Digital Logic Design & Computer Architecture Lab report

First assignment

PREPARED FOR

Miss. Talebpour

PREPARED BY

Mohsen Karbalaei Amini, 98242128

Mohammad Mehdi Parchami, 400243084

March 4, 2023,

Ripple-Carry

Implementation (Part a)

We constructed a full adder making use of *data flow* coding style and used that module to build the 12-bit ripple carry (structural coding style).

Here you can see the codes:

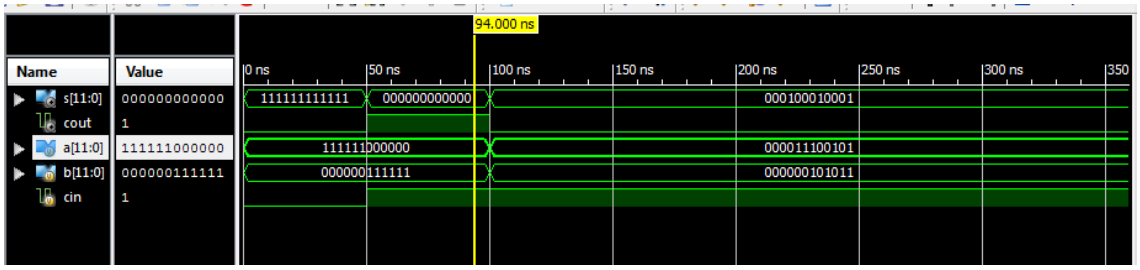
```
20  //////////////////////////////////////////////////////////////////////////////////
21  module full_adder(
22      input a,
23      input b,
24      input cin,
25      output s,
26      output cout
27      );
28        assign s = a ^ b ^ cin ;
29        assign cout = ( a & b ) | ( a & cin ) | ( b & cin );
30
31
32  endmodule
33
```

```
21  module ripple_carry12(
22      input [11:0] a,
23      input [11:0] b,
24      input cin,
25      output [11:0] s,
26      output cout
27
28      );
29
30     wire w0,w1,w2,w3,w4,w5,w6,w7,w8,w9,w10;
31     full_adder FA0 (a[0], b[0], cin, s[0], w0);
32     full_adder FA1 (a[1], b[1], w0, s[1], w1);
33     full_adder FA2 (a[2], b[2], w1, s[2], w2);
34     full_adder FA3 (a[3], b[3], w2, s[3], w3);
35     full_adder FA4 (a[4], b[4], w3, s[4], w4);
36     full_adder FA5 (a[5], b[5], w4, s[5], w5);
37     full_adder FA6 (a[6], b[6], w5, s[6], w6);
38     full_adder FA7 (a[7], b[7], w6, s[7], w7);
39     full_adder FA8 (a[8], b[8], w7, s[8], w8);
40     full_adder FA9 (a[9], b[9], w8, s[9], w9);
41     full_adder FA10 (a[10], b[10], w9, s[10], w10);
42     full_adder FA11 (a[11], b[11], w10, s[11], cout);
43
44  endmodule
45
```

Test Bench (part b)

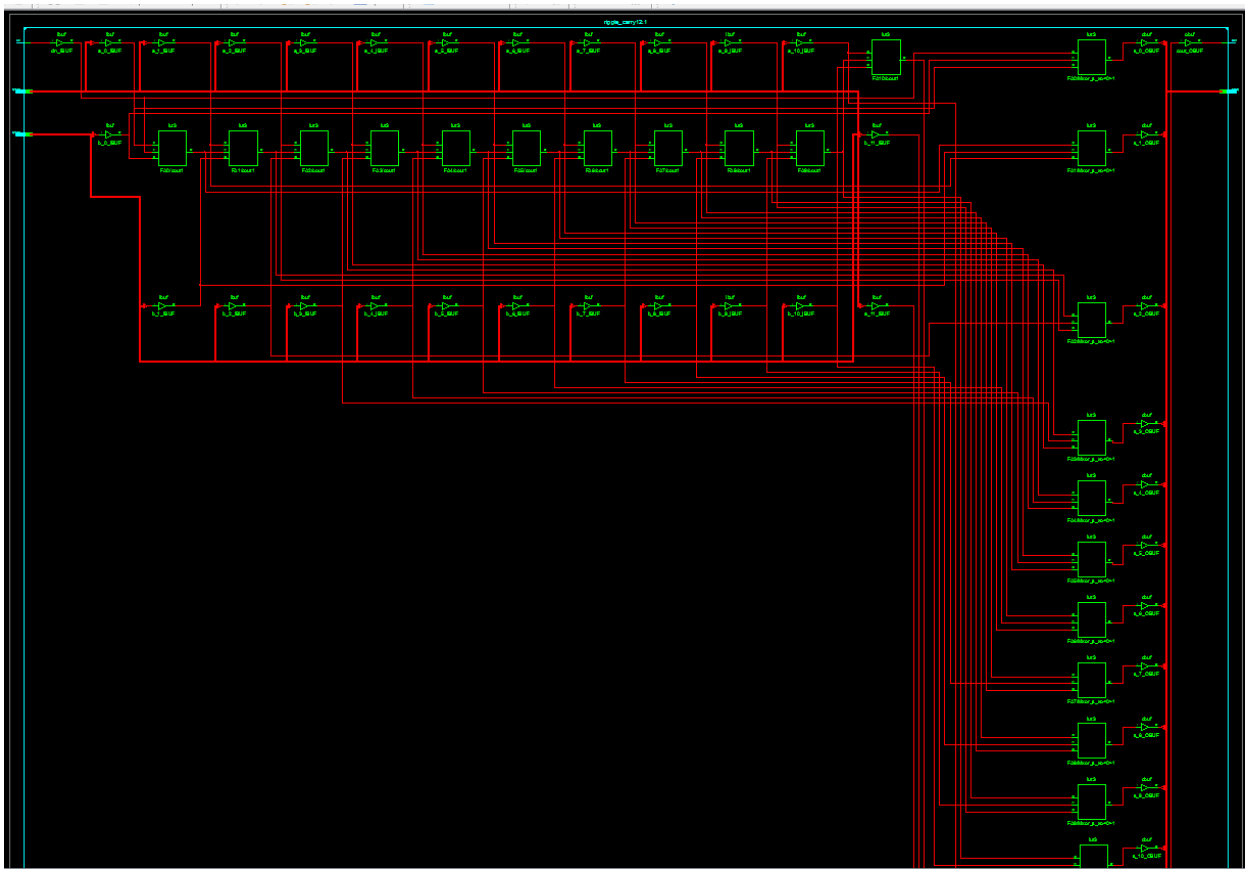
Code and simulation:

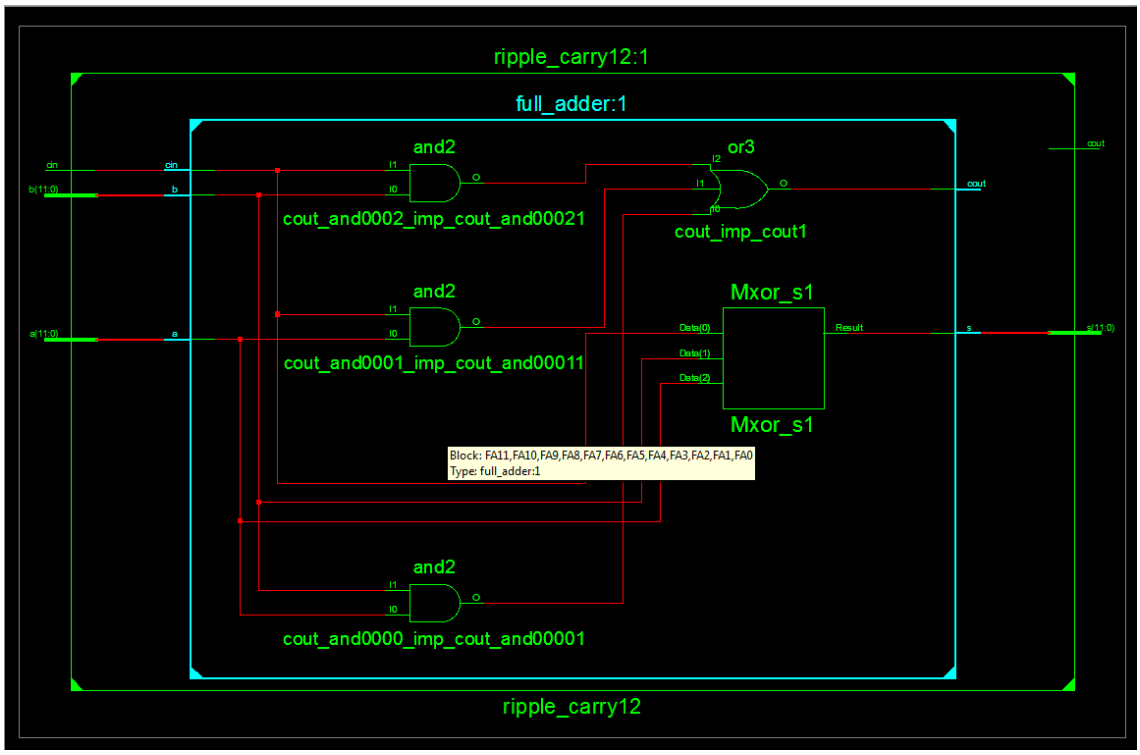
```
25  module ripple_carry12_tb;
26
27     // Inputs
28     reg [11:0] a;
29     reg [11:0] b;
30     reg cin;
31
32     // Outputs
33     wire [11:0] s;
34     wire cout;
35
36     // Instantiate the Unit Under Test (UUT)
37     ripple_carry12 uut (
38        .a(a),
39        .b(b),
40        .cin(cin),
41        .s(s),
42        .cout(cout)
43     );
44
45     initial begin
46        // Initialize Inputs
47        a = 12'b111111000000;
48        b = 12'b000000111111;
49        cin = 0;
50
51        #50;
52
53        a = 12'b111111000000;
54        b = 12'b000000111111;
55        cin = 1;
56
57        #50
58
59
60        a = 12'b000011100101;
61        b = 12'b000000101011;
62        cin = 1;
63        // Add stimulus here
64
65     end
```



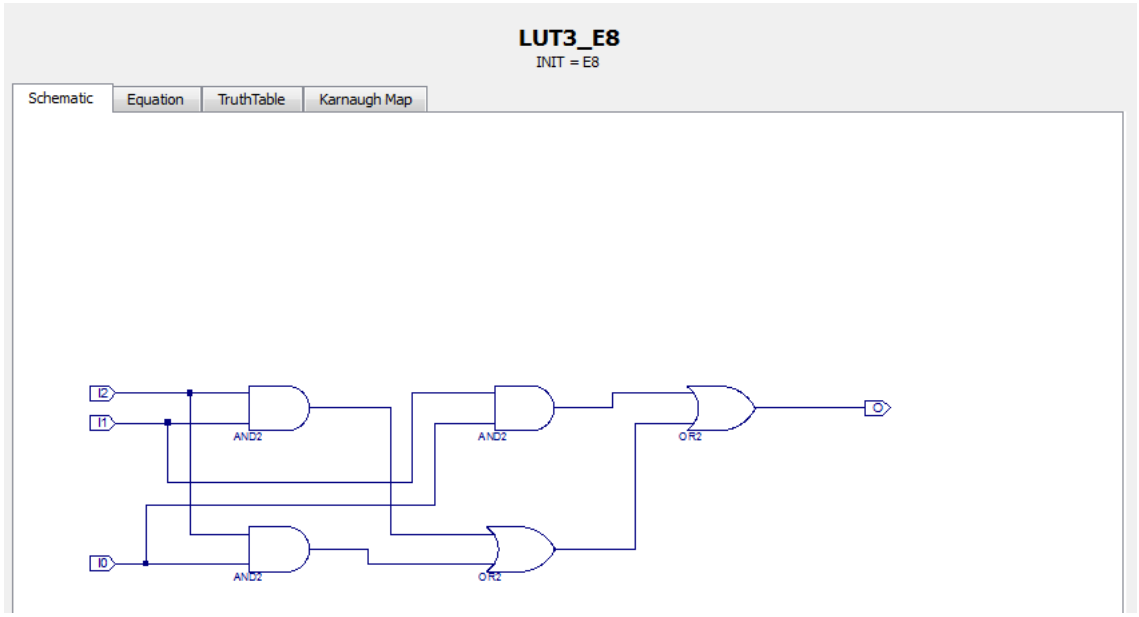
Synthesize (part c)

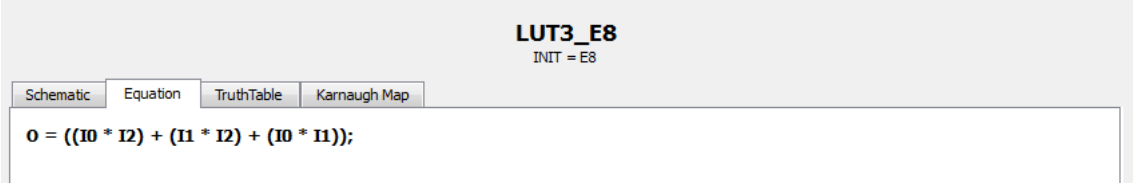
Here’s the technology and the RTL schematic view.

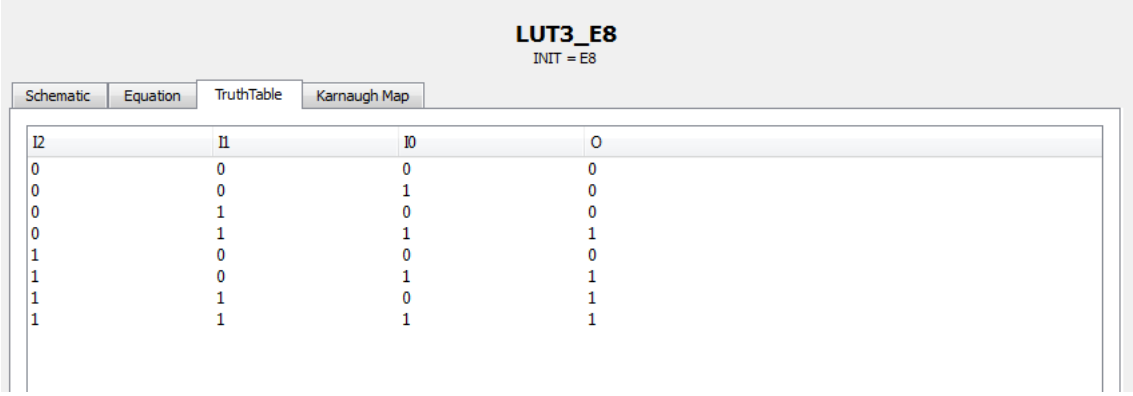




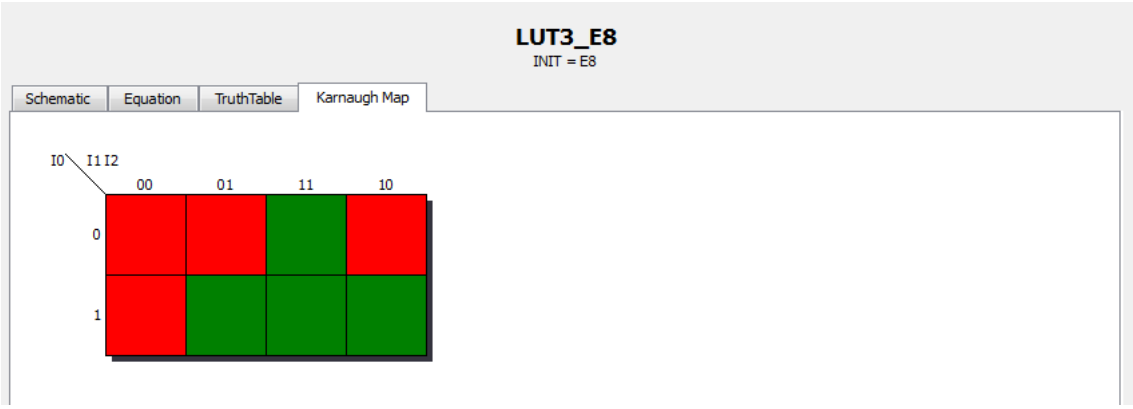
LUT Details (part d)







I2	I1	I0	O
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



Critical Path Delay Analysis (part e)

Based on the post-PAR static timing report, the critical path in this Circuit is the way from ***cin to s[11]***, with a delay of 17.145 ns.

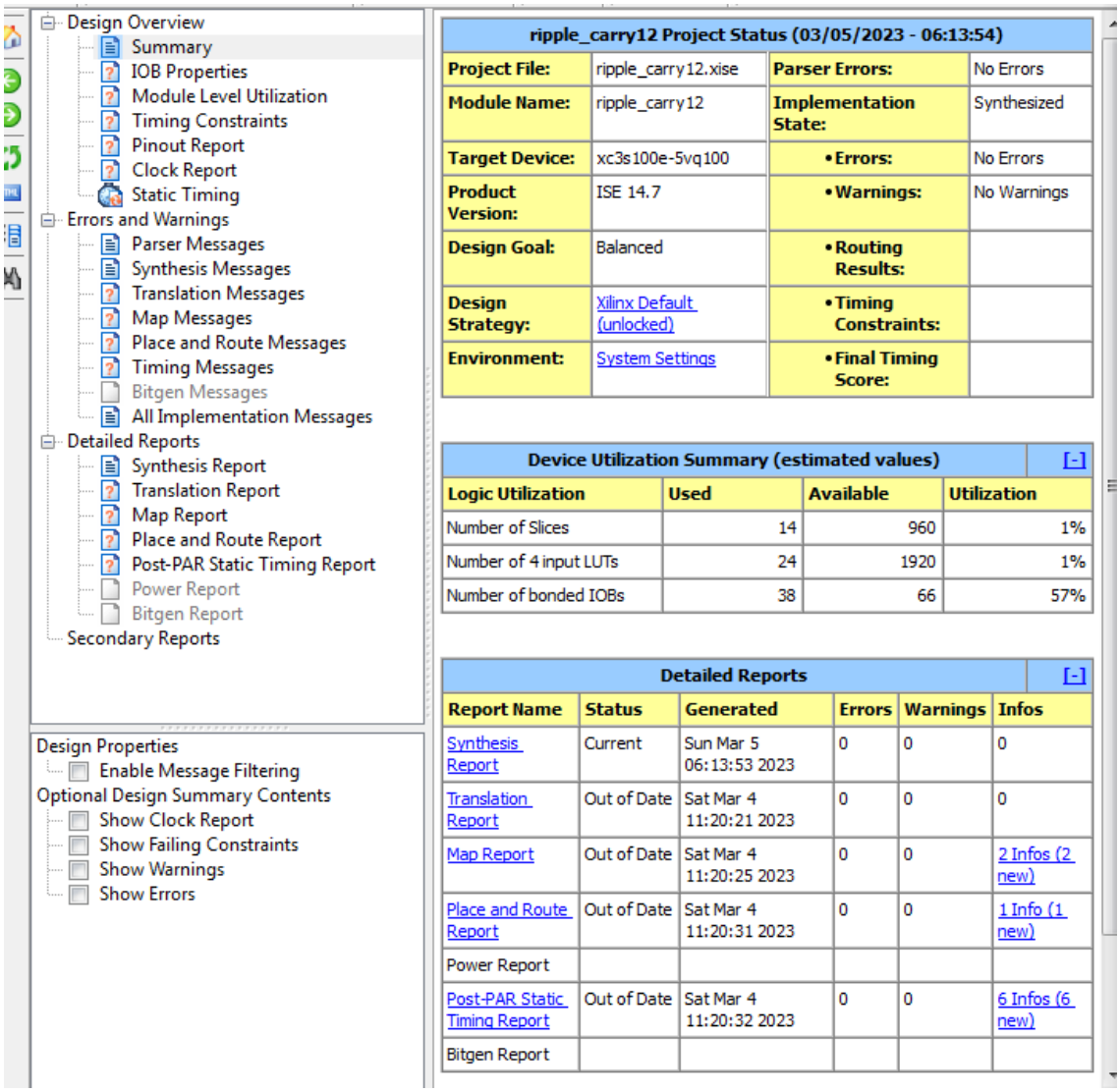
Here's a bit summary of the most delayed paths.

Source Pad	Destination Pad	Delay
a<0>	cout	16.443
a<0>	s<10>	15.731
a<0>	s<11>	16.579
a<1>	cout	15.924

...

```
b<0>        |s<8>        |  14.122|
b<0>        |s<9>        |  14.700|
b<0>        |s<10>       |  15.904|
b<0>        |s<11>       |  16.752|
b<1>        |cout        |  15.964|
...
cin         |s<9>        | 15.093|
cin         |s<10>       |  16.297|
cin         |s<11>       |  17.145|
```

Design Summary (part f)



ripple_carry12 Project Status (03/05/2023 - 06:13:54)			
Project File:	ripple_carry12.xise	**Parser Errors:**	No Errors
Module Name:	ripple_carry12	**Implementation State:**	Synthesized
Target Device:	xc3s100e-5vq100	• **Errors:**	No Errors
Product Version:	ISE 14.7	• **Warnings:**	No Warnings
Design Goal:	Balanced	• **Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	• **Timing Constraints:**	
Environment:	System Settings	• **Final Timing Score:**	

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	14	960	1%
Number of 4 input LUTs	24	1920	1%
Number of bonded IOBs	38	66	57%

Detailed Reports					[-]
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	Sun Mar 5 06:13:53 2023	0	0	0
Translation Report	Out of Date	Sat Mar 4 11:20:21 2023	0	0	0
Map Report	Out of Date	Sat Mar 4 11:20:25 2023	0	0	2 Infos (2 new)
Place and Route Report	Out of Date	Sat Mar 4 11:20:31 2023	0	0	1 Info (1 new)
Power Report					
Post-PAR Static Timing Report	Out of Date	Sat Mar 4 11:20:32 2023	0	0	6 Infos (6 new)
Bitgen Report					

Carry-Look-Ahead

Implementation (Part a)

We define two variables as **'carry generate'** G_i and **'carry propagate'** P_i then,

$$P_i = A_i \oplus B_i$$
$$G_i = A_i B_i$$

The sum output and carry output can be expressed in terms of carry generate G_i and carry propagate P_i as

$$S_i = P_i \oplus C_i$$
$$C_{i+1} = G_i + P_i C_i$$

where G_i produces the carry when both A_i, B_i are 1 regardless of the input carry. P_i is associated with the propagation of carry from C_i to C_{i+1}.

So we write this code to implement carry-look-ahead adder:

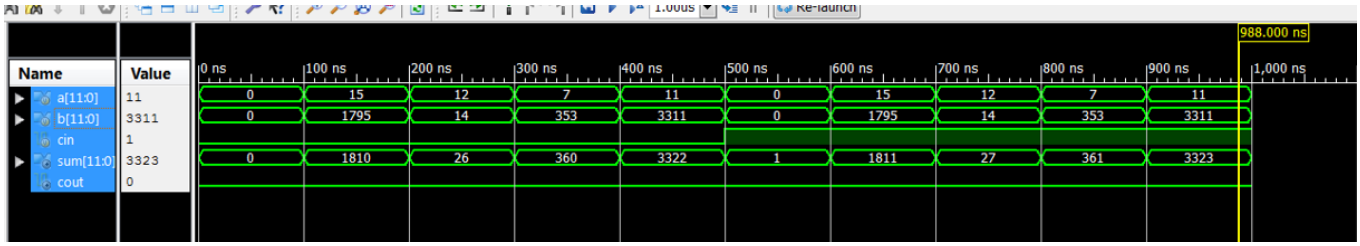
```
module cla12 (a, b, cin, sum, cout);
    input [11:0] a, b;
    input cin;
    output reg [11:0] sum;
    output reg cout;
    integer i, j, k;
    reg [11:0] g,p;
    reg [12:0] c;
    reg temp;
    always @(*) begin
        c[0] = cin;
        temp = 1;
        for(i = 0; i < 12; i = i + 1) begin
            g[i] = a[i] & b[i];
            p[i] = a[i] ^ b[i];
            c[i + 1] = g[i] | (p[i] & c[i]);
            sum[i] = p[i] ^ c[i];
        end
        cout = c[12];
    end
endmodule
```

Test Bench and Results (part b)

Code and simulation:

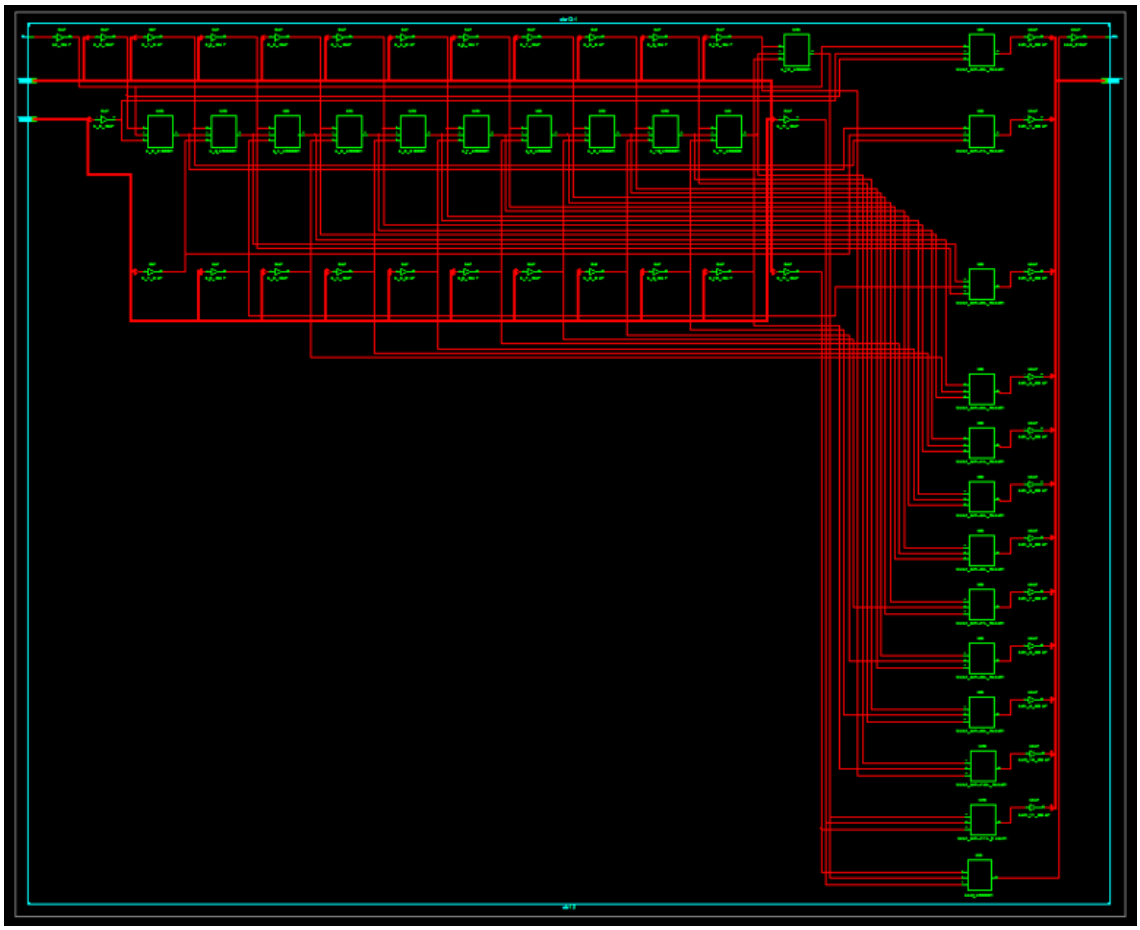
```
`timescale 1ns / 1ps
module cla12_tb;
        reg [11:0] a, b;
        reg cin;
        wire [11:0] sum;
        wire cout;
        cla12 uut (.a(a), .b(b), .cin(cin), .sum(sum), .cout(cout));
        initial begin
        cin = 0;
        a = 12'd0;   b = 12'd0;    #100;
        a = 12'd15;  b = 12'd1795; #100;
        a = 12'd12;  b = 12'd14;   #100;
        a = 12'd7;   b = 12'd353;  #100;
        a = 12'd11;  b = 12'd3311; #100;
        cin = 1;
        a = 12'd0;   b = 12'd0;    #100;
        a = 12'd15;  b = 12'd1795; #100;
        a = 12'd12;  b = 12'd14;   #100;
        a = 12'd7;   b = 12'd353;  #100;
        a = 12'd11;  b = 12'd3311; #100;
        end

endmodule
```

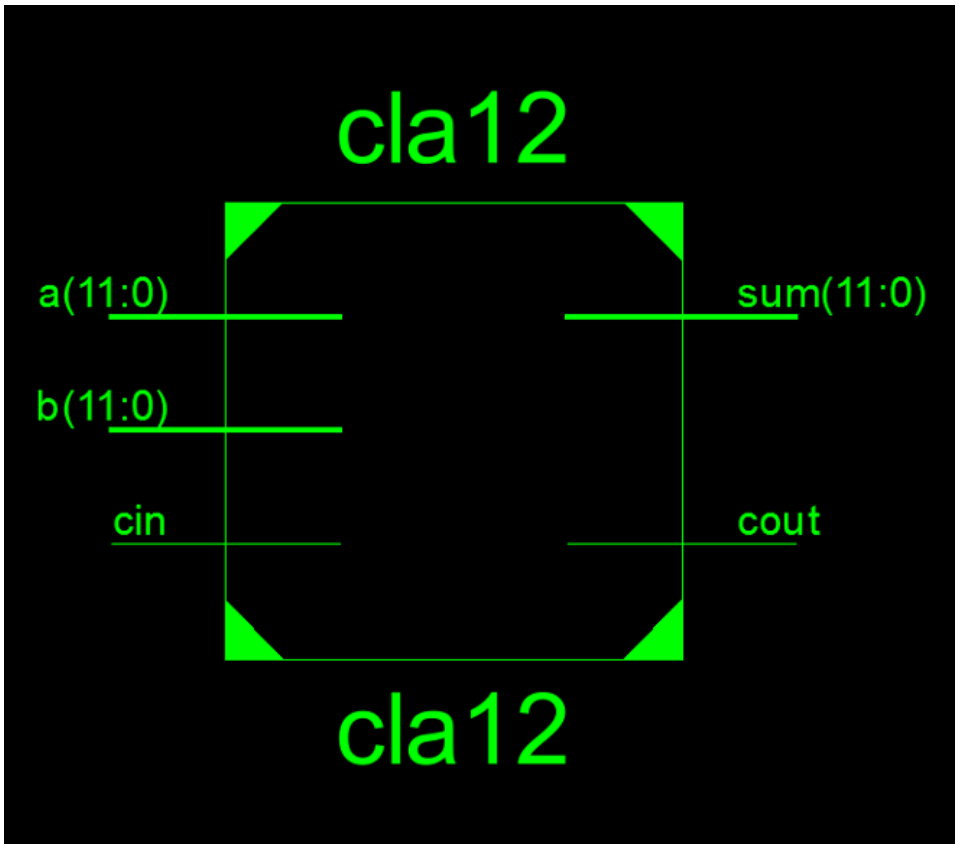
Results:

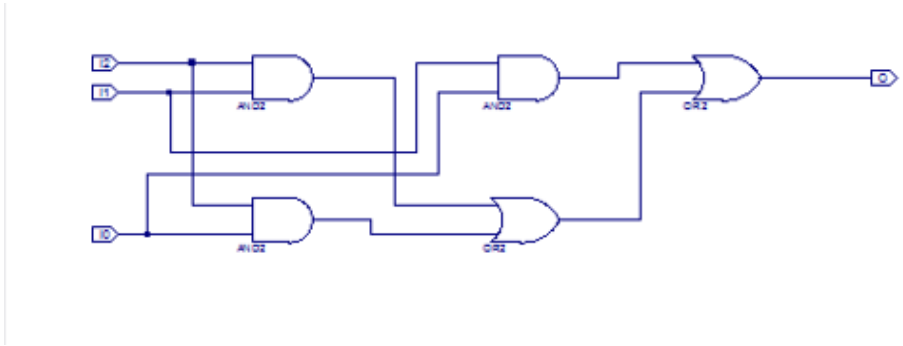


Synthesize (part c)

Here’s the technology and the RTL schematic view.







Results show the fact that after synthesizing, our implementation was completely **changed**! The new structure is very similar to rca structure.

Critical Path Delay Analysis (part e)

Based on the post-PAR static timing report, the critical path in this cla implementation is the way from ***b[0] to s[11]***, with a delay of 17.040 ns.

```
--------------+---------------+---------+
Source Pad    |Destination Pad|  Delay  |
--------------+---------------+---------+
a<0>          |cout           |  16.123|
b<0>          |cout           |  16.854|
b<0>          |sum<11>        |  17.040|
cin           |cout           |  16.629|
cin           |sum<11>        |  16.815|
--------------+---------------+---------+
```

Design Summary (part f)

cla12 Project Status (03/08/2023 - 11:26:30)			
Project File:	cla12.xise	**Parser Errors:**	No Errors
Module Name:	cla12	**Implementation State:**	Placed and Routed
Target Device:	xc3s100e-5vq100	• **Errors:**	No Errors
Product Version:	ISE 14.7	• **Warnings:**	6 Warnings (0 new)
Design Goal:	Balanced	• **Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	• **Timing Constraints:**	
Environment:	System Settings	• **Final Timing Score:**	0 (Timing Report)

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of 4 input LUTs	24	1,920	1%	
Number of occupied Slices	18	960	1%	
Number of Slices containing only related logic	18	18	100%	
Number of Slices containing unrelated logic	0	18	0%	
Total Number of 4 input LUTs	24	1,920	1%	
Number of bonded IOBs	38	66	57%	
Average Fanout of Non-Clock Nets	1.73			

Performance Summary				[-]
Final Timing Score:	0 (Setup: 0, Hold: 0)	**Pinout Data:**	Pinout Report	
Routing Results:	All Signals Completely Routed	**Clock Data:**	Clock Report	
Timing Constraints:				

Performance Comparison

Delay

Comparing the critical path delay of the two adder circuits, indicates that the carry look ahead is faster than ripple carry. Although we were expecting more difference, if we want to have results much clearer, we should have them scaled much more than only 12 bits.

	Ripple Carry	Carry Look Ahead
Critical Delay	17.145 ns	17.040 ns

Resource Utilization

As illustrated above, in case of *number of 4 input LUTs, occupied slices and IOBs*, both circuits have the same resource usage.