DIGITAL SYSTEM DESIGN APPLICATIONS

Experiment 4

ARITHMETIC CIRCUITS

Berfin Duman
040190108

Contents

1 ARITHMETIC CIRCUITS

1.1 Half Adder

As expected, i created HA module in arithmetic_circuits.v file This module takes 1-bit inputs x and y ant 1-bit outputs cout, which means carry out, and 1-bit sum, which means sum. I complete this HA module provides the truth table (?) using dont_touch constraint.
And then for testing, i wrote a testbench using for loop to simplify it. I ran behaviorla simulation and took rtl analysis and schematic to make sure module was written correctly. To simplicity, i wroted results of test on tcl console and i added it on this report .And after ensure correctness, i contine synthesize and made implementation steps.

X	Y	Co	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Table 1: Half Adder Truth Table and block representation

Listing 1: arithmetic_circutis.v -Ha module

```
`timescale 1ns / 1ps
module HA(input X, Y, output  C_o,S);
    (* dont_touch="true" *) wire x_wires=X;
    (* dont_touch="true" *) wire y_wires=Y;
    (* dont_touch="true" *) wire Co_wires=C_o;
    (* dont_touch="true" *) wire S_wires=S;
    assign S=X^Y ;
    assign C_o=X&Y;
endmodule
```

Listing 2: ha_tb.v testbench module

```
‘timescale 1ns / 1ps
module HA_tb();
  reg X,Y;
  wire C_o,S;
  parameter wait_time = 50;
  integer i;
  reg [1:0]correct_results[0:3];
  initial begin
    correct_results[0]  = 00; correct_results[1]  = 01;
       correct_results[2]  = 01 ;correct_results[3] = 10;
  end
  HA UUT(.X(X),.Y(Y),.C_o(C_o),.S(S));
  initial
  begin
    for(i=0;i<4;i=i+1)
    begin
      {X,Y} = i;
      #(wait_time);
      $write("{X,Y}=%d%d => {C_o,S} = %d%d -- ",X,Y,C_o,S);
      if({C_o,S} == correct_results[i])
        $display("TRUE");
    else
        $display("FALSE");
    end
    $finish();
  end
endmodule
```

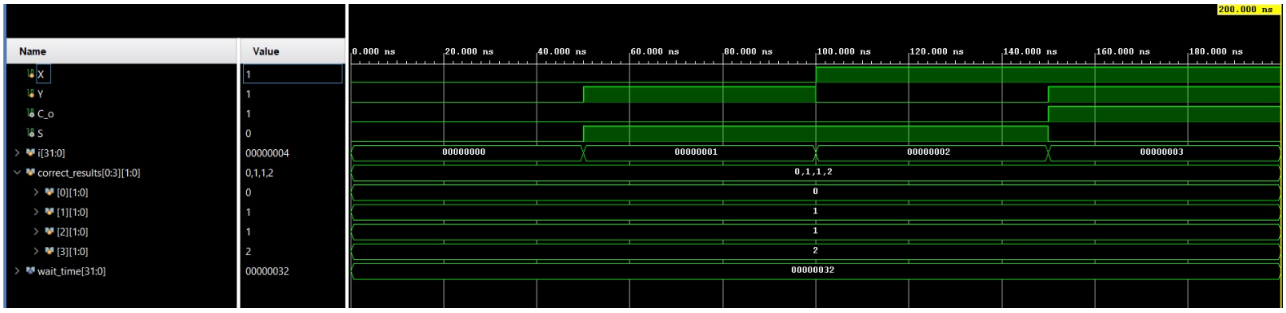


Figure 1: Results of simulation of HA module

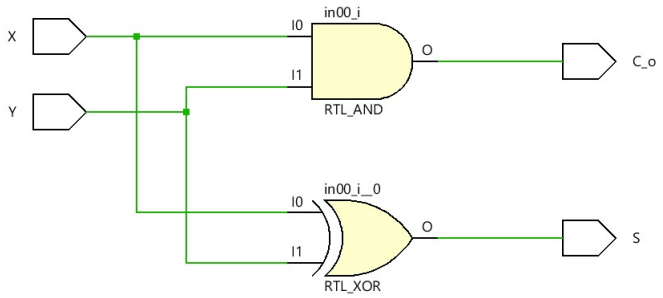


Figure 2: Rtl schematics of HA module

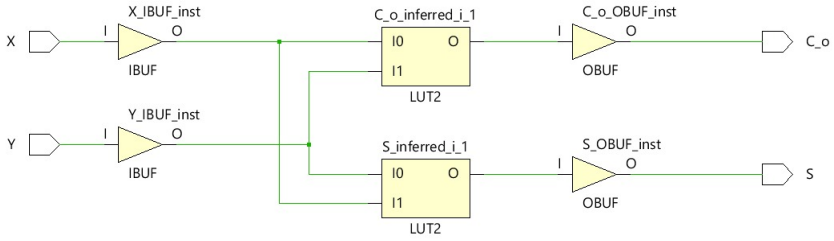


Figure 3: Technology schematics of HA module

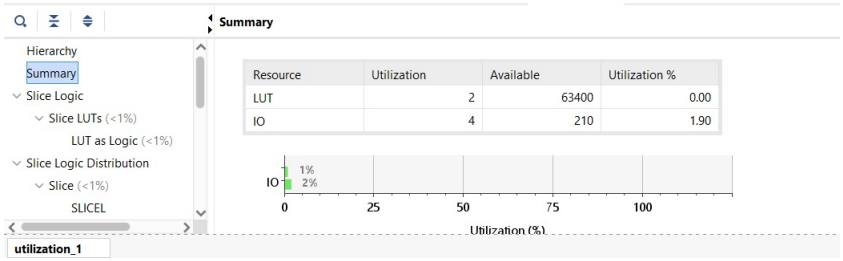


Figure 4: Utilization report of HA module

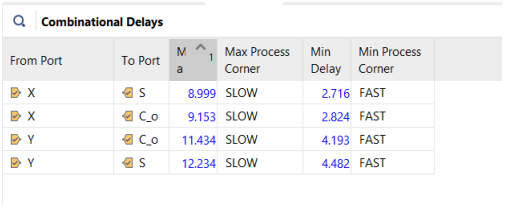


Figure 5: Combinational delay of HA module

Review HA module and results of rtl schematic, I saw that this module included two gates, "AND", and "XOR" gates, as I expected that results. And these gates was expressed as two inputs LUTs tables. And then, I came implementation part and i took utilization

report, technology schematic was verification by this report, we used two luts and 4 i/o parts as i expected. And then i took combinational delays and added on report.

1.2 Full Adder

A full adder is a combinational digital circuit that performs binary addition. It takes in three binary inputs: A, B, and an input carry (Cin), and produces two outputs: a sum (S) and an output carry (Cout). The full adder is a building block in digital circuit design and is commonly used in the construction of more complex arithmetic circuits.
As requested, I realized FA module using HA module and OR gate, which i wrote before. For simplicity coding, i drew hand FA circuit before and then i wrote verilog code. And as always, i wrote testbench code for all 3 inputs (A,B and Carry in), and simulate it.

Listing 3: arithmetic_circutis.v -FA module

```
module FA(  input X, Y, C_i, output  C_o, S);
    (* dont_touch="true" *) wire x_wires=X;
    (* dont_touch="true" *) wire y_wires=Y;
    (* dont_touch="true" *) wire Ci_wires=C_i;
    (* dont_touch="true" *) wire Co_wires=C_o;
    (* dont_touch="true" *) wire S_wires=S;
    (* dont_touch="true" *)  wire [2:0] wires;
    HA h_ad1(.X(X), .Y(Y), .C_o(wires[1]), .S(wires[0]));
    HA h_ad2(.X(wires[0]), .Y(C_i), .C_o(wires[2]), .S(S));
    or or1(C_o, wires[1], wires[2]);
endmodule
```

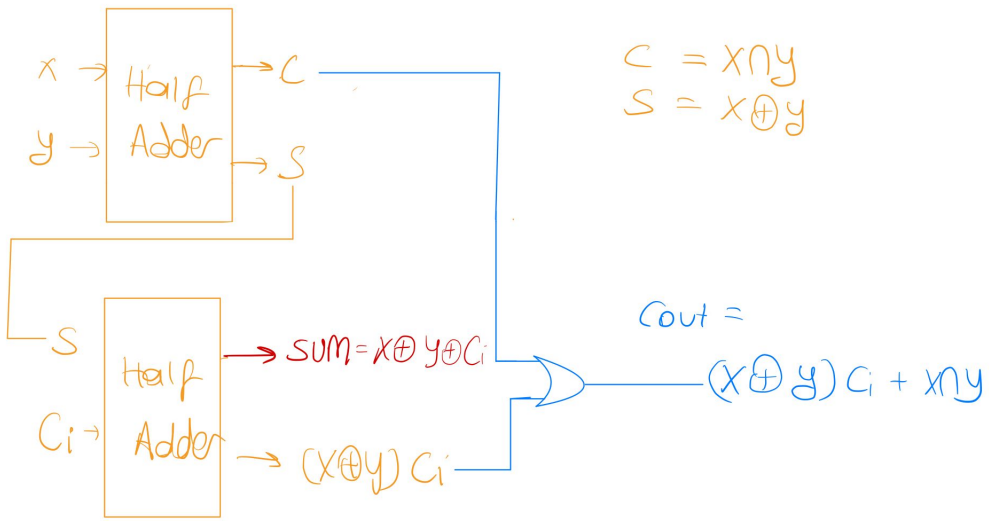


Figure 6: Hand drawn circuits of FA module

Listing 4: FA_tb.v testbench module

```
`timescale 1ns / 1ps

module FA_tb();
  // ----- Inputs & Outputs
  reg X,Y,C_i;
  wire C_o,S;
  // ----- Testbench Parameters ----- //
  parameter wait_time = 50;
  integer i;
  reg [2:0]correct_results[0:7];
  initial begin
    correct_results[0]  = 00; correct_results[1]  = 10;
       correct_results[2]  = 10 ;correct_results[3] = 01;
    correct_results[4]  = 10; correct_results[5]  = 01;
       correct_results[6]  = 01 ;correct_results[7] = 11;
  end

  // ---------- UUT Instantiation -------------- //
  FA UUT(.X(X),.Y(Y),.C_i(C_i),.C_o(C_o),.S(S));
  // ------- Test Procedure ------- //
  initial
  begin
    for(i=0;i<8;i=i+1)
    begin
      {X,Y,C_i} = i;
      #(wait_time);
      $write("{X,Y,C_i}=%d%d%d => {S,C_o} = %d%d --
         ",X,Y,C_i,S,C_o);
      if({S,C_o} == correct_results[i])
        $display("TRUE");
      else
        $display("FALSE");
    end
    $finish();
  end
endmodule
```

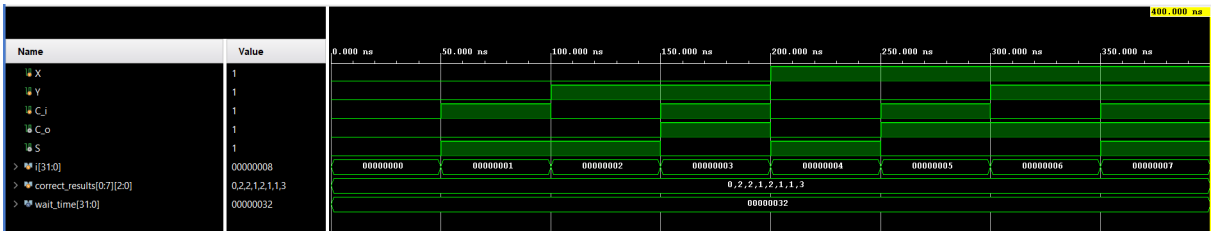


Figure 7: Results of simulation of FA module

Listing 5: TCL Results of Testbench Code

```
# run 1000ns
{X,Y,C_i}=000 => {S,C_o} = 00 -- TRUE
{X,Y,C_i}=001 => {S,C_o} = 10 -- TRUE
{X,Y,C_i}=010 => {S,C_o} = 10 -- TRUE
{X,Y,C_i}=011 => {S,C_o} = 01 -- TRUE
{X,Y,C_i}=100 => {S,C_o} = 10 -- TRUE
{X,Y,C_i}=101 => {S,C_o} = 01 -- TRUE
{X,Y,C_i}=110 => {S,C_o} = 01 -- TRUE
{X,Y,C_i}=111 => {S,C_o} = 11 -- TRUE
```

This rtl schematic was consistent with hand drawn circuit, and results of simulation and tcl output was as expected too. So, i thought i was on the right track and i was continue synthesize part.

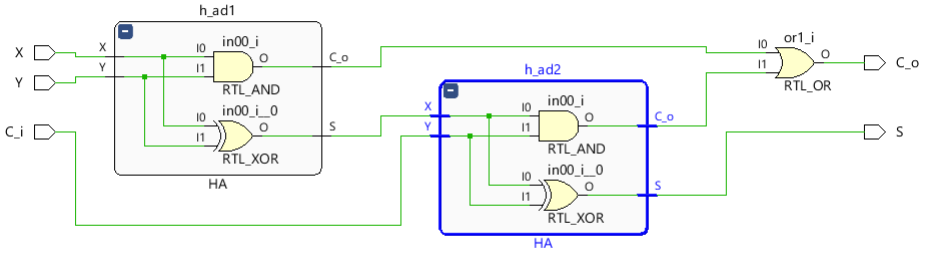


Figure 8: Rtl schematics of FA module

Technology schematic was as expected, I use 2 HA module, which module had 2 gates and these gates represented 2 LUT2 tables as we saw before, so 4 Lut2 tables came to 2 HA Modules, and for carry out bit we use OR gates so, expected a LUT2 tables for this gate, finally I totally expected 5 LUT2 tables, which this schematic was consistent with my expectations.

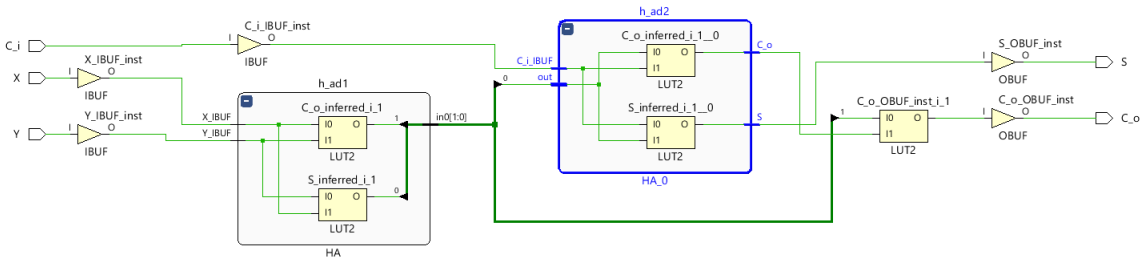


Figure 9: Technology schematics of FA module

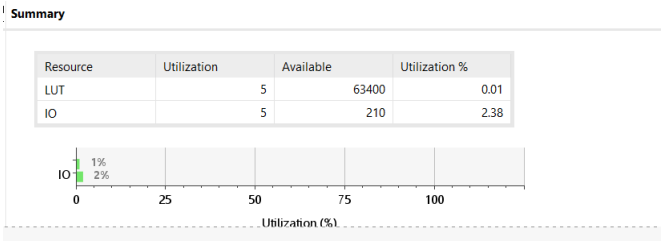


Figure 10: Utilization report of FA module

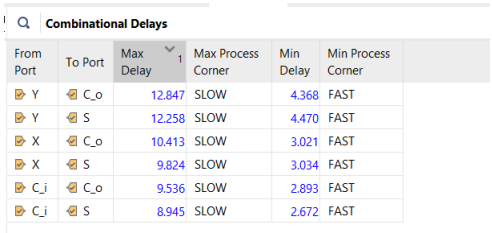


Figure 11: Combinational delay of FA module

Everything saw correct, and i continue implementation part, i created 2 report, utilization and combinational delay. Utilization report was consistent with technology schematic 3 input (X, Y, Cin) and 2 output (Cout, Sum), and 5 LUTS. And when I looked combinational delay report, I saw max delay was 7.930 from X to C_o.

1.3 Ripple Carry Adder

1.3.1 First Case

Ripple carry adder (RCA) is used to add two or more multi-bit numbers. Its main purpose is to add numbers digit by digit and transmit the carry bit from one digit to another.
For realize RCA, I wrote RCA module in arithemetics_circuits.vf file. This module realizes summing binary four-bit, for this this module has 2 4-bit inputs as X and Y, 1-bir carry input as ci, and 1-bit output as Carry out and finally 4-bir output as sum. I wrote this module as requested in report using 4 FA circuits and I used dont_touch constrain for prevent wire optimization. Then follow up as always i wrote testbench,(I wrote a test on 5 different X and Y inputs) i simulate module, i organized constrain file looking at inputs and outputs of RCA module. Then i followed synthesize and implementation part modules respectively, and analysed this outputs

Listing 6: arithmetic_circutis.v -RCA module

```
1 module RCA(  input C_i,[3:0] X, [3:0] Y,  output  C_o, [3:0]
     S);
2     (* dont_touch="true" *) wire x_wires=X;
3     (* dont_touch="true" *) wire y_wires=Y;
4     (* dont_touch="true" *) wire Ci_wires=C_i;
```

```
    (* dont_touch="true" *) wire Co_wires=C_o;
    (* dont_touch="true" *) wire S_wires=S;
    (* dont_touch="true" *) wire [2:0] wires;
    FA fa1(X[0],Y[0],C_i,wires[0], S[0]);
    FA fa2(X[1],Y[1],wires[0],wires[1], S[1]);
    FA fa3(X[2],Y[2],wires[1],wires[2], S[2]);
    FA fa4(X[3],Y[3],wires[2],C_o, S[3]);

    // assign C_o = wires[0] & wires[2];
endmodule
```

Listing 7: RCA_tb.v testbench module

```
`timescale 1ns / 1ps

module RCA_tb();

  // ----- Inputs & Outputs
  reg [3:0] X,Y;
  reg C_i;
  wire C_o;
  wire [3:0] S;

  // ----- Testbench Parameters ----- //
  parameter wait_time = 50;
  integer i;
  reg [3:0] A_numbers[0:4];
  reg [3:0] B_numbers[0:4];
  reg [3:0] correct_results[0:4];
  reg [4:0] C= 5'd0;
  reg[0:4] C_out= 5'b11100;
  initial begin
        A_numbers[0] = 4'hD;
        A_numbers[1] = 4'hE;
        A_numbers[2] = 4'hB;
        A_numbers[3] = 4'h7;
        A_numbers[4] = 4'h5;
        B_numbers[0] = 4'h6;
        B_numbers[1] = 4'hA;
        B_numbers[2] = 4'hE;
        B_numbers[3] = 4'h8;
        B_numbers[4] = 4'hA;
        correct_results[0] = 4'h3;
        correct_results[1] = 4'h8;
        correct_results[2] = 4'h9;
        correct_results[3] = 4'hF;
```

```
34          correct_results[4] = 4'hF;
35      end
36
37
38    // --------------------------------- //
39
40    // ---------- UUT Instantiation -------------- //
41    //RCA RCA(input [3:0] X, [3:0] Y, C_i, output C_o, [3:0] S)
42    //parametric_RCA (input C_i,[size-1:0] X, [size-1:0] Y,
          output C_o, [size-1:0] S);
43    //CLA UUT(.c0(C_i),.X(X),.Y(Y),.cout(C_o),.sum(S));
44    parametric_RCA UUT(.C_i(C_i),.X(X),.Y(Y),.C_o(C_o),.S(S));
45
46
47    // ------- Test Procedure ------- //
48    initial
49    begin
50
51      for(i=0;i<5;i=i+1)
52      begin
53              X=A_numbers[i];
54              Y=B_numbers[i];
55        C_i=C[i];
56        #(wait_time);
57  //      $write("{X,+ Y ,C_i}=%d%d%d%d + %d%d%d%d, Ci= %d =>
      {C_o,S} = %d%d%d%d%d  --
      ",X[0],X[1],X[2],X[3],Y[0],Y[1],Y[2],Y[3],C_i,C_o,S[0],S[1],S[2],S[3]);
58
59        $write("{X+ Y ,C_i}=%b + %b, Ci= %b => {C_o,S} = %b%b --
              ",X,Y,C_i,C_o, S );
60        if( {C_o, S} == {C_out[i],correct_results[i]})
61          $display("TRUE");
62        else
63          $display("FALSE");
64
65
66      end
67
68      $finish();
69
70    end
```

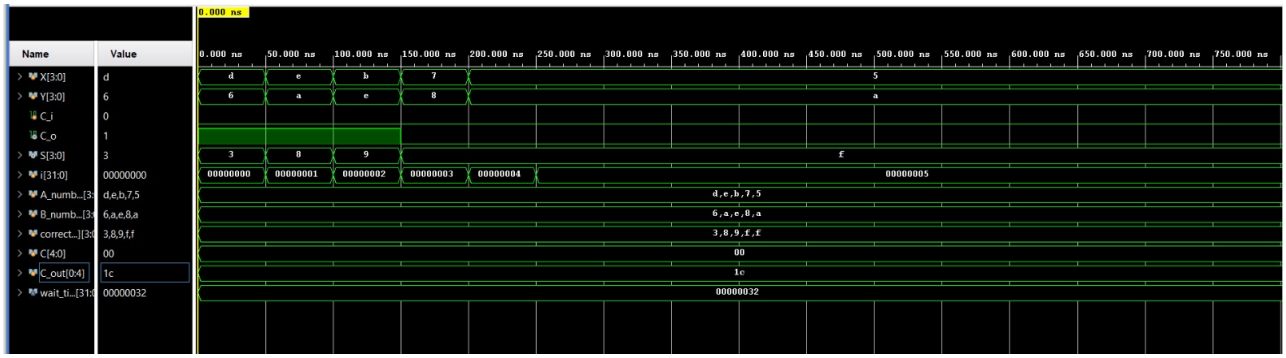


Figure 12: Results of simulation of RCA module

Listing 8: TCL Results of Testbench Code

```
# run 1000ns
{X+ Y ,C_i}=1101 + 0110, Ci= 0 => {C_o,S} = 10011 -- TRUE
{X+ Y ,C_i}=1110 + 1010, Ci= 0 => {C_o,S} = 11000 -- TRUE
{X+ Y ,C_i}=1011 + 1110, Ci= 0 => {C_o,S} = 11001 -- TRUE
{X+ Y ,C_i}=0111 + 1000, Ci= 0 => {C_o,S} = 01111 -- TRUE
{X+ Y ,C_i}=0101 + 1010, Ci= 0 => {C_o,S} = 01111 -- TRUE
```

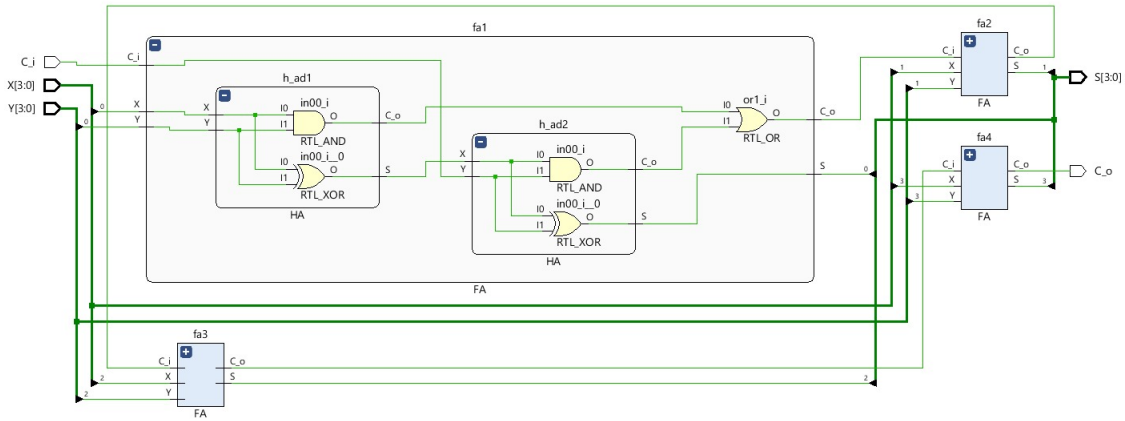


Figure 13: Rtl schematics of RCA module

```
## LEDs
set_property -dict { PACKAGE_PIN H17   IOSTANDARD LVCMOS33 } [get_ports { S[0] }]; #IO_L18P_T2_A24_15 Sch=led[0]
set_property -dict { PACKAGE_PIN K15   IOSTANDARD LVCMOS33 } [get_ports { S[1] }]; #IO_L24P_T3_RS1_15 Sch=led[1]
set_property -dict { PACKAGE_PIN J13   IOSTANDARD LVCMOS33 } [get_ports { S[2] }]; #IO_L17N_T2_A25_15 Sch=led[2]
set_property -dict { PACKAGE_PIN N14   IOSTANDARD LVCMOS33 } [get_ports { S[3] }]; #IO_L8P_T1_D11_14 Sch=led[3]
set_property -dict { PACKAGE_PIN R18   IOSTANDARD LVCMOS33 } [get_ports { C_o }]; #IO_L7P_T1_D09_14 Sch=led[4]
##Switches
set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 } [get_ports { X[0] }]; #IO_L24N_T3_RS0_15 Sch=sw[0]
set_property -dict { PACKAGE_PIN L16   IOSTANDARD LVCMOS33 } [get_ports { X[1] }]; #IO_L3N_T0_DQS_EMCCLK_14 Sch=sw[1]
set_property -dict { PACKAGE_PIN M13   IOSTANDARD LVCMOS33 } [get_ports { X[2] }]; #IO_L6N_T0_D08_VREF_14 Sch=sw[2]
set_property -dict { PACKAGE_PIN R15   IOSTANDARD LVCMOS33 } [get_ports { X[3] }]; #IO_L13N_T2_MRCC_14 Sch=sw[3]
set_property -dict { PACKAGE_PIN R17   IOSTANDARD LVCMOS33 } [get_ports { Y[0] }]; #IO_L12N_T1_MRCC_14 Sch=sw[4]
set_property -dict { PACKAGE_PIN T18   IOSTANDARD LVCMOS33 } [get_ports { Y[1] }]; #IO_L7N_T1_D10_14 Sch=sw[5]
set_property -dict { PACKAGE_PIN U18   IOSTANDARD LVCMOS33 } [get_ports { Y[2] }]; #IO_L17N_T2_A13_D29_14 Sch=sw[6]
set_property -dict { PACKAGE_PIN R13   IOSTANDARD LVCMOS33 } [get_ports { Y[3] }]; #IO_L5N_T0_D07_14 Sch=sw[7]
##Buttons
set_property -dict { PACKAGE_PIN N17   IOSTANDARD LVCMOS33 } [get_ports { C_i }]; #IO_L9P_T1_DQS_14 Sch=btnc
```

Figure 14: Constrain File for RCA Module

]

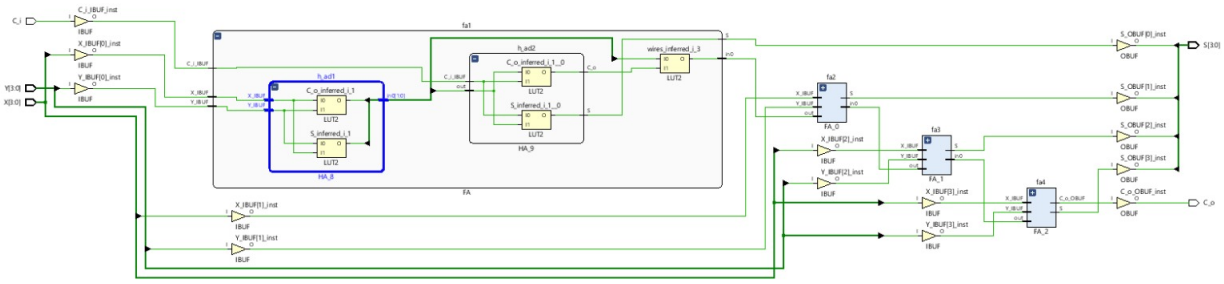


Figure 15: Technology schematics of RCA module

Summary

Resource	Utilization	Available	Utilization %
LUT	20	63400	0.03
IO	14	210	6.67

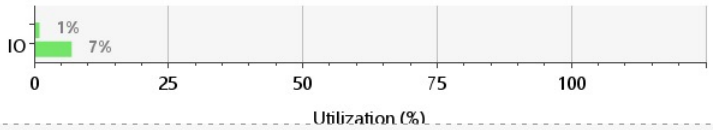


Figure 16: Utilization report of RCA module

Combinational Delays

From Port	To Port	Max Delay	Max Process Corner	Min Delay	Min Process Corner
Y[0]	C_o	18.449	SLOW	6.390	FAST
Y[0]	S[3]	17.296	SLOW	5.940	FAST
Y[0]	S[2]	16.178	SLOW	5.597	FAST
Y[1]	C_o	16.098	SLOW	5.869	FAST
X[0]	C_o	15.460	SLOW	4.768	FAST
Y[1]	S[3]	14.945	SLOW	5.419	FAST
C_i	C_o	14.372	SLOW	4.665	FAST
X[0]	S[3]	14.307	SLOW	4.318	FAST
Y[0]	S[1]	14.179	SLOW	4.825	FAST
Y[1]	S[2]	13.827	SLOW	5.076	FAST
C_i	S[3]	13.219	SLOW	4.216	FAST
X[0]	S[2]	13.189	SLOW	3.975	FAST
X[1]	C_o	13.139	SLOW	4.178	FAST
Y[0]	S[0]	13.043	SLOW	4.823	FAST
C_i	S[2]	12.101	SLOW	3.872	FAST
X[1]	S[3]	11.987	SLOW	3.728	FAST
Y[2]	C_o	11.922	SLOW	3.595	FAST
Y[1]	S[1]	11.837	SLOW	4.387	FAST
X[0]	S[1]	11.190	SLOW	3.204	FAST
X[2]	C_o	11.101	SLOW	3.257	FAST
X[1]	S[2]	10.868	SLOW	3.385	FAST
Y[3]	C_o	10.861	SLOW	2.832	FAST
Y[2]	S[3]	10.769	SLOW	3.145	FAST
Y[3]	S[3]	10.274	SLOW	3.266	FAST
C_i	S[1]	10.102	SLOW	3.101	FAST
X[0]	S[0]	10.054	SLOW	3.116	FAST
X[3]	C_o	9.975	SLOW	2.445	FAST
X[2]	S[3]	9.948	SLOW	2.807	FAST
Y[2]	S[2]	9.653	SLOW	3.035	FAST

Figure 17: Combinational delay of RCA module

when I look at reports, i saw all reports consistent each other, After schematic we saw that 20 lut, each 5 lut became from a FA module. I try to added 4 binary bits, so i used 4 FA module, this calculation explained 20 lut. And finally when i look at combinational delays,i saw max delay became 13.177 from C_i to C_o.

1.3.2 Parameter Based Rippled Carry Adder

When i look at ripple carry adder, I saw this circuit was realized adder whatever number of bits, so if I took bits number as parameter, this circuit could realized as adder whatever the number of parameters. So i create new module called parametric_RCA and i wrote verilog code for i realize for the parameter based adder circuit. When i wrote verilog code, I used generate - for struct and I determined how many bits it would be with the SIZE parameter. As always for this arithmetic_circuits.v file, i used dont_touch constraint too. For the first part i set parameter SIZE as 4.
After wrote module, i set this module as top, and i wrote testbench code to be sure, and simulated it.

Listing 9: arithmetic_circutis.v -parametrize_RCA module

```
1 module parametric_RCA #( parameter size = 4)( input
    C_i,[size-1:0] X, [size-1:0] Y,   output C_o, [size-1:0]
    S);
```

```
(* dont_touch="true" *) wire x_wires=X;
(* dont_touch="true" *) wire y_wires=Y;
(* dont_touch="true" *) wire Ci_wires=C_i;
(* dont_touch="true" *) wire Co_wires=C_o;
(* dont_touch="true" *) wire S_wires=S;
(* dont_touch="true" *) wire [size:0] wires;
assign wires[0]=C_i;
assign C_o=wires[size];
genvar j; // temp loop variable, used only
 // in the evaluation of the generate blocks
generate
for( j=0; j<size; j=j+1 )
 begin : FA_loop
 FA fa1(X[j],Y[j],wires[j], wires[j+1], S[j]);
 end
endgenerate
endmodule
```

Listing 10: parametric_RCA_tb.v testbench module

```
‘‘timescale 1ns / 1ps
module RCA_tb();
  // ----- Inputs & Outputs
  reg [3:0] X, Y;
  reg C_i;
  wire C_o;
  wire [3:0] S;
  // ----- Testbench Parameters ----- //
  parameter wait_time = 50;
  integer i;
  reg [3:0] A_numbers[0:4];
  reg [3:0] B_numbers[0:4];
  reg [3:0] correct_results[0:4];
  reg [4:0] C= 5'd0;
  reg[0:4] C_out= 5'b11100;
  initial begin
        A_numbers[0] = 4'hD;
        A_numbers[1] = 4'hE;
        A_numbers[2] = 4'hB;
        A_numbers[3] = 4'h7;
        A_numbers[4] = 4'h5;
        B_numbers[0] = 4'h6;
        B_numbers[1] = 4'hA;
        B_numbers[2] = 4'hE;
        B_numbers[3] = 4'h8;
        B_numbers[4] = 4'hA;
        correct_results[0] = 4'h3;
        correct_results[1] = 4'h8;
        correct_results[2] = 4'h9;
        correct_results[3] = 4'hF;
        correct_results[4] = 4'hF;
    end
  // --------------------------------- //
  // ---------- UUT Instantiation -------------- //
  //RCA RCA(input [3:0] X, [3:0] Y, C_i, output C_o, [3:0] S)
  //parametric_RCA (input C_i,[size-1:0] X, [size-1:0] Y,
     output C_o, [size-1:0] S);
  //CLA UUT(.c0(C_i),.X(X),.Y(Y),.cout(C_o),.sum(S));
  parametric_RCA UUT(.C_i(C_i),.X(X),.Y(Y),.C_o(C_o),.S(S));
  // ------- Test Procedure ------- //
  initial
  begin
    for(i=0;i<5;i=i+1)
    begin
```

```
44              X=A_numbers[i];
45              Y=B_numbers[i];
46        C_i=C[i];
47        #(wait_time);
48 //        $write("{X,+ Y ,C_i}=%d%d%d%d + %d%d%d%d, Ci= %d =>
     {C_o,S} = %d%d%d%d%d  --
     ",X[0],X[1],X[2],X[3],Y[0],Y[1],Y[2],Y[3],C_i,C_o,S[0],S[1],S[2],S[3]);

49        $write("{X+ Y ,C_i}=%b + %b, Ci= %b => {C_o,S} = %b%b --
            ",X,Y,C_i,C_o, S );
50        if( {C_o, S} == {C_out[i],correct_results[i]})
51          $display("TRUE");
52        else
53          $display("FALSE");
54      end
55      $finish();
56    end
57    // ------------------------------- //
58 endmodule
59 endmodule
```

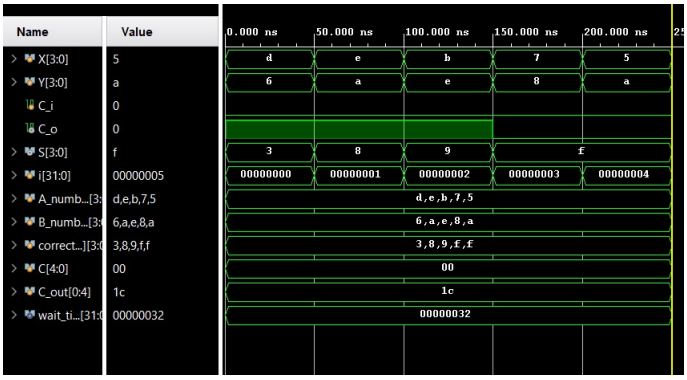


Figure 18: Results of simulation of parameter$_R CA module$

Listing 11: TCL Results of Testbench Code

```
# run 1000ns
{X+ Y ,C_i}=1101 + 0110, Ci= 0 => {C_o,S} = 10011 -- TRUE
{X+ Y ,C_i}=1110 + 1010, Ci= 0 => {C_o,S} = 11000 -- TRUE
{X+ Y ,C_i}=1011 + 1110, Ci= 0 => {C_o,S} = 11001 -- TRUE
{X+ Y ,C_i}=0111 + 1000, Ci= 0 => {C_o,S} = 01111 -- TRUE
{X+ Y ,C_i}=0101 + 1010, Ci= 0 => {C_o,S} = 01111 -- TRUE
```

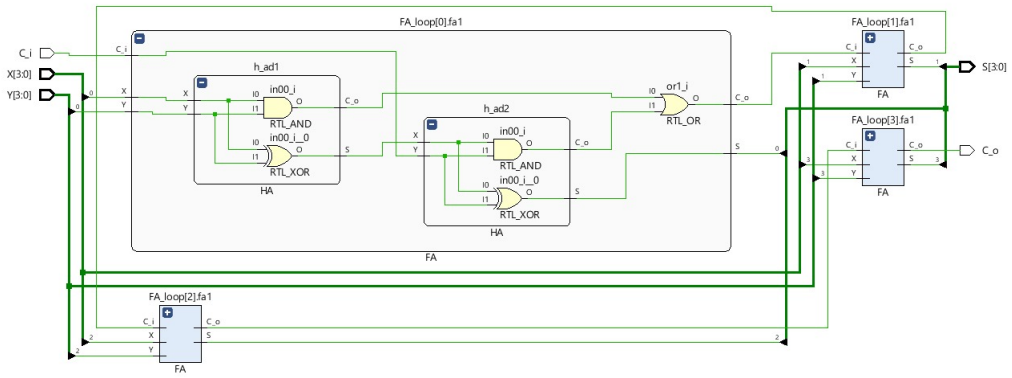


Figure 19: Rtl schematics of parameter RCA module

```
## LEDs
set_property -dict { PACKAGE_PIN H17   IOSTANDARD LVCMOS33 } [get_ports { S[0] }]; #IO_L18P_T2_A24_15 Sch=led[0]
set_property -dict { PACKAGE_PIN K15   IOSTANDARD LVCMOS33 } [get_ports { S[1] }]; #IO_L24P_T3_RS1_15 Sch=led[1]
set_property -dict { PACKAGE_PIN J13   IOSTANDARD LVCMOS33 } [get_ports { S[2] }]; #IO_L17N_T2_A25_15 Sch=led[2]
set_property -dict { PACKAGE_PIN N14   IOSTANDARD LVCMOS33 } [get_ports { S[3] }]; #IO_L8P_T1_D11_14 Sch=led[3]
set_property -dict { PACKAGE_PIN R18   IOSTANDARD LVCMOS33 } [get_ports { C_o }]; #IO_L7P_T1_D09_14 Sch=led[4]
##Switches
set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 } [get_ports { X[0] }]; #IO_L24N_T3_RS0_15 Sch=sw[0]
set_property -dict { PACKAGE_PIN L16   IOSTANDARD LVCMOS33 } [get_ports { X[1] }]; #IO_L3N_T0_DQS_EMCCLK_14 Sch=sw[1]
set_property -dict { PACKAGE_PIN M13   IOSTANDARD LVCMOS33 } [get_ports { X[2] }]; #IO_L6N_T0_D08_VREF_14 Sch=sw[2]
set_property -dict { PACKAGE_PIN R15   IOSTANDARD LVCMOS33 } [get_ports { X[3] }]; #IO_L13N_T2_MRCC_14 Sch=sw[3]
set_property -dict { PACKAGE_PIN R17   IOSTANDARD LVCMOS33 } [get_ports { Y[0] }]; #IO_L12N_T1_MRCC_14 Sch=sw[4]
set_property -dict { PACKAGE_PIN T18   IOSTANDARD LVCMOS33 } [get_ports { Y[1] }]; #IO_L7N_T1_D10_14 Sch=sw[5]
set_property -dict { PACKAGE_PIN U18   IOSTANDARD LVCMOS33 } [get_ports { Y[2] }]; #IO_L17N_T2_A13_D29_14 Sch=sw[6]
set_property -dict { PACKAGE_PIN R13   IOSTANDARD LVCMOS33 } [get_ports { Y[3] }]; #IO_L5N_T0_D07_14 Sch=sw[7]
##Buttons
set_property -dict { PACKAGE_PIN N17   IOSTANDARD LVCMOS33 } [get_ports { C_i }]; #IO_L9P_T1_DQS_14 Sch=btnc
```

Figure 20: Constrain File for RCA Module

]

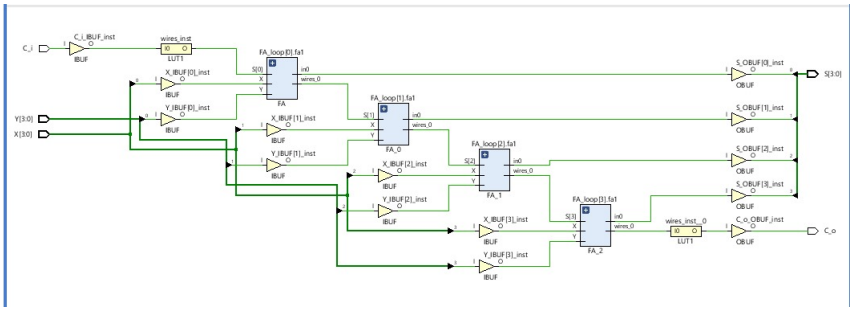


Figure 21: Technology schematics of RCA module

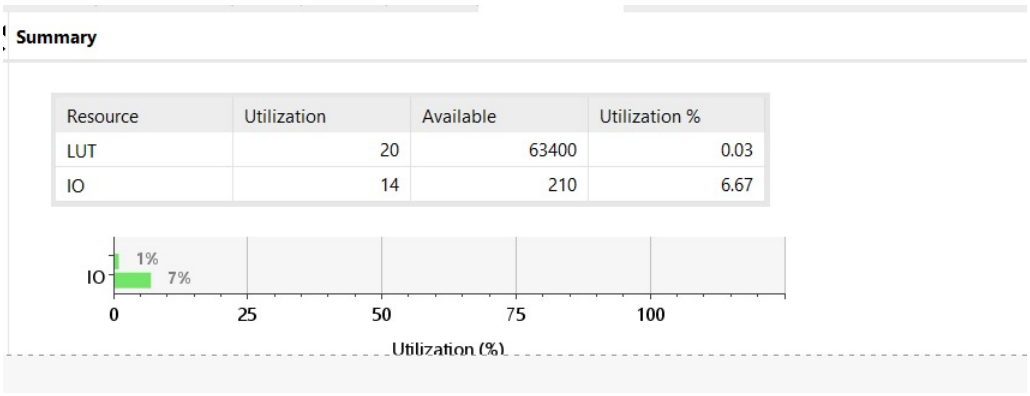


Figure 22: Utilization report of RCA module

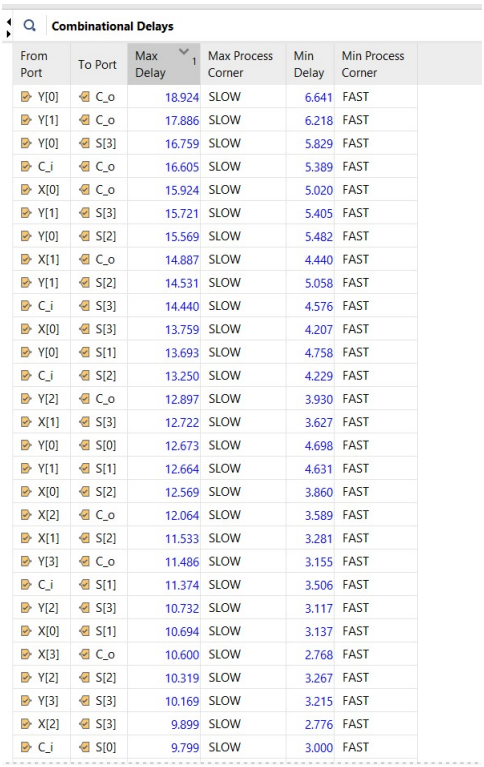


Figure 23: Combinational delay of RCA module

When I used "for" structure in this module actually I didn't change number of i/o and luts, so utilization report, rtl and technology schematic are same for this modules. But I saw the change of delay times. In my opinion, when i used "for" structure, max delay times was decreasing. It can be thought that it increases performance use "for" structure.
Then, i generate bit file and saved it for programming on FPGA.

1.3.3 Comments For Results

We made two different circuit implementations in the RCA circuit. As a result, while the Lot numbers remain the same, the Times change. Since I used the For structure when I specified the parameter, the time delay was slightly higher, but the input and output ports where the delays are located are equally correct from Y[0] to Cout.

1.4 Case 2: Parameter Size=8

Afterwards, I tested the module again by setting the parameter to 8 bits. Since I adjusted the module according to the parameter, I updated the circuit by only changing the size. Accordingly, I rewrote the constrain file and testbench code and examined the results. In the constrain file, I provided the X and Y inputs with switches, Cin with the button, and all the outputs with LEDs.

Listing 12: arithmetic_circutis.v -Parametrice RCA (parameter size 8) module

```
module parametric_RCA #( parameter size = 8)( input
    C_i,[size-1:0] X, [size-1:0] Y,   output C_o, [size-1:0]
    S);
(* dont_touch="true" *) wire x_wires=X;
(* dont_touch="true" *) wire y_wires=Y;
(* dont_touch="true" *) wire Ci_wires=C_i;
(* dont_touch="true" *) wire Co_wires=C_o;
(* dont_touch="true" *) wire S_wires=S;
(* dont_touch="true" *) wire [size:0] wires;
assign wires[0]=C_i;
assign C_o=wires[size];
genvar j; // temp loop variable, used only
 // in the evaluation of the generate blocks
generate
for( j=0; j<size; j=j+1 )
 begin : FA_loop
 FA fa1(X[j],Y[j],wires[j], wires[j+1], S[j]);
 end
endgenerate
endmodule
```

Listing 13: parametric_RCA_tb.v 8 bit testbench module

```
`timescale 1ns / 1ps
module RCA_tb2();
  reg [7:0] X;
  reg [7:0] Y;
  reg C_i;
  wire C_o;
```

```
7   wire [7:0] S;
8   parameter wait_time = 100;
9   integer i;
10  reg [0:7] A_numbers[4:0];
11  reg [0:7] B_numbers [4:0];
12  reg [0:7] correct_results[4:0];
13  reg [4:0] C= 5'd10101;
14  reg[0:4] C_out= 5'b01101;
15  initial begin
16        A_numbers[0] = 8'h1D; A_numbers[1] = 8'h96;
             A_numbers[2] = 8'hfb; A_numbers[3] = 8'h17;
             A_numbers[4] = 8'hFE;
17        B_numbers[0] = 8'h26; B_numbers[1] = 8'h88;
             B_numbers[2] = 8'h1E; B_numbers[3] = 8'h28;
             B_numbers[4] = 8'h3;
18        correct_results[0] = 8'h44; correct_results[1] =
             8'h1E; correct_results[2] = 8'h1a;
             correct_results[3] = 8'h3F; correct_results[4] =
             8'h2;
19    end
20
21  // --------------------------------- //
22
23  // ---------- UUT Instantiation -------------- //
24  //RCA
25  parametric_RCA #(8) UUT
       (.C_i(C_i),.X(X),.Y(Y),.C_o(C_o),.S(S));
26
27  // ------- Test Procedure ------- //
28  initial
29  begin
30    for(i=0;i<5;i=i+1)
31    begin
32            X=A_numbers[i];
33            Y=B_numbers[i];
34      C_i=C[i];
35      #(wait_time);
36//      $write("{X,+ Y ,C_i}=%d%d%d%d + %d%d%d%d, Ci= %d =>
    {C_o,S} = %d%d%d%d%d  --
    ",X[0],X[1],X[2],X[3],Y[0],Y[1],Y[2],Y[3],C_i,C_o,S[0],S[1],S[2],S[3]);
37      //$write("!!!!!!{X,+ Y ,C_i}=%b + %b, Ci= %b => {C_o,S}
          = %b %b %b %b, Cout= %b correct_result:  %b -- ", X,
          Y, C_i, C_o, S[0], S[1], S[2], S[3], C_out[i],
          correct_results[i]);
38      $write("{X,+ Y ,C_i}=%b + %b, Ci= %b => {C_o,S} = %b%b
```

```
          -- ",X,Y,C_i,C_o, S );
39      if( {C_o, S} == {C_out[i],correct_results[i]})
40        $display("TRUE");
41      else
42        $display("FALSE");
43    end
44    $finish();
45  end
46  // ------------------------------- //
47 endmodule
```

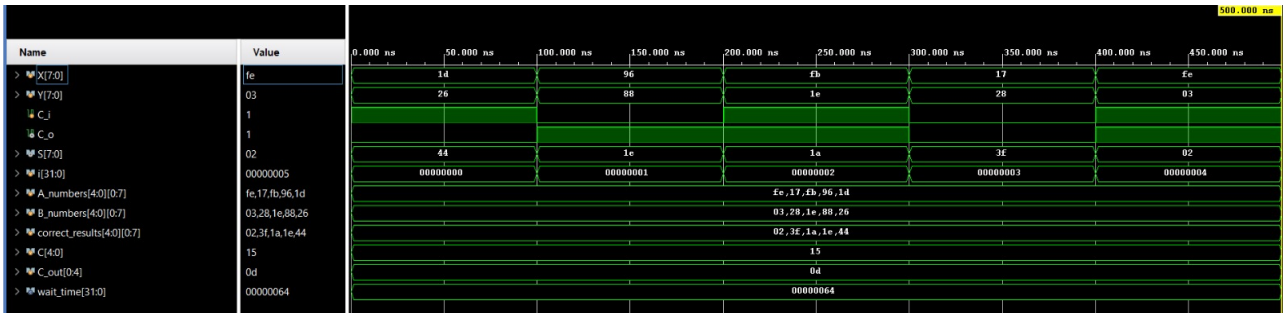


Figure 24: Results of simulation of parameter$_R CA module - 8bit$

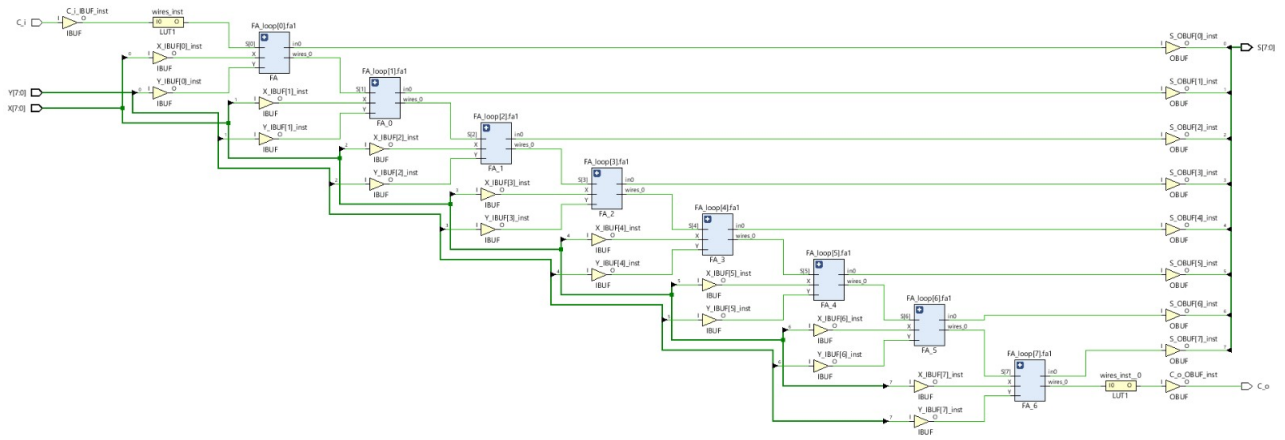


Figure 25: Technology schematic of parameter$_R CA module - 8bit$

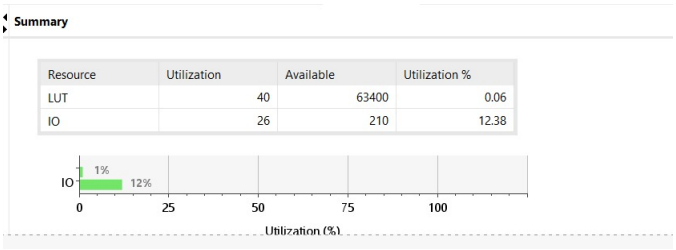
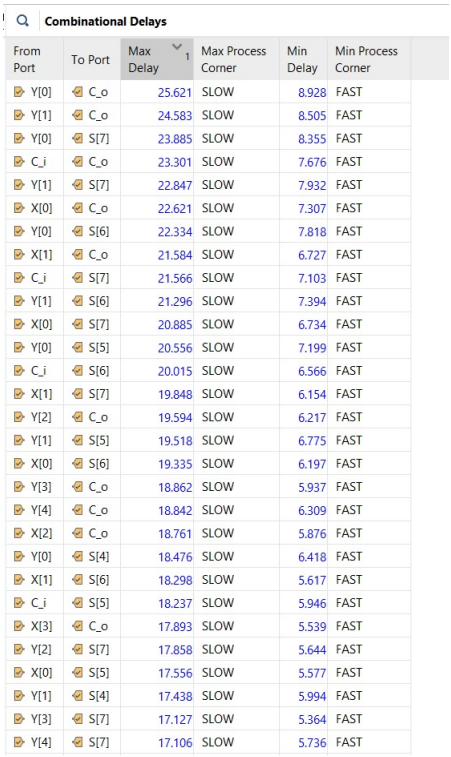


Figure 26: UTilization report of parameter$_R CA module - 8bit$



From Port	To Port	Max Delay	Max Process Corner	Min Delay	Min Process Corner
Y[0]	C_o	25.621	SLOW	8.928	FAST
Y[1]	C_o	24.583	SLOW	8.505	FAST
Y[0]	S[7]	23.885	SLOW	8.355	FAST
C_i	C_o	23.301	SLOW	7.676	FAST
Y[1]	S[7]	22.847	SLOW	7.932	FAST
X[0]	C_o	22.621	SLOW	7.307	FAST
Y[0]	S[6]	22.334	SLOW	7.818	FAST
X[1]	C_o	21.584	SLOW	6.727	FAST
C_i	S[7]	21.566	SLOW	7.103	FAST
Y[1]	S[6]	21.296	SLOW	7.394	FAST
X[0]	S[7]	20.885	SLOW	6.734	FAST
Y[0]	S[5]	20.556	SLOW	7.199	FAST
C_i	S[6]	20.015	SLOW	6.566	FAST
X[1]	S[7]	19.848	SLOW	6.154	FAST
Y[2]	C_o	19.594	SLOW	6.217	FAST
Y[1]	S[5]	19.518	SLOW	6.775	FAST
X[0]	S[6]	19.335	SLOW	6.197	FAST
Y[3]	C_o	18.862	SLOW	5.937	FAST
Y[4]	C_o	18.842	SLOW	6.309	FAST
X[2]	C_o	18.761	SLOW	5.876	FAST
Y[0]	S[4]	18.476	SLOW	6.418	FAST
X[1]	S[6]	18.298	SLOW	5.617	FAST
C_i	S[5]	18.237	SLOW	5.946	FAST
X[3]	C_o	17.893	SLOW	5.539	FAST
Y[2]	S[7]	17.858	SLOW	5.644	FAST
X[0]	S[5]	17.556	SLOW	5.577	FAST
Y[1]	S[4]	17.438	SLOW	5.994	FAST
Y[3]	S[7]	17.127	SLOW	5.364	FAST
Y[4]	S[7]	17.106	SLOW	5.736	FAST

Figure 27: Timing Report of parameter$_R CA module - 8bit$

Listing 14: TCL Results of Testbench Code

```
Vivado Simulator 2020.1
Time resolution is 1 ps
{X,+ Y ,C_i}=00011101 + 00100110, Ci= 1 => {C_o,S} = 001000100 -- TRUE
{X,+ Y ,C_i}=10010110 + 10001000, Ci= 0 => {C_o,S} = 100011110 -- TRUE
{X,+ Y ,C_i}=11111011 + 00011110, Ci= 1 => {C_o,S} = 100011010 -- TRUE
{X,+ Y ,C_i}=00010111 + 00101000, Ci= 0 => {C_o,S} = 000111111 -- TRUE
{X,+ Y ,C_i}=11111110 + 00000011, Ci= 1 => {C_o,S} = 100000010 -- TRUE
$finish called at time : 500 ns : File
```

As seen in the simulation results, the RCA module worked correctly for 8 bits.

```
##Switches
set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 } [get_ports { X[0] }]; #IO_L24N_T3_RS0_15 Sch=sw[0]
set_property -dict { PACKAGE_PIN L16   IOSTANDARD LVCMOS33 } [get_ports { X[1] }]; #IO_L3N_T0_DQS_EMCCLK_14 Sch=sw[1]
set_property -dict { PACKAGE_PIN M13   IOSTANDARD LVCMOS33 } [get_ports { X[2] }]; #IO_L6N_T0_D08_VREF_14 Sch=sw[2]
set_property -dict { PACKAGE_PIN R15   IOSTANDARD LVCMOS33 } [get_ports { X[3] }]; #IO_L13N_T2_MRCC_14 Sch=sw[3]
set_property -dict { PACKAGE_PIN R17   IOSTANDARD LVCMOS33 } [get_ports { X[4]  }]; #IO_L12N_T1_MRCC_14 Sch=sw[4]
set_property -dict { PACKAGE_PIN T18   IOSTANDARD LVCMOS33 } [get_ports { X[5]  }]; #IO_L7N_T1_D10_14 Sch=sw[5]
set_property -dict { PACKAGE_PIN U18   IOSTANDARD LVCMOS33 } [get_ports { X[6]  }]; #IO_L17N_T2_A13_D29_14 Sch=sw[6]
set_property -dict { PACKAGE_PIN R13   IOSTANDARD LVCMOS33 } [get_ports { X[7]  }]; #IO_L5N_T0_D07_14 Sch=sw[7]
set_property -dict { PACKAGE_PIN T8    IOSTANDARD LVCMOS18 } [get_ports { Y[0]}]; #IO_L24N_T3_34 Sch=sw[8]
set_property -dict { PACKAGE_PIN U8    IOSTANDARD LVCMOS18 } [get_ports { Y[1]}]; #IO_25_34 Sch=sw[9]
set_property -dict { PACKAGE_PIN R16   IOSTANDARD LVCMOS33 } [get_ports { Y[2] }]; #IO_L15P_T2_DQS_RDWR_B_14 Sch=sw[10]
set_property -dict { PACKAGE_PIN T13   IOSTANDARD LVCMOS33 } [get_ports { Y[3] }]; #IO_L23P_T3_A03_D19_14 Sch=sw[11]
set_property -dict { PACKAGE_PIN H6    IOSTANDARD LVCMOS33 } [get_ports { Y[4] }]; #IO_L24P_T3_35 Sch=sw[12]
set_property -dict { PACKAGE_PIN U12   IOSTANDARD LVCMOS33 } [get_ports { Y[5] }]; #IO_L20P_T3_A08_D24_14 Sch=sw[13]
set_property -dict { PACKAGE_PIN U11   IOSTANDARD LVCMOS33 } [get_ports { Y[6] }]; #IO_L19N_T3_A09_D25_VREF_14 Sch=sw[14]
set_property -dict { PACKAGE_PIN V10   IOSTANDARD LVCMOS33 } [get_ports { Y[7] }]; #IO_L21P_T3_DQS_14 Sch=sw[15]

## LEDs
set_property -dict { PACKAGE_PIN H17   IOSTANDARD LVCMOS33 } [get_ports { S[0] }]; #IO_L18P_T2_A24_15 Sch=led[0]
set_property -dict { PACKAGE_PIN K15   IOSTANDARD LVCMOS33 } [get_ports { S[1] }]; #IO_L24P_T3_RS1_15 Sch=led[1]
set_property -dict { PACKAGE_PIN J13   IOSTANDARD LVCMOS33 } [get_ports { S[2] }]; #IO_L17N_T2_A25_15 Sch=led[2]
set_property -dict { PACKAGE_PIN N14   IOSTANDARD LVCMOS33 } [get_ports { S[3] }]; #IO_L8P_T1_D11_14 Sch=led[3]
set_property -dict { PACKAGE_PIN R18   IOSTANDARD LVCMOS33 } [get_ports { S[4]}]; #IO_L7P_T1_D09_14 Sch=led[4]
set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports { S[5] }]; #IO_L18N_T2_A11_D27_14 Sch=led[5]
set_property -dict { PACKAGE_PIN U17   IOSTANDARD LVCMOS33 } [get_ports { S[6] }]; #IO_L17P_T2_A14_D30_14 Sch=led[6]
set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports { S[7] }]; #IO_L18P_T2_A12_D28_14 Sch=led[7]
set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 } [get_ports { C_o }]; #IO_L16N_T2_A15_D31_14 Sch=led[8]

##Buttons
set_property -dict { PACKAGE_PIN N17   IOSTANDARD LVCMOS33 } [get_ports { C_i }]; #IO_L9P_T1_DQS_14 Sch=btnc
```

Figure 28: Constrain File for 8-bit RCA Module

]

1.5 Carry Lookahead Adder

I did research before creating the CLA module. I researched the propogate and generate functions and tried to explain this circuit by drawing it on paper. I then created the module as requested. I simulated it by writing the testbench code that included all the inputs the module could receive and compared the outputs with my expectations. For example, when I added 15 and 1 when Cin was 0, Cout was 1 and Sum was 00, or when I added 8 and 9, Cout was 1 and Sum was 1. Since the simulation result was correct, I looked at the RTL analysis and saw that the generate and propagete functions connected with xor and and gates were as I expected, as I wrote on the paper.

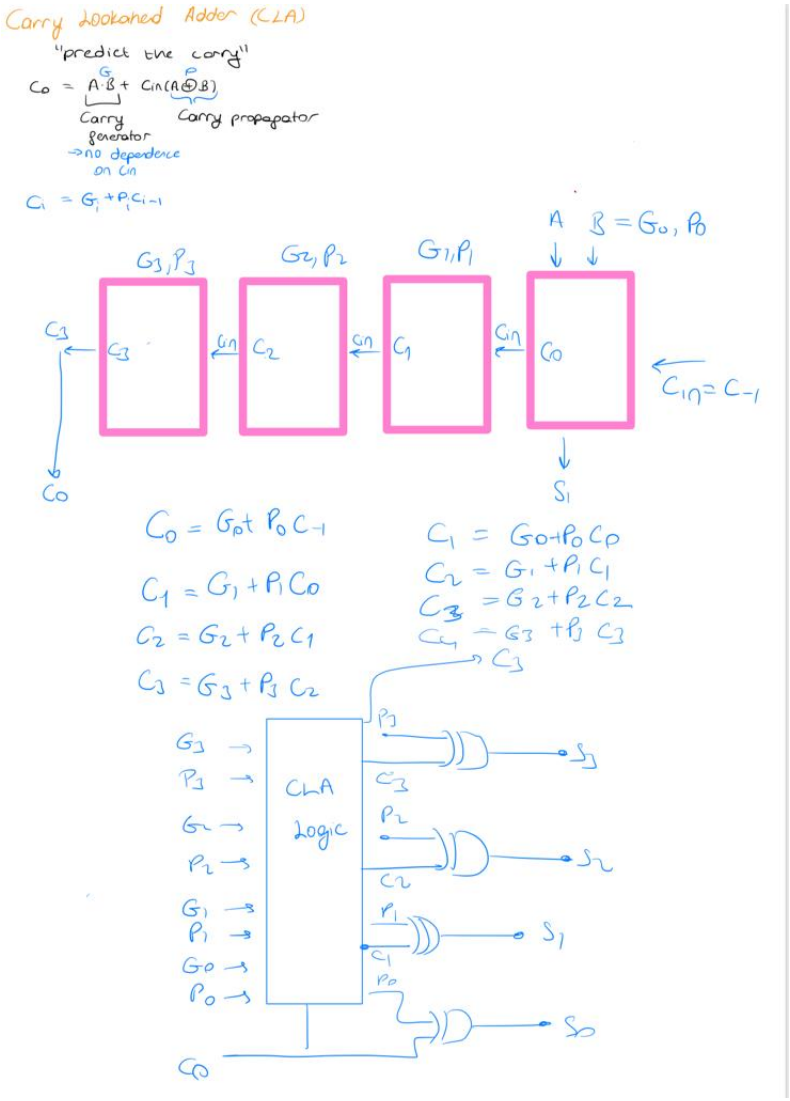


Figure 29: hand drawn version of the CLA module

Listing 15: arithmetic_circutis.v - CLA module

```
module CLA (input C_i,[3:0] X,Y,
  output C_o,[3:0] S);
(* dont_touch="true" *) wire x_wires=X;
(* dont_touch="true" *) wire y_wires=Y;
(* dont_touch="true" *) wire Ci_wires=C_i;
(* dont_touch="true" *) wire Co_wires=C_o;
(* dont_touch="true" *) wire S_wires=S;
(* dont_touch="true" *)   wire [3:0] g_wires;
(* dont_touch="true" *)   wire [3:0] p_wires;
(* dont_touch="true" *)   wire [4:0] c_wires;

  assign c_wires[0] = C_i;
```

```
  assign C_o = c_wires[4];

  genvar j;
  generate
    for (j = 0; j < 4; j = j + 1) begin : GandP_loop
      assign g_wires[j] = X[j] & Y[j];
      assign p_wires[j] = X[j] ^ Y[j];
      assign S[j] = p_wires[j] ^ c_wires[j];
      assign c_wires[j + 1] = g_wires[j] | (p_wires[j] &
         c_wires[j]);
    end
  endgenerate
endmodule
```

Listing 16: CLA_tb.v testbench module

```
`timescale 1ns / 1ps
module CLA_tb();
  // ----- Inputs & Outputs
  reg [3:0] X;
  reg [3:0] Y;
  reg C_i;
  wire C_o;
  wire [3:0] S;
  // ----- Testbench Parameters ----- //
  parameter wait_time = 30;
  reg [7:0] i;
  // ---------- UUT Instantiation -------------- //
  //RCA RCA(input [3:0] X, [3:0] Y, C_i, output C_o, [3:0] S)
  //parametric_RCA (input C_i,[size-1:0] X, [size-1:0] Y,
     output C_o, [size-1:0] S);
  //parametric_RCA UUT(.C_i(C_i),.X(X),.Y(Y),.C_o(C_o),.S(S));
  CLA UUT(.C_i(C_i),.X(X),.Y(Y),.C_o(C_o),.S(S));
  // ------- Test Procedure ------- //
  initial
  begin
  for(i=0;i<10;i=i+1)
    begin
    {X,Y, C_i}= i;
      #(10);
    $write(" {X+ Y }=%b + %b, Ci= %b => {C_o,S} = %b%b --
       ",X,Y,C_i,C_o, S );
    $display("");
    #(wait_time);
    end
  end
```

```
29 endmodule
```

In the testbench I wrote to observe all the values at the inputs, it shows 1000 ns at the first start of the simulation. We can confirm all the situations ourselves by scrolling through the time bar. Likewise, I printed all the results in the TCL console; I took a particular section as an example to preserve the readability of both. The results gave the sum of two 4-bit numbers, as I expected in CLA.

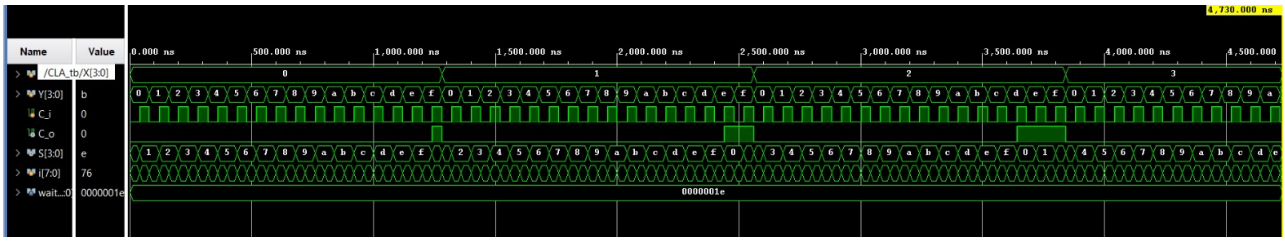


Figure 30: Results of simulation of Cla module

Listing 17: TCL Results of Testbench Code

```
Vivado Simulator 2020.1
Time resolution is 1 ps
 {X+ Y }=0000 + 0000, Ci= 0 => {C_o,S} = 00000 --
 {X+ Y }=0000 + 0000, Ci= 1 => {C_o,S} = 00001 --
 {X+ Y }=0000 + 0001, Ci= 0 => {C_o,S} = 00001 --
 {X+ Y }=0000 + 0001, Ci= 1 => {C_o,S} = 00010 --
 {X+ Y }=0000 + 0010, Ci= 0 => {C_o,S} = 00010 --
 {X+ Y }=0000 + 0010, Ci= 1 => {C_o,S} = 00011 --
 {X+ Y }=0000 + 0011, Ci= 0 => {C_o,S} = 00011 --
 {X+ Y }=0000 + 0011, Ci= 1 => {C_o,S} = 00100 --
 {X+ Y }=0000 + 0100, Ci= 0 => {C_o,S} = 00100 --
 {X+ Y }=0000 + 0100, Ci= 1 => {C_o,S} = 00101 --
 {X+ Y }=0000 + 0101, Ci= 0 => {C_o,S} = 00101 --
 {X+ Y }=0000 + 0101, Ci= 1 => {C_o,S} = 00110 --
 {X+ Y }=0000 + 0110, Ci= 0 => {C_o,S} = 00110
```

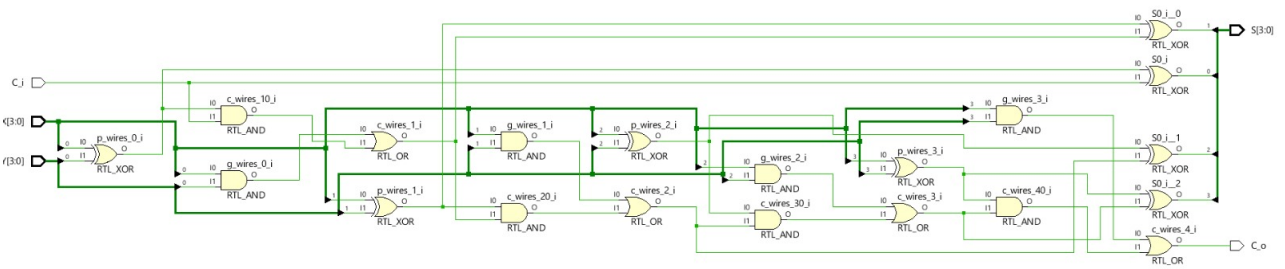


Figure 31: Rtl schematics of parameter CLA module

```
## LEDs
set_property -dict { PACKAGE_PIN H17   IOSTANDARD LVCMOS33 } [get_ports { S[0] }]; #IO_L18P_T2_A24_15 Sch=led[0]
set_property -dict { PACKAGE_PIN K15   IOSTANDARD LVCMOS33 } [get_ports { S[1] }]; #IO_L24P_T3_RS1_15 Sch=led[1]
set_property -dict { PACKAGE_PIN J13   IOSTANDARD LVCMOS33 } [get_ports { S[2] }]; #IO_L17N_T2_A25_15 Sch=led[2]
set_property -dict { PACKAGE_PIN N14   IOSTANDARD LVCMOS33 } [get_ports { S[3] }]; #IO_L8P_T1_D11_14 Sch=led[3]
set_property -dict { PACKAGE_PIN R18   IOSTANDARD LVCMOS33 } [get_ports { C_o }]; #IO_L7P_T1_D09_14 Sch=led[4]
##Switches
set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 } [get_ports { X[0] }]; #IO_L24N_T3_RS0_15 Sch=sw[0]
set_property -dict { PACKAGE_PIN L16   IOSTANDARD LVCMOS33 } [get_ports { X[1] }]; #IO_L3N_T0_DQS_EMCCLK_14 Sch=sw[1]
set_property -dict { PACKAGE_PIN M13   IOSTANDARD LVCMOS33 } [get_ports { X[2] }]; #IO_L6N_T0_D08_VREF_14 Sch=sw[2]
set_property -dict { PACKAGE_PIN R15   IOSTANDARD LVCMOS33 } [get_ports { X[3] }]; #IO_L13N_T2_MRCC_14 Sch=sw[3]
set_property -dict { PACKAGE_PIN R17   IOSTANDARD LVCMOS33 } [get_ports { Y[0] }]; #IO_L12N_T1_MRCC_14 Sch=sw[4]
set_property -dict { PACKAGE_PIN T18   IOSTANDARD LVCMOS33 } [get_ports { Y[1] }]; #IO_L7N_T1_D10_14 Sch=sw[5]
set_property -dict { PACKAGE_PIN U18   IOSTANDARD LVCMOS33 } [get_ports { Y[2] }]; #IO_L17N_T2_A13_D29_14 Sch=sw[6]
set_property -dict { PACKAGE_PIN R13   IOSTANDARD LVCMOS33 } [get_ports { Y[3] }]; #IO_L5N_T0_D07_14 Sch=sw[7]
##Buttons
set_property -dict { PACKAGE_PIN N17   IOSTANDARD LVCMOS33 } [get_ports { C_i }]; #IO_L9P_T1_DQS_14 Sch=btnc
```

Figure 32: Constrain File for CLA Module

]

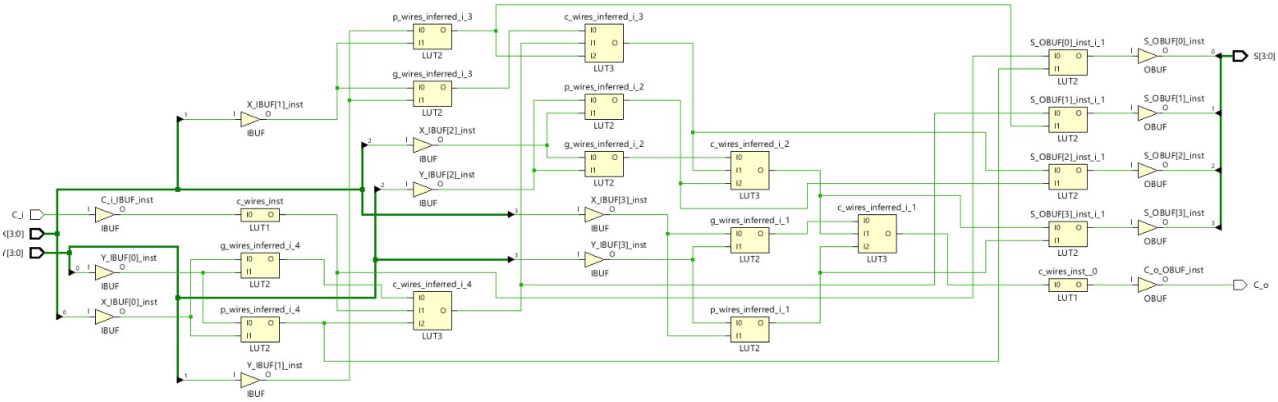


Figure 33: Technology schematics of CLA module

Summary

Resource	Utilization	Available	Utilization %
LUT	18	63400	0.03
IO	14	210	6.67

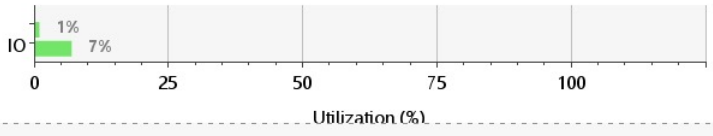
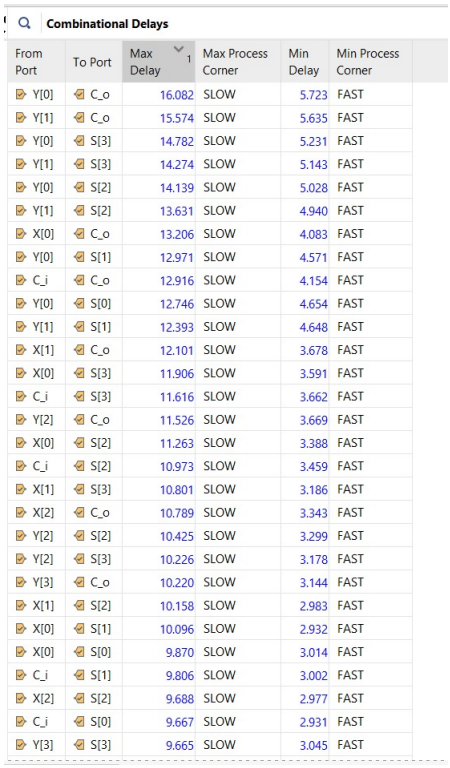


Figure 34: Utilization report of CLA module



Combinational Delays

From Port	To Port	Max Delay	Max Process Corner	Min Delay	Min Process Corner
Y[0]	C_o	16.082	SLOW	5.723	FAST
Y[1]	C_o	15.574	SLOW	5.635	FAST
Y[0]	S[3]	14.782	SLOW	5.231	FAST
Y[1]	S[3]	14.274	SLOW	5.143	FAST
Y[0]	S[2]	14.139	SLOW	5.028	FAST
Y[1]	S[2]	13.631	SLOW	4.940	FAST
X[0]	C_o	13.206	SLOW	4.083	FAST
Y[0]	S[1]	12.971	SLOW	4.571	FAST
C_i	C_o	12.916	SLOW	4.154	FAST
Y[0]	S[0]	12.746	SLOW	4.654	FAST
Y[1]	S[1]	12.393	SLOW	4.648	FAST
X[1]	C_o	12.101	SLOW	3.678	FAST
X[0]	S[3]	11.906	SLOW	3.591	FAST
C_i	S[3]	11.616	SLOW	3.662	FAST
Y[2]	C_o	11.526	SLOW	3.669	FAST
X[0]	S[2]	11.263	SLOW	3.388	FAST
C_i	S[2]	10.973	SLOW	3.459	FAST
X[1]	S[3]	10.801	SLOW	3.186	FAST
X[2]	C_o	10.789	SLOW	3.343	FAST
Y[2]	S[2]	10.425	SLOW	3.299	FAST
Y[2]	S[3]	10.226	SLOW	3.178	FAST
Y[3]	C_o	10.220	SLOW	3.144	FAST
X[1]	S[2]	10.158	SLOW	2.983	FAST
X[0]	S[1]	10.096	SLOW	2.932	FAST
X[0]	S[0]	9.870	SLOW	3.014	FAST
C_i	S[1]	9.806	SLOW	3.002	FAST
X[2]	S[2]	9.688	SLOW	2.977	FAST
C_i	S[0]	9.667	SLOW	2.931	FAST
Y[3]	S[3]	9.665	SLOW	3.045	FAST

Figure 35: Combinational delay of CLA module

1.5.1 Comments For Results

Propoaget and generate functions are used in the CLA circuit. In this way, the collector circuit has actually been realized with less lut. In this circuit, where 18 Lut was used, the maximum delay was 16.082 from Y[0] to Cout, from the beginning to the end.

1.6 Adder-Subtractor Circuit with Overflow detection

For the last circuit, I first did research and tried to understand the purpose of the circuit in my own way. Later, I realized that an XOR gate should be placed in the question marked parts of the circuit shown in the picture in the report. When Ci is 1, the circuit acts as a subtractor, and when Ci = 0, the circuit acts as an adder. It achieves this in the following way: In fact, when Ci is one, it takes the 2's complement of B (B(XOR)1 +(C=)1) and accordingly, it now works as a subtractor.
Overflow, which is the V output, detects the change in the last bit of the circuit. If there is a changer in the last bit, overflow occurs because the last bit is not a number, it indicates the signed status, so it should not change with the carry values.
After understanding the circuit, I wrote the Verilog code to implement it and tested all situations with testbench as requested.

Listing 18: arithmetic_circutis.v - Add_Sub module

```
module Add_Sub(input C_i, [3:0] X, [3:0] Y, output C_o, V,
    [3:0] S);
(* dont_touch="true" *) wire x2_wires=X;
(* dont_touch="true" *) wire y_wires=Y;
(* dont_touch="true" *) wire y_wires=V;
(* dont_touch="true" *) wire Ci_wires=C_i;
(* dont_touch="true" *) wire Co_wires=C_o;
(* dont_touch="true" *) wire S_wires=S;
(* dont_touch="true" *) wire [3:0] x_wires;
(* dont_touch="true" *) wire [4:0] c_wires;
assign V= c_wires[3]^c_wires[4];
assign c_wires[0]=C_i;
assign C_o=c_wires[4];
genvar j; // temp loop variable, used only
 // in the evaluation of the generate blocks
generate
for( j=0; j<4; j=j+1 )
 begin : ASCw0
 //and and1(g_wires[j], X[j], Y[j]);
 assign x_wires[j]= Y[j]^ C_i ;
 FA fa1(x_wires[j],X[j],c_wires[j],c_wires[j+1], S[j]);
 end
endgenerate
endmodule
```

Listing 19: Add_Subtb.v testbench module

```
`timescale 1ns / 1ps
module ASC_tb();
  // ----- Inputs & Outputs
  reg [3:0] X;
  reg [3:0] Y;
  reg C_i;
  wire C_o;
  wire V;
  wire [3:0] S;
  // ----- Testbench Parameters ----- //
  parameter wait_time = 30;
  reg [7:0] i;
  Add_Sub UUT(.C_i(C_i),.X(X),.Y(Y),.C_o(C_o),.V(V), .S(S));
  // ------- Test Procedure ------- //
  initial
  begin
  for(i=0;i<512;i=i+1)
    begin
    {X,Y, C_i}= i;
```

```
20        #(10);
21      $write(" {X+ Y }=%b + %b, Ci= %b => C_o:%b,V:%b,S:%b --
            ",X,Y,C_i,C_o,V,S );
22      $display("");
23      #(wait_time);
24      end
25    end
26  endmodule
```

In the testbench I wrote to observe all the values at the inputs, it shows 1000 ns at the first start of the simulation. We can confirm all the situations ourselves by scrolling through the time bar. Likewise, I printed all the results in the TCL console; I took a particular section as an example to preserve the readability of both. The results gave the sum of two 4-bit numbers, as I expected in CLA.

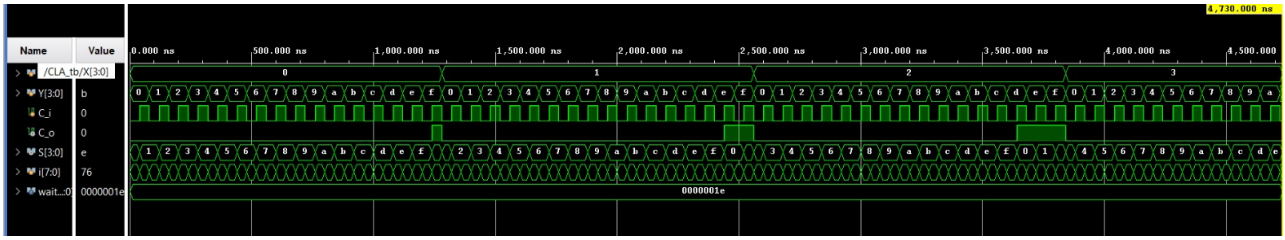


Figure 36: Results of simulation of Cla module

Listing 20: TCL Results of Testbench Code

```
Vivado Simulator 2020.1
Time resolution is 1 ps
 {X+ Y }=0000 + 0000, Ci= 0 => {C_o,S} = 00000 --
 {X+ Y }=0000 + 0000, Ci= 1 => {C_o,S} = 00001 --
 {X+ Y }=0000 + 0001, Ci= 0 => {C_o,S} = 00001 --
 {X+ Y }=0000 + 0001, Ci= 1 => {C_o,S} = 00010 --
 {X+ Y }=0000 + 0010, Ci= 0 => {C_o,S} = 00010 --
 {X+ Y }=0000 + 0010, Ci= 1 => {C_o,S} = 00011 --
 {X+ Y }=0000 + 0011, Ci= 0 => {C_o,S} = 00011 --
 {X+ Y }=0000 + 0011, Ci= 1 => {C_o,S} = 00100 --
 {X+ Y }=0000 + 0100, Ci= 0 => {C_o,S} = 00100 --
 {X+ Y }=0000 + 0100, Ci= 1 => {C_o,S} = 00101 --
 {X+ Y }=0000 + 0101, Ci= 0 => {C_o,S} = 00101 --
 {X+ Y }=0000 + 0101, Ci= 1 => {C_o,S} = 00110 --
 {X+ Y }=0000 + 0110, Ci= 0 => {C_o,S} = 00110
```

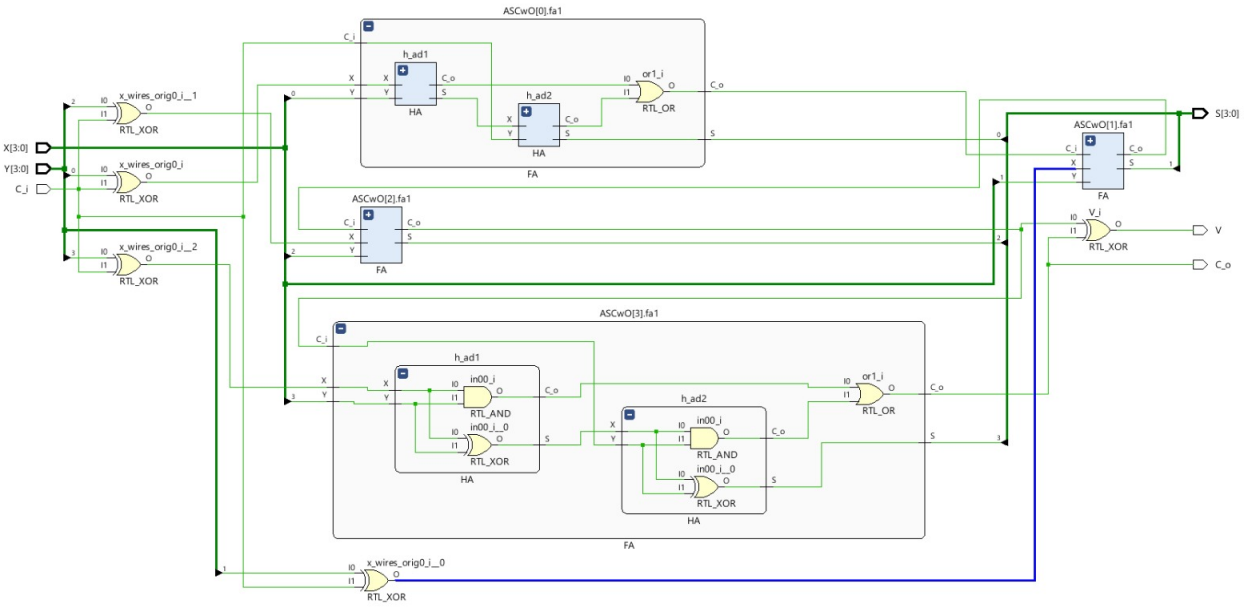


Figure 37: Rtl schematics of parameter Add_Sub module

```
## LEDs
set_property -dict { PACKAGE_PIN H17   IOSTANDARD LVCMOS33 } [get_ports { S[0] }]; #IO_L18P_T2_A24_15 Sch=led[0]
set_property -dict { PACKAGE_PIN K15   IOSTANDARD LVCMOS33 } [get_ports { S[1] }]; #IO_L24P_T3_RS1_15 Sch=led[1]
set_property -dict { PACKAGE_PIN J13   IOSTANDARD LVCMOS33 } [get_ports { S[2] }]; #IO_L17N_T2_A25_15 Sch=led[2]
set_property -dict { PACKAGE_PIN N14   IOSTANDARD LVCMOS33 } [get_ports { S[3] }]; #IO_L8P_T1_D11_14 Sch=led[3]
set_property -dict { PACKAGE_PIN R18   IOSTANDARD LVCMOS33 } [get_ports { C_o }]; #IO_L7P_T1_D09_14 Sch=led[4]
##Switches
set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 } [get_ports { X[0] }]; #IO_L24N_T3_RS0_15 Sch=sw[0]
set_property -dict { PACKAGE_PIN L16   IOSTANDARD LVCMOS33 } [get_ports { X[1] }]; #IO_L3N_T0_DQS_EMCCLK_14 Sch=sw[1]
set_property -dict { PACKAGE_PIN M13   IOSTANDARD LVCMOS33 } [get_ports { X[2] }]; #IO_L6N_T0_D08_VREF_14 Sch=sw[2]
set_property -dict { PACKAGE_PIN R15   IOSTANDARD LVCMOS33 } [get_ports { X[3] }]; #IO_L13N_T2_MRCC_14 Sch=sw[3]
set_property -dict { PACKAGE_PIN R17   IOSTANDARD LVCMOS33 } [get_ports { Y[0] }]; #IO_L12N_T1_MRCC_14 Sch=sw[4]
set_property -dict { PACKAGE_PIN T18   IOSTANDARD LVCMOS33 } [get_ports { Y[1] }]; #IO_L7N_T1_D10_14 Sch=sw[5]
set_property -dict { PACKAGE_PIN U18   IOSTANDARD LVCMOS33 } [get_ports { Y[2] }]; #IO_L17N_T2_A13_D29_14 Sch=sw[6]
set_property -dict { PACKAGE_PIN R13   IOSTANDARD LVCMOS33 } [get_ports { Y[3] }]; #IO_L5N_T0_D07_14 Sch=sw[7]
##Buttons
set_property -dict { PACKAGE_PIN N17   IOSTANDARD LVCMOS33 } [get_ports { C_i }]; #IO_L9P_T1_DQS_14 Sch=btnc
```

Figure 38: Constrain File for Add_Sub Module

]

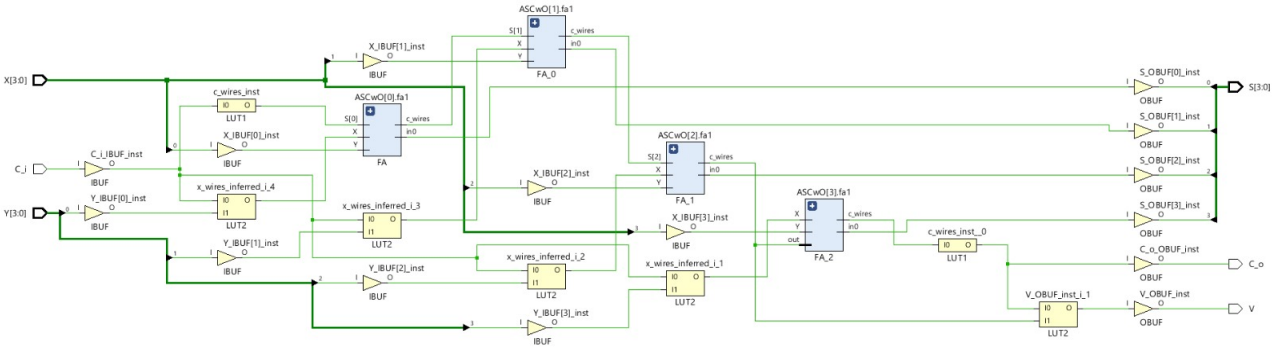


Figure 39: Technology schematics of Add_Sub module

Summary

Resource	Utilization	Available	Utilization %
LUT	24	63400	0.04
IO	15	210	7.14

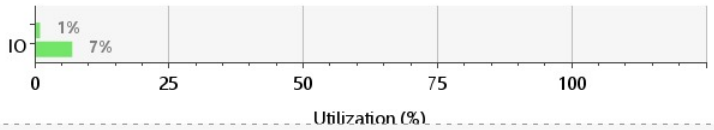


Figure 40: Utilization report of Add_Sub module

Combinational Delays

From Port	To Port	Max Delay	Max Process Corner	Min Delay	Min Process Corner
Y[0]	V	21.275	SLOW	6.593	FAST
Y[0]	C_o	20.678	SLOW	7.183	FAST
Y[1]	V	19.655	SLOW	6.085	FAST
Y[1]	C_o	19.058	SLOW	6.676	FAST
Y[0]	S[3]	18.761	SLOW	6.357	FAST
C_i	V	18.610	SLOW	3.677	FAST
C_i	C_o	18.013	SLOW	3.510	FAST
X[0]	V	17.685	SLOW	4.714	FAST
Y[1]	S[3]	17.141	SLOW	5.850	FAST
X[0]	C_o	17.088	SLOW	5.304	FAST
Y[0]	S[2]	16.754	SLOW	5.684	FAST
C_i	S[3]	16.096	SLOW	3.418	FAST
X[1]	V	15.266	SLOW	3.979	FAST
X[0]	S[3]	15.171	SLOW	4.478	FAST
Y[1]	S[2]	15.134	SLOW	5.176	FAST
Y[2]	V	14.918	SLOW	3.898	FAST
Y[0]	S[1]	14.858	SLOW	4.945	FAST
X[1]	C_o	14.669	SLOW	4.570	FAST
Y[2]	C_o	14.321	SLOW	4.488	FAST
Y[0]	S[0]	14.228	SLOW	5.168	FAST
C_i	S[2]	14.088	SLOW	3.667	FAST
Y[1]	S[1]	13.560	SLOW	4.929	FAST
X[2]	V	13.435	SLOW	3.266	FAST
X[0]	S[2]	13.163	SLOW	3.805	FAST
Y[3]	V	13.000	SLOW	3.796	FAST
X[2]	C_o	12.838	SLOW	3.857	FAST
X[1]	S[3]	12.752	SLOW	3.744	FAST
Y[2]	S[3]	12.404	SLOW	3.662	FAST
Y[3]	C_o	12.403	SLOW	3.629	FAST

Figure 41: Combinational delay of Add_Sub module

1.6.1 Comments For Results

We see that our Adder subtractor circuit uses 24 luts after implementation. Even though we added the don't touch constraint, one of the 25 lots I expected was intertwined in the optimization. As for Max delay, I saw that there was a delay of 21.75 ns from Y_0 to V. The highest end-to-end delay was a result I expected.

Circuit	Utilization	Max Delay
RCA	20	18.49
RCA (parameter based)	20	18.924
CLA	18	16.082
Add_Sub	24	21.275

Table 2: Table of Utilization And Timing Results

As a result, we can see that as the number of luts increases in 4-bit addition and subtraction circuits, the $\max_{d}elayalsoincreases$.