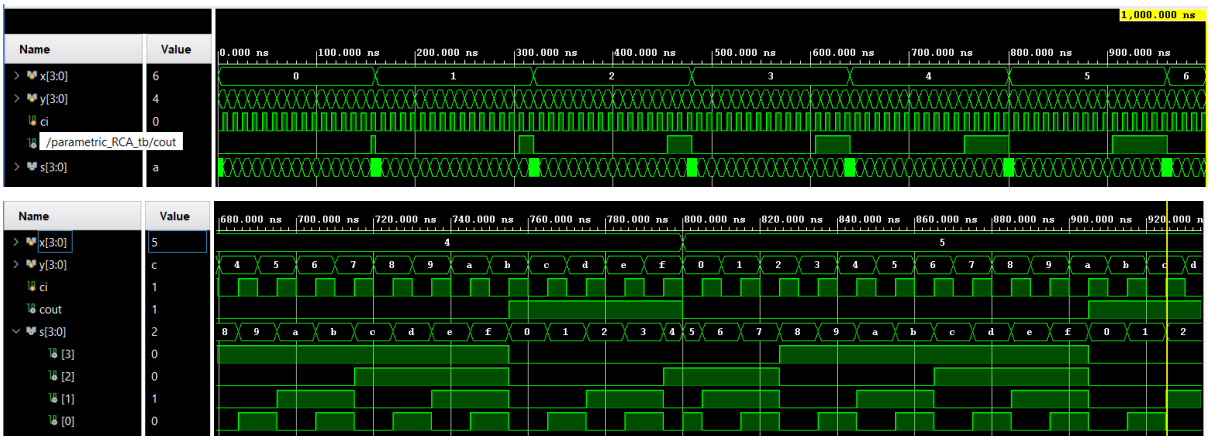
- Testbench Code:

```
module parametric_RCA_tb;

  module parametric_RCA_tb;
  reg [7:0] x, y;
  reg ci;
  wire cout;
  wire [7:0] s;

  parametric_RCA #(.SIZE(4)) uut (
.x(x),.y(y),.ci(ci),.cout(cout),.s(s));

  initial begin
    for (integer i = 0; i < 65000; i = i + 16) begin
```

```
        {x, ci, y} = i;
        #5;
      end
      #10 $finish;
    end

endmodule
```

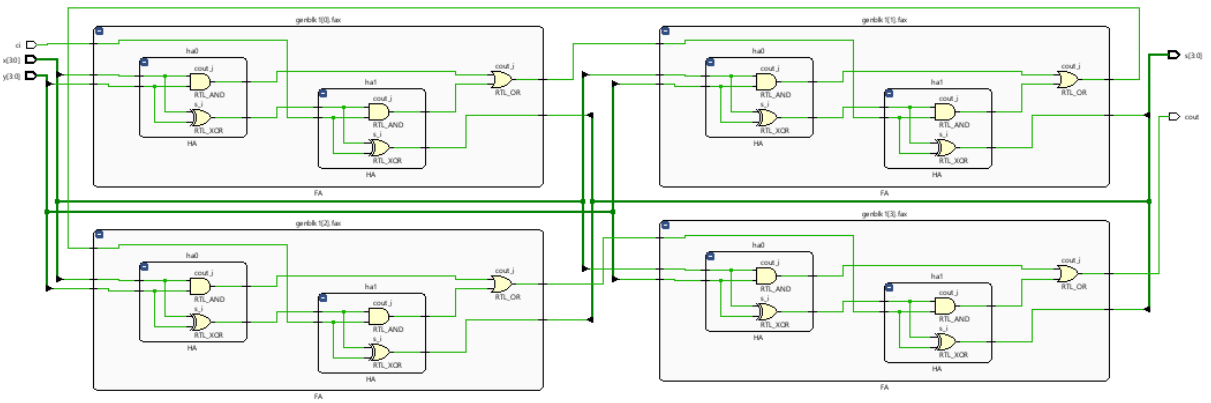
- Simulation results:

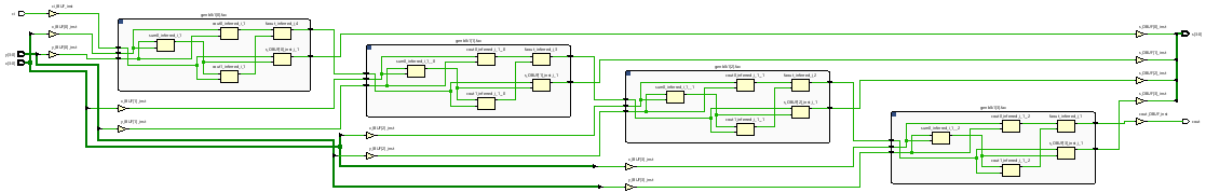
Simulation results are same with the RCA.



- RTL and Technology schematics:

same with the RCA





- Timing and Utilization reports:

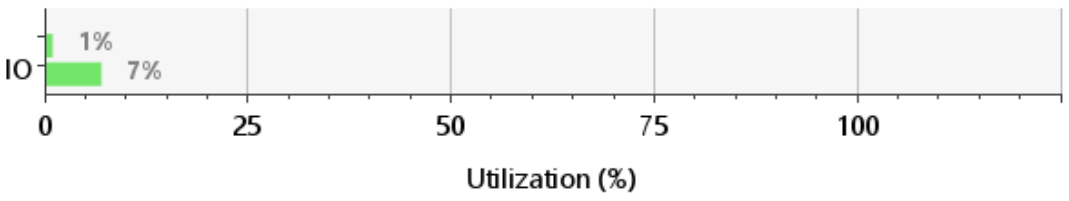
Same with RCA

From Port	To Port	M a 1	Max Process Corner	Min Delay	Min Process Corner
ci	cout	15.792	SLOW	5.747	FAST
ci	s[3]	15.097	SLOW	5.517	FAST
ci	s[2]	14.469	SLOW	5.314	FAST
y[0]	cout	13.314	SLOW	4.213	FAST
x[0]	cout	13.262	SLOW	4.072	FAST
ci	s[1]	12.774	SLOW	4.686	FAST
y[0]	s[3]	12.619	SLOW	3.982	FAST
x[0]	s[3]	12.567	SLOW	3.841	FAST
y[1]	cout	12.438	SLOW	3.674	FAST
x[1]	cout	12.429	SLOW	3.542	FAST
y[0]	s[2]	11.991	SLOW	3.780	FAST
x[0]	s[2]	11.939	SLOW	3.638	FAST
ci	s[0]	11.891	SLOW	4.400	FAST
y[1]	s[3]	11.744	SLOW	3.444	FAST
x[1]	s[3]	11.735	SLOW	3.312	FAST

From Port	To Port	M a 1	Max Process Corner	Min Delay	Min Process Corner
y[2]	cout	11.242	SLOW	3.348	FAST
y[1]	s[2]	11.115	SLOW	3.241	FAST
x[1]	s[2]	11.106	SLOW	3.109	FAST
y[2]	s[2]	10.692	SLOW	3.412	FAST
y[2]	s[3]	10.547	SLOW	3.117	FAST
x[2]	cout	10.410	SLOW	2.839	FAST
y[0]	s[1]	10.296	SLOW	3.152	FAST
x[0]	s[1]	10.244	SLOW	3.010	FAST
x[0]	s[0]	10.115	SLOW	3.112	FAST
y[0]	s[0]	10.065	SLOW	3.127	FAST
x[2]	s[2]	9.860	SLOW	3.024	FAST
x[2]	s[3]	9.715	SLOW	2.609	FAST
x[3]	cout	9.707	SLOW	2.670	FAST
y[1]	s[1]	9.422	SLOW	2.844	FAST
x[1]	s[1]	9.413	SLOW	2.804	FAST

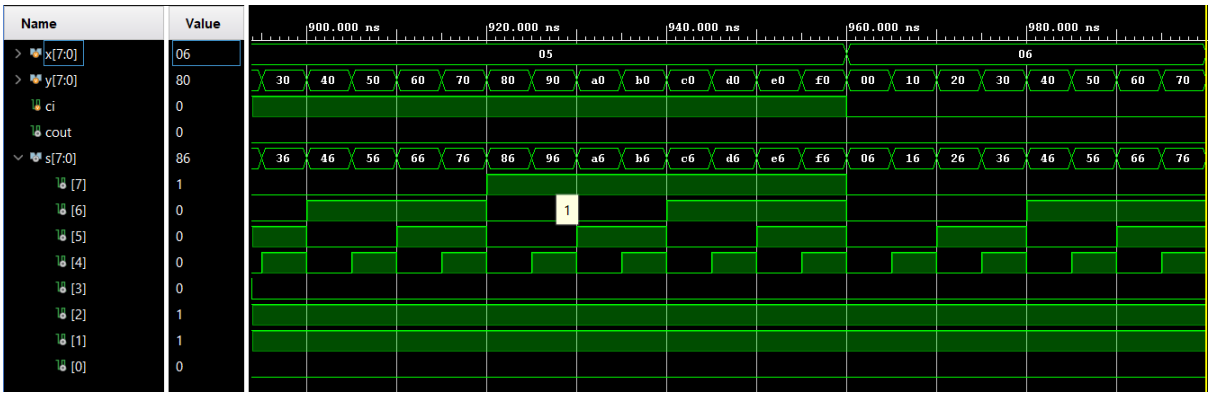
y[3]	cout	9.010	SLOW	2.660	FAST
x[3]	s[3]	8.903	SLOW	2.689	FAST
y[3]	s[3]	8.207	SLOW	2.452	FAST

Resource	Utilization	Available	Utilization %
LUT	19	63400	0.03
IO	14	210	6.67



8bit Ripple Carry Adder

The code for 8 bit carry adder is just a modified version of 4 bit carry adder. The only differentiation is parameter.



Timing And Utilization Reports:
Maximum combinational delay is 24.115ns and 34 LUTs are used in this circuit.

From Port	To Port	Max Delay	Max Process Corner	Min Delay	Min Process Corner
y[0]	s[7]	24.125	SLOW	8.423	FAST
y[0]	cout	24.075	SLOW	8.414	FAST
y[1]	s[7]	22.508	SLOW	7.601	FAST
y[1]	cout	22.457	SLOW	7.592	FAST
y[0]	s[5]	22.279	SLOW	7.760	FAST
y[0]	s[6]	22.004	SLOW	7.853	FAST
y[1]	s[5]	20.661	SLOW	6.939	FAST
y[1]	s[6]	20.387	SLOW	7.031	FAST
x[0]	s[7]	20.214	SLOW	6.510	FAST
x[0]	cout	20.164	SLOW	6.500	FAST
ci	s[7]	20.119	SLOW	6.663	FAST
ci	cout	20.069	SLOW	6.654	FAST
y[0]	s[4]	20.003	SLOW	6.898	FAST
y[2]	s[7]	18.399	SLOW	5.658	FAST
y[1]	s[4]	18.385	SLOW	6.076	FAST
x[0]	s[5]	18.368	SLOW	5.847	FAST
y[2]	cout	18.349	SLOW	5.649	FAST
x[1]	s[7]	18.339	SLOW	5.815	FAST
x[1]	cout	18.289	SLOW	5.805	FAST
ci	s[5]	18.273	SLOW	6.000	FAST
x[0]	s[6]	18.094	SLOW	5.939	FAST
ci	s[6]	17.999	SLOW	6.092	FAST
x[2]	s[7]	17.422	SLOW	5.407	FAST
x[2]	cout	17.371	SLOW	5.398	FAST
y[3]	s[7]	17.054	SLOW	5.624	FAST
y[4]	s[7]	17.027	SLOW	5.778	FAST
y[3]	cout	17.004	SLOW	5.614	FAST

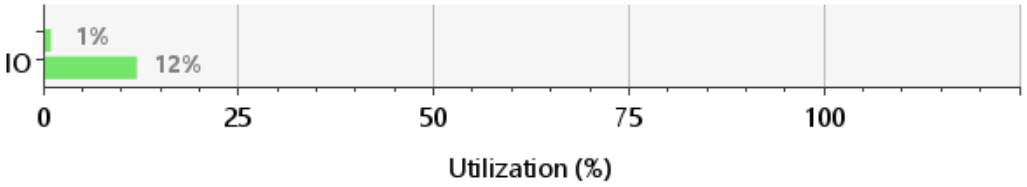
From Port	To Port	Max Delay	Max Process Corner	Min Delay	Min Process Corner
y[4]	cout	16.977	SLOW	5.769	FAST
y[2]	s[5]	16.553	SLOW	4.996	FAST
x[1]	s[5]	16.492	SLOW	5.152	FAST
y[0]	s[3]	16.403	SLOW	5.837	FAST
y[2]	s[6]	16.279	SLOW	5.088	FAST
x[1]	s[6]	16.218	SLOW	5.244	FAST
x[0]	s[4]	16.092	SLOW	4.985	FAST
ci	s[4]	15.997	SLOW	5.138	FAST
x[2]	s[5]	15.575	SLOW	4.744	FAST
x[2]	s[6]	15.301	SLOW	4.836	FAST
x[3]	s[7]	15.264	SLOW	4.957	FAST
x[3]	cout	15.213	SLOW	4.948	FAST
y[3]	s[5]	15.207	SLOW	4.961	FAST
y[4]	s[5]	15.181	SLOW	5.115	FAST
y[3]	s[6]	14.933	SLOW	5.053	FAST
y[4]	s[6]	14.907	SLOW	5.207	FAST
y[0]	s[2]	14.798	SLOW	5.286	FAST
y[1]	s[3]	14.786	SLOW	5.015	FAST
y[2]	s[4]	14.277	SLOW	4.133	FAST
x[1]	s[4]	14.217	SLOW	4.289	FAST
x[4]	s[7]	13.575	SLOW	4.302	FAST
x[4]	cout	13.524	SLOW	4.292	FAST
x[3]	s[5]	13.417	SLOW	4.295	FAST
x[2]	s[4]	13.300	SLOW	3.882	FAST
y[5]	s[7]	13.288	SLOW	3.948	FAST
y[5]	cout	13.237	SLOW	3.939	FAST
y[1]	s[2]	13.180	SLOW	4.464	FAST

From Port	To Port	Max Delay	Max Process Corner	Min Delay	Min Process Corner
y[1]	s[2]	13.180	SLOW	4.464	FAST
x[3]	s[6]	13.143	SLOW	4.387	FAST
y[3]	s[4]	12.932	SLOW	4.099	FAST
y[4]	s[4]	12.666	SLOW	4.408	FAST
x[5]	s[7]	12.543	SLOW	3.797	FAST
y[0]	s[1]	12.535	SLOW	4.514	FAST
x[5]	cout	12.493	SLOW	3.787	FAST
x[0]	s[3]	12.492	SLOW	3.923	FAST
ci	s[3]	12.398	SLOW	4.076	FAST
x[4]	s[5]	11.728	SLOW	3.639	FAST
y[6]	s[7]	11.535	SLOW	3.573	FAST
y[6]	cout	11.485	SLOW	3.564	FAST
x[4]	s[6]	11.454	SLOW	3.731	FAST
y[1]	s[1]	11.329	SLOW	4.313	FAST
y[5]	s[6]	11.167	SLOW	3.378	FAST
y[0]	s[0]	11.155	SLOW	4.336	FAST
x[3]	s[4]	11.142	SLOW	3.432	FAST
x[6]	s[7]	10.975	SLOW	3.645	FAST
x[6]	cout	10.925	SLOW	3.635	FAST
x[0]	s[2]	10.887	SLOW	3.373	FAST
ci	s[2]	10.792	SLOW	3.526	FAST
y[2]	s[3]	10.677	SLOW	3.072	FAST
x[1]	s[3]	10.617	SLOW	3.228	FAST
x[7]	cout	10.451	SLOW	3.172	FAST
x[5]	s[6]	10.422	SLOW	3.226	FAST
y[5]	s[5]	10.185	SLOW	3.275	FAST
y[7]	cout	9.899	SLOW	2.681	FAST

From Port	To Port	Max Delay	Max Process Corner	Min Delay	Min Process Corner
x[6]	s[7]	10.975	SLOW	3.645	FAST
x[6]	cout	10.925	SLOW	3.635	FAST
x[0]	s[2]	10.887	SLOW	3.373	FAST
ci	s[2]	10.792	SLOW	3.526	FAST
y[2]	s[3]	10.677	SLOW	3.072	FAST
x[1]	s[3]	10.617	SLOW	3.228	FAST
x[7]	cout	10.451	SLOW	3.172	FAST
x[5]	s[6]	10.422	SLOW	3.226	FAST
y[5]	s[5]	10.185	SLOW	3.275	FAST
y[7]	cout	9.899	SLOW	2.681	FAST
x[7]	s[7]	9.831	SLOW	3.257	FAST
x[2]	s[3]	9.700	SLOW	2.820	FAST
x[5]	s[5]	9.440	SLOW	3.118	FAST
y[7]	s[7]	9.279	SLOW	2.845	FAST
x[4]	s[4]	9.213	SLOW	2.936	FAST
y[3]	s[3]	9.047	SLOW	3.102	FAST
y[6]	s[6]	9.016	SLOW	3.047	FAST
x[1]	s[2]	9.012	SLOW	2.678	FAST
y[2]	s[2]	8.681	SLOW	2.838	FAST
x[0]	s[1]	8.624	SLOW	2.601	FAST
ci	s[1]	8.530	SLOW	2.754	FAST
x[6]	s[6]	8.443	SLOW	2.965	FAST
ci	s[0]	7.708	SLOW	2.455	FAST
x[2]	s[2]	7.704	SLOW	2.586	FAST
x[3]	s[3]	7.257	SLOW	2.413	FAST
x[0]	s[0]	7.244	SLOW	2.406	FAST
x[1]	s[1]	7.161	SLOW	2.378	FAST

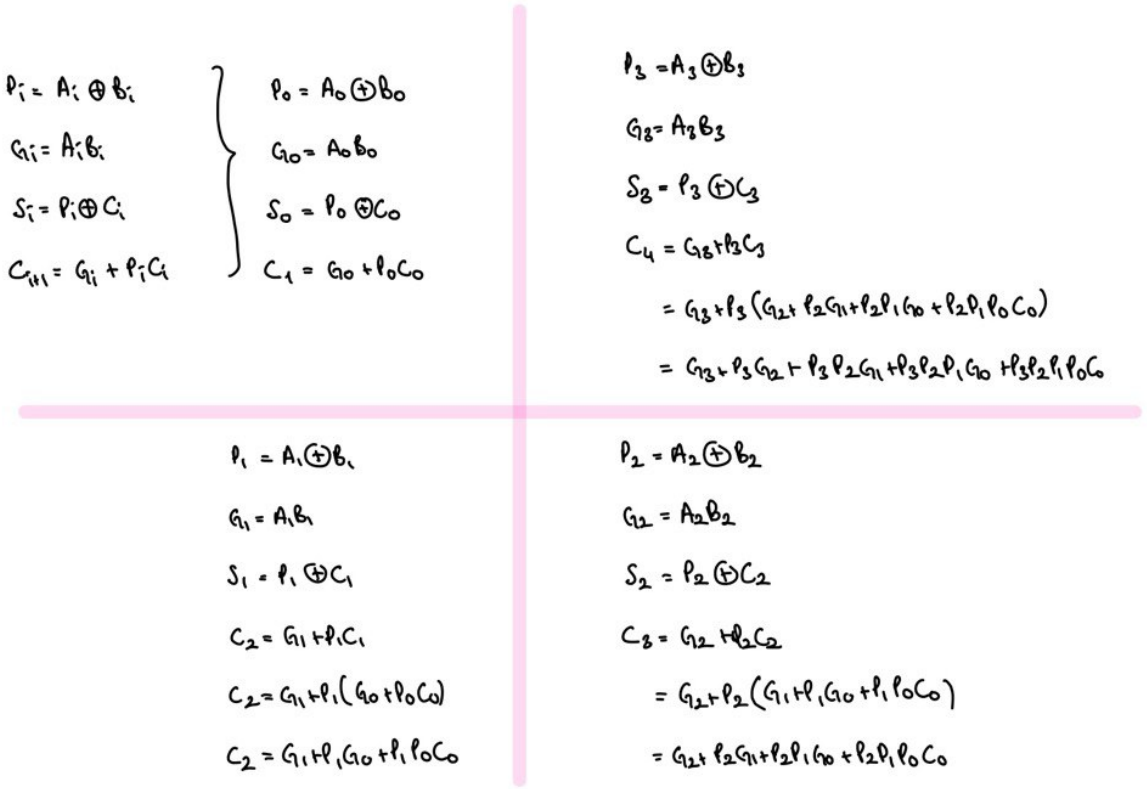
Summary

Resource	Utilization	Available	Utilization %
LUT	34	63400	0.05
IO	26	210	12.38



CARRY LOOKAHEAD ADDER

With the help of the book i write carry and propagate function for all bits then write the code according to.



- Verilog Code:

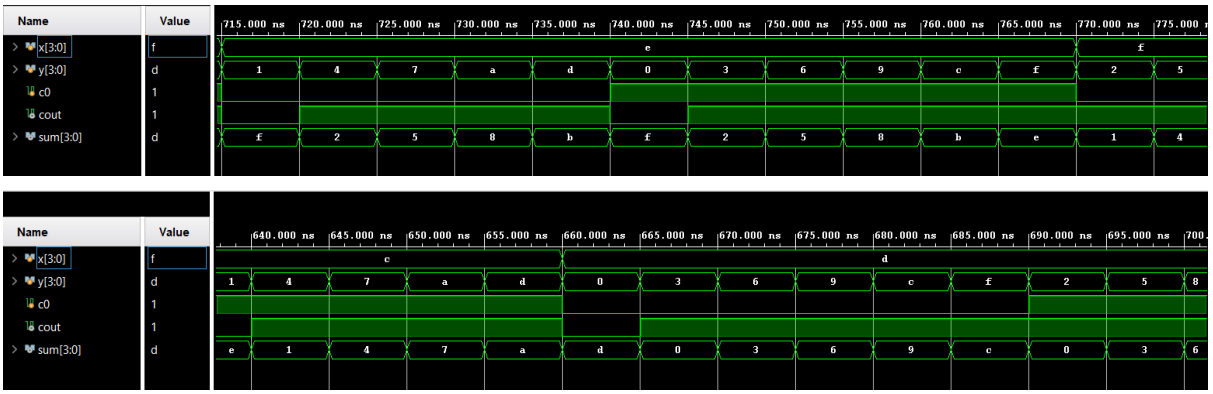
```
module CLA(
    input [3:0] x,y,
    input c0,
    output cout,
    output [3:0] sum
    );
    (*DONT_TOUCH = "TRUE"*) wire [4:0] c;
    (*DONT_TOUCH = "TRUE"*) wire [3:0] p,g;

    assign c[0]=c0;
    assign cout = c[4];

    genvar i;
    generate
        for(i=0; i<4; i=i+1)begin

        assign p[i]= x[i] ^ y[i];
```

```
        assign g[i] = x[i] && y[i];
        assign sum[i] = p[i] ^c[i];
        assign c[i+1] = g[i] || (p[i] && c[i]);

        end
    endgenerate

endmodule
```

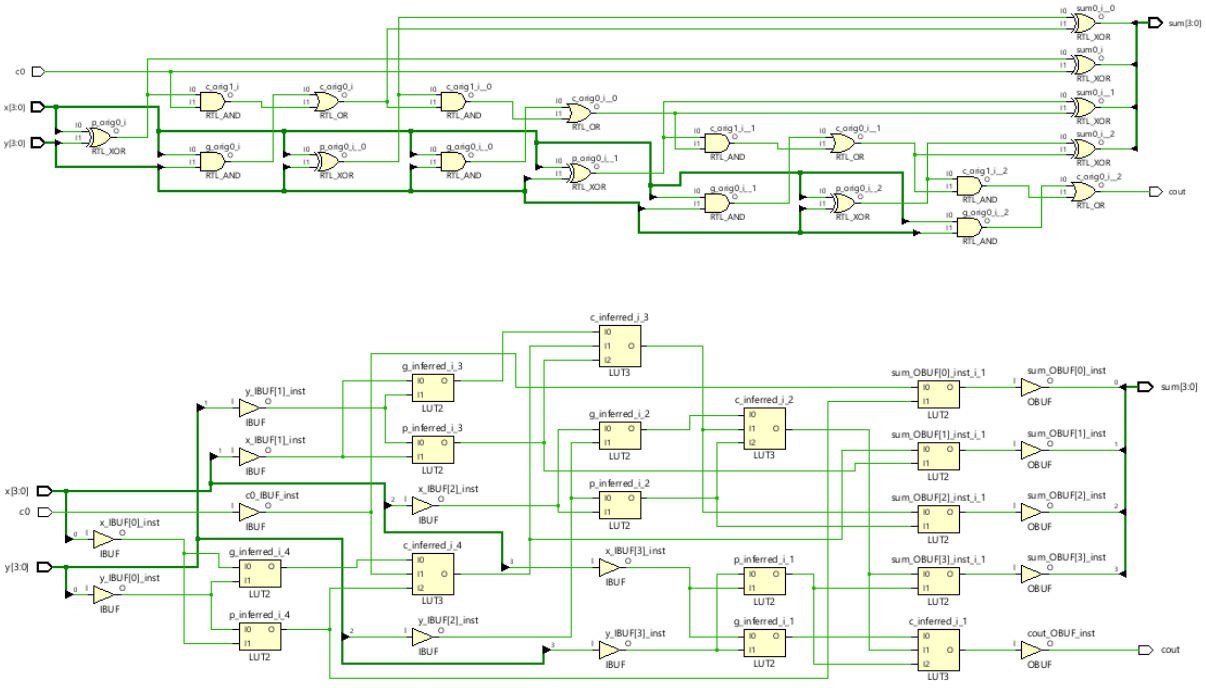
- Testbench Code:

```
module CLA_tb;
  CLA uut (.x(x),.y(y), .c0(c0),.cout(cout),.sum(sum));

  reg [3:0] x;
  reg [3:0] y;
  reg c0;
  wire cout;
  wire [3:0] sum;

  initial begin
    for (integer i = 20; i < 511; i = i + 3) begin
      {x, c0, y} = i;
      #5;
    end
    #10 $finish;
  end
endmodule
```

- Simulation results:



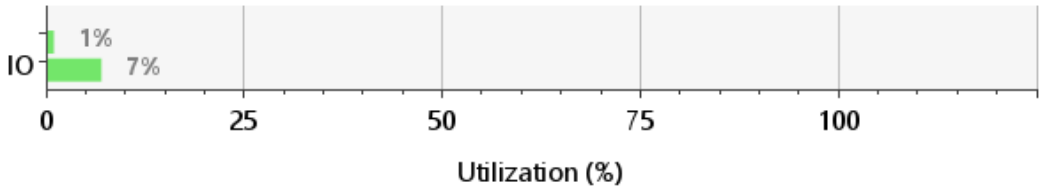
- RTL and technology Schematşcs:



- Timing and Utilization Reports:

Maximum combinational delay is 14.121 and 16 LUTs are used.Both combinational delay and used LUTs are less compared to 4 bit RCA.

Resource	Utilization	Available	Utilization %
LUT	16	63400	0.03
IO	14	210	6.67



From Port	To Port	M a 1	Max Process Corner	Min Delay	Min Process Corner
c0	sum[3]	14.121	SLOW	5.175	FAST
c0	sum[2]	14.046	SLOW	5.197	FAST
c0	cout	13.813	SLOW	5.048	FAST
c0	sum[1]	12.253	SLOW	4.524	FAST
y[0]	sum[3]	12.031	SLOW	3.698	FAST
y[0]	sum[2]	11.956	SLOW	3.719	FAST
c0	sum[0]	11.842	SLOW	4.388	FAST
y[0]	cout	11.723	SLOW	3.571	FAST
x[0]	sum[3]	11.711	SLOW	3.607	FAST
x[0]	sum[2]	11.636	SLOW	3.628	FAST
y[1]	sum[3]	11.478	SLOW	3.413	FAST
y[1]	sum[2]	11.404	SLOW	3.434	FAST
x[0]	cout	11.403	SLOW	3.480	FAST
y[1]	cout	11.170	SLOW	3.286	FAST
x[1]	sum[3]	11.023	SLOW	3.191	FAST
x[1]	sum[2]	10.948	SLOW	3.213	FAST
y[2]	sum[3]	10.758	SLOW	3.258	FAST
x[1]	cout	10.715	SLOW	3.064	FAST
y[2]	cout	10.450	SLOW	3.131	FAST
y[2]	sum[2]	10.325	SLOW	3.304	FAST
y[0]	sum[1]	10.163	SLOW	3.046	FAST
y[0]	sum[0]	10.084	SLOW	3.126	FAST
x[0]	sum[1]	9.843	SLOW	2.955	FAST
y[1]	sum[1]	9.642	SLOW	2.956	FAST
x[2]	sum[3]	9.561	SLOW	2.870	FAST
x[0]	sum[0]	9.546	SLOW	2.926	FAST
x[2]	sum[2]	9.494	SLOW	2.915	FAST
y[1]	sum[1]	9.642	SLOW	2.956	FAST
x[2]	sum[3]	9.561	SLOW	2.870	FAST
x[0]	sum[0]	9.546	SLOW	2.926	FAST
x[2]	sum[2]	9.494	SLOW	2.915	FAST
x[2]	cout	9.254	SLOW	2.743	FAST
x[3]	sum[3]	9.198	SLOW	2.798	FAST
x[1]	sum[1]	9.186	SLOW	2.763	FAST
x[3]	cout	8.883	SLOW	2.647	FAST
y[3]	sum[3]	8.500	SLOW	2.557	FAST
y[3]	cout	8.357	SLOW	2.406	FAST

ADD-SUB

ı studied this circuit from the book and there should be xor gates in the boxes. I write my code with the help of the book.

- Verilog Code

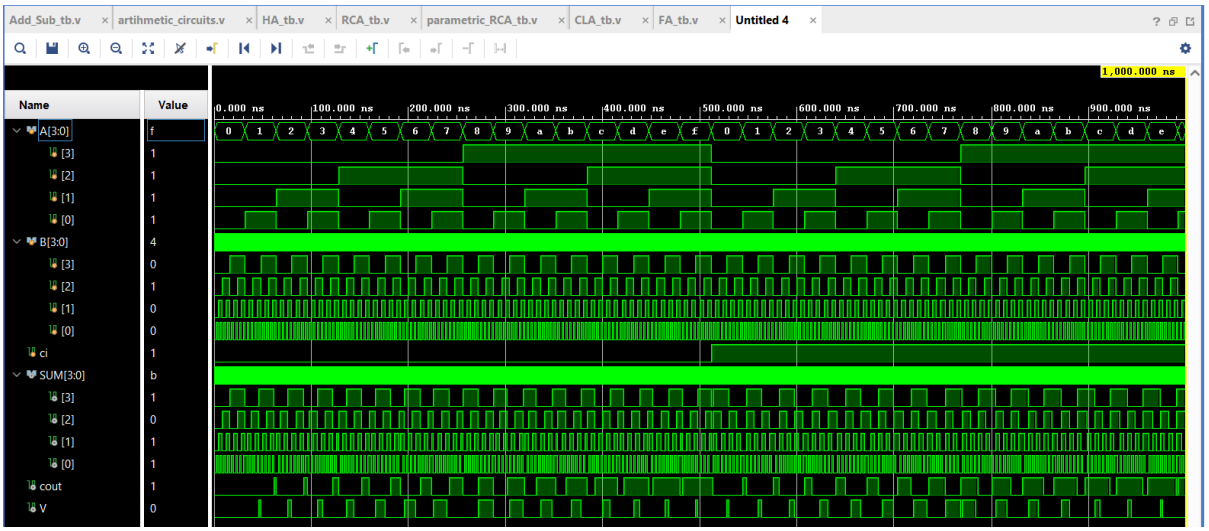
```
module Add_Sub(
  input [3:0] A, B,
  input ci,
  output [3:0] SUM,
  output cout,
  output V
);
```

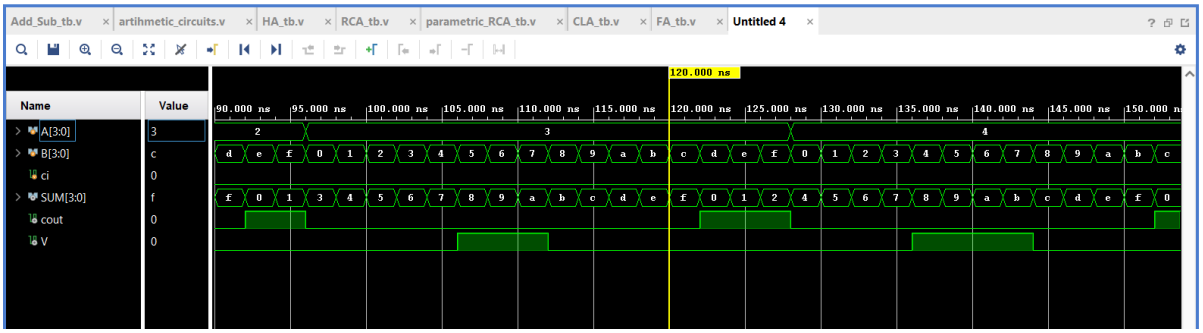
```
  (* dont_touch="true" *) wire [3:0] xxor ;
  (* dont_touch="true" *) wire [4:0] c ;

  assign c[0] = ci;
  assign cout = c[4];

  genvar i;
  generate
    for (i = 0; i < 4; i = i + 1) begin : add_sub_loop
      assign xxor[i] = ci^B[i];
      FA fai (A[i], xxor[i], c[i], c[i + 1], SUM[i]);
    end
  endgenerate

  assign V = c[3] ^ c[4];

endmodule
```

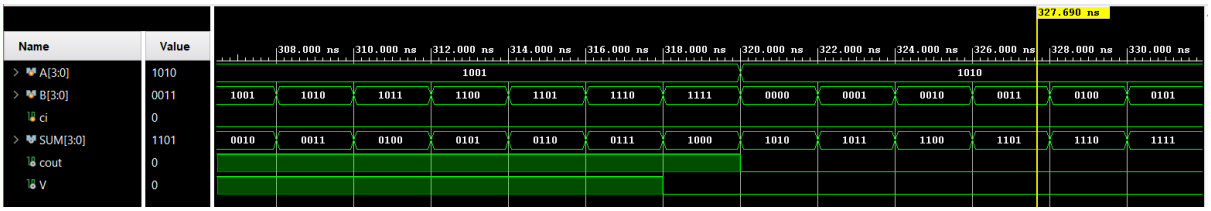
- Testbench Code

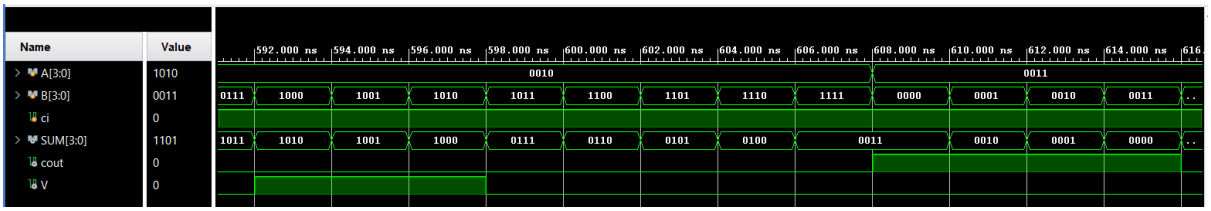
```
module add_sub_tb();
reg [3:0] A,B;
reg ci;
wire [3:0] SUM;
wire cout;
wire V;

Add_Sub add_subx(A, B, ci, SUM, cout, V);

initial
begin

for(integer i=0;i<512;i=i+1)
        begin
        {ci,A,B}=i;
        #2;
        end
        $finish();
end
endmodule
```

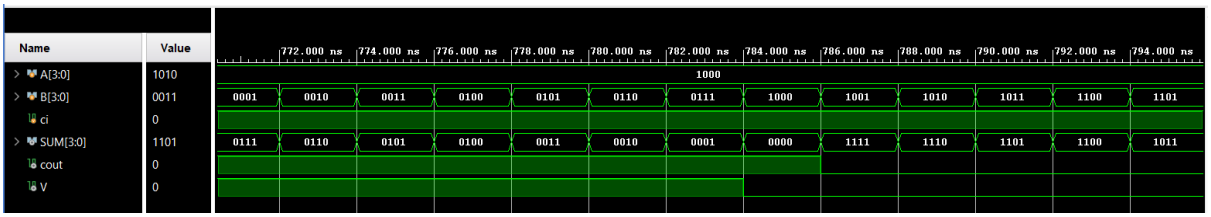
- Simulation Results



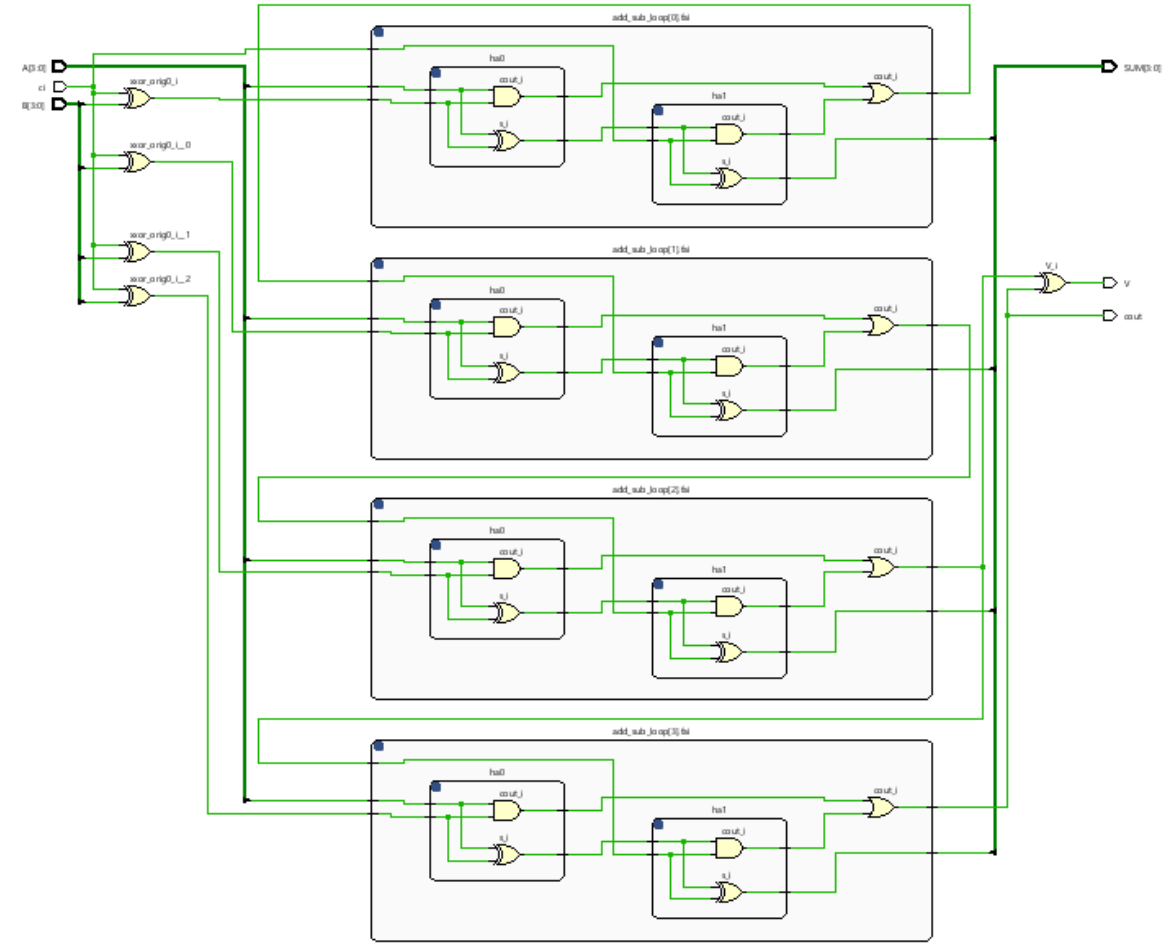


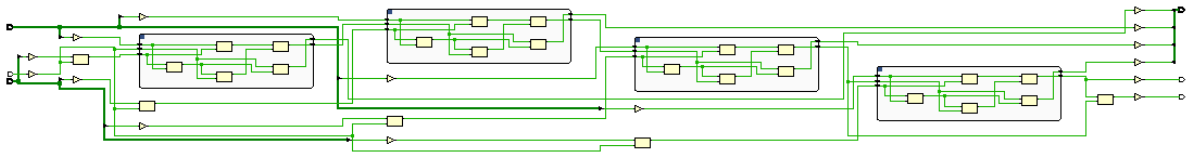






- RTL and Technology Schematics:



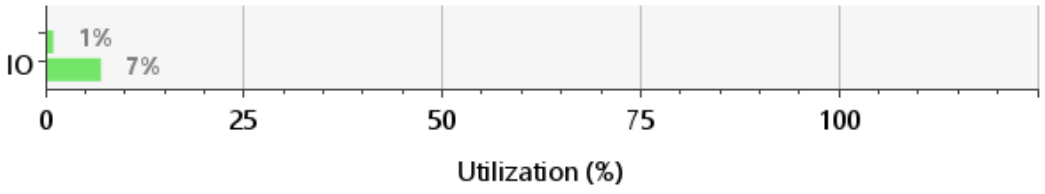


- Timing and Simulation Reports:

Maximum combinational delay is 11.22ns and 19 LUTs are used

From Port	To Port	Max Delay	Max Process Corner	Min Delay	Min Process Corner
A[0]	SUM[0]	7.799	SLOW	2.618	FAST
A[0]	SUM[1]	9.024	SLOW	2.859	FAST
A[0]	SUM[2]	10.670	SLOW	3.437	FAST
A[0]	SUM[3]	11.984	SLOW	3.844	FAST
A[0]	V	12.595	SLOW	3.919	FAST
A[0]	cout	14.493	SLOW	4.619	FAST
A[1]	SUM[1]	7.499	SLOW	2.539	FAST
A[1]	SUM[2]	9.854	SLOW	2.678	FAST
A[1]	SUM[3]	11.167	SLOW	3.085	FAST
A[1]	V	11.778	SLOW	3.160	FAST
A[1]	cout	13.677	SLOW	3.860	FAST
A[2]	SUM[2]	8.192	SLOW	2.763	FAST
A[2]	SUM[3]	9.056	SLOW	2.874	FAST
A[2]	V	9.667	SLOW	2.949	FAST
A[2]	cout	11.566	SLOW	3.649	FAST
A[3]	SUM[3]	7.659	SLOW	2.542	FAST
A[3]	V	8.403	SLOW	2.742	FAST
A[3]	cout	10.301	SLOW	3.277	FAST
B[0]	SUM[0]	7.790	SLOW	2.629	FAST
B[0]	SUM[1]	9.015	SLOW	2.861	FAST
B[0]	SUM[2]	10.662	SLOW	3.439	FAST
B[0]	SUM[3]	11.975	SLOW	3.846	FAST
B[0]	V	12.587	SLOW	3.921	FAST
B[0]	cout	14.485	SLOW	4.621	FAST
B[1]	SUM[1]	7.554	SLOW	2.568	FAST
B[1]	SUM[2]	9.909	SLOW	2.912	FAST
B[1]	SUM[2]	9.909	SLOW	2.912	FAST
B[1]	SUM[3]	11.222	SLOW	3.319	FAST
B[1]	V	11.833	SLOW	3.394	FAST
B[1]	cout	13.732	SLOW	4.094	FAST
B[2]	SUM[2]	9.340	SLOW	3.143	FAST
B[2]	SUM[3]	10.204	SLOW	3.038	FAST
B[2]	V	10.816	SLOW	3.113	FAST
B[2]	cout	12.714	SLOW	3.813	FAST
B[3]	SUM[3]	8.241	SLOW	2.758	FAST
B[3]	V	8.985	SLOW	2.946	FAST
B[3]	cout	10.884	SLOW	3.481	FAST
ci	SUM[0]	9.105	SLOW	2.676	FAST
ci	SUM[1]	10.330	SLOW	3.073	FAST
ci	SUM[2]	11.977	SLOW	3.416	FAST
ci	SUM[3]	13.291	SLOW	3.162	FAST
ci	V	13.902	SLOW	3.350	FAST
ci	cout	15.800	SLOW	3.885	FAST

Resource	Utilization	Available	Utilization %
LUT	19	63400	0.03
IO	15	210	7.14



Maximum Combinational Delay and Used LUTs for 4 circuit

Using Carry Lookahead Addder is better in terms of delay and used LUT number.

	Maximum Comb. delay	LUT
4 bit RCA	15.782	19
8 bit RCA	24.115	34
CLA 4 bit	14.121	16
Add-Sub	15.800	19