CS2610: Computer Organization and Architecture Lab Report 3

Devansh Singh(111701011), Himanshu Jain(111701013)

Objective:

To analyse the efficiency of array multiplier and Multiplier using Booth's algorithm in terms of processing delay and power consumption.

Problem:

To implement the below mentioned scenarios in Verilog:

a.) Implement an array multiplier that would multiply two unsigned numbers A and B.
b.) Implement Booth's algorithm that would multiply two 2's complement numbers A and B.

Theory and Approach:

Half Adder: *A half adder is a type of adder, an electronic circuit that performs the addition of numbers. The half adder is able to add two single binary digits and provide the output plus a carry value. It has two inputs, called A and B, and two outputs S (sum) and C (carry).*

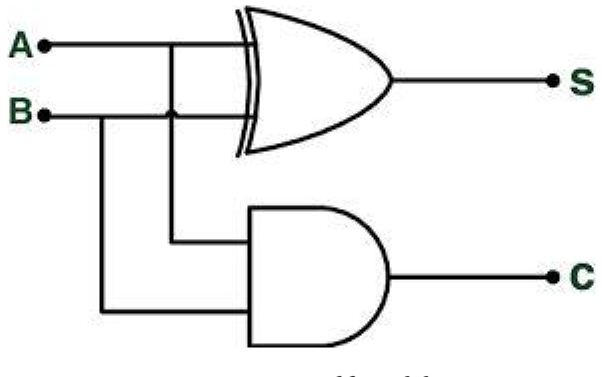


Figure: Half Adder

Code for half adder

```
module half_adder(input a,b, output c,s);

 assign s = a^b;
 assign c = a&b;

endmodule
```

Full Adder: *A full adder is a logical circuit that performs an addition operation on three one-bit binary numbers. The full adder produces a sum of the two inputs and carry value. It gives a two bit output. First bit as sum and the second bit is the carry bit. Both these represents the sum of the given three one bit inputs.*

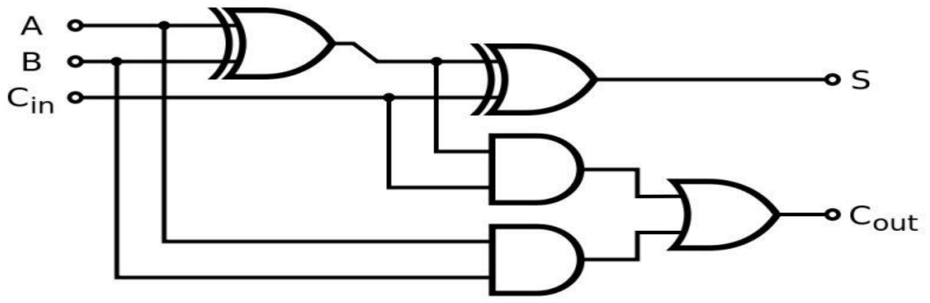


Figure 1: Full Adder Circuit

Code for full adder

```
module  full_adder(input a, input b, input c, output s, output C);

        assign  s =  a^b^c;
        assign  C = ((a^b)&c) | (a&b);

endmodule
```

1.) Array multiplier of two unsigned numbers

Array multiplier is well known due to its regular structure. Multiplier circuit is based on add and shift algorithm. Each partial product is generated by the multiplication of the multiplicand with one multiplier bit. The partial product are shifted according to their bit orders and then added.
We have used full adders and half adders to implement the array multiplier in the way as given in the below image.

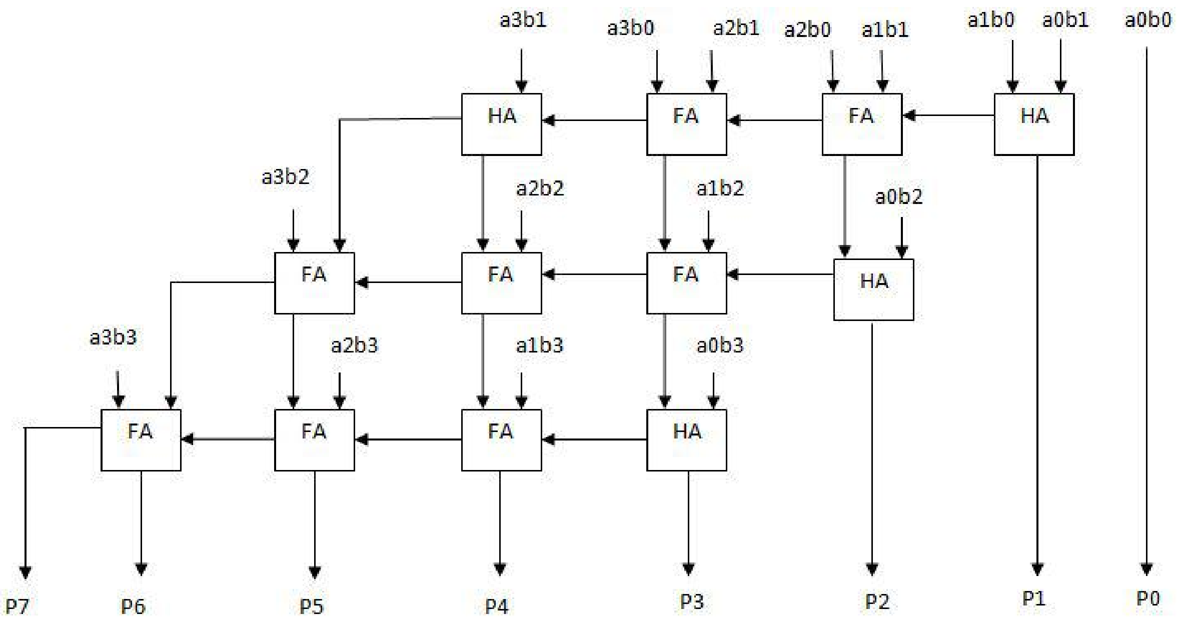


Figure 2: 4 Array Multiplier

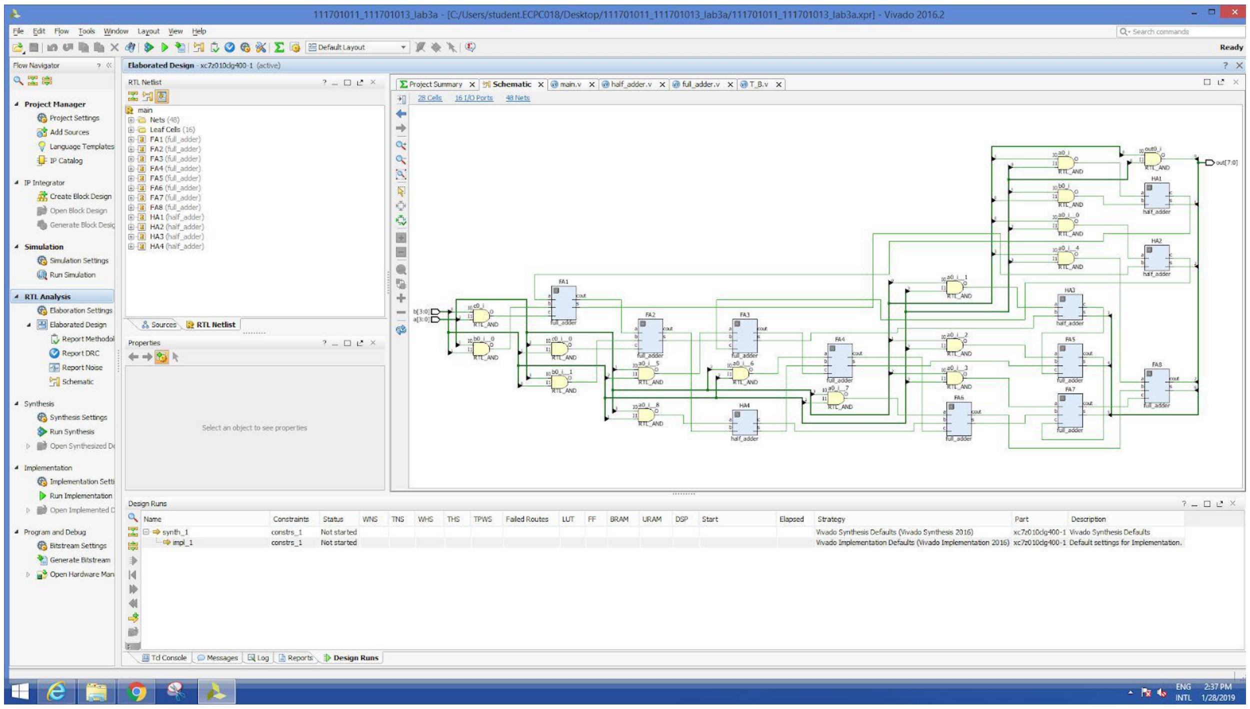
Code for Array Multiplier

```
module main(input [3:0]a,b,
        output [7:0] out);

        wire c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11;
        wire s1,s2,s3,s4,s5,s6;
        assign out[0] = a[0]&b[0];
        half_adder HA1 (a[1]&b[0],b[1]&a[0],c1,out[1]);
        full_adder FA1 (c1,a[1]&b[1],b[0]&a[2],c2,s1);
        half_adder HA2 (a[0]&b[2],s1,c3,out[2]);
        full_adder FA2 (c2,a[3]&b[0],a[2]&b[1],c4,s2);
        full_adder FA3 (a[1]&b[2],s2,c3,c5,s3);
        half_adder HA3 (a[0]&b[3],s3,c6,out[3]);
        half_adder HA4 (a[3]&b[1],c4,c7,s4);
        full_adder FA4 (a[2]&b[2],s4,c5,c8,s5);
        full_adder FA5 (a[1]&b[3],s5,c6,c9,out[4]);
        full_adder FA6 (a[3]&b[2],c7,c8,c10,s6);
        full_adder FA7 (a[2]&b[3],s6,c9,c11,out[5]);
        full_adder FA8 (a[3]&b[3],c10,c11,out[7],out[6]);


endmodule
```

Elaborated Design



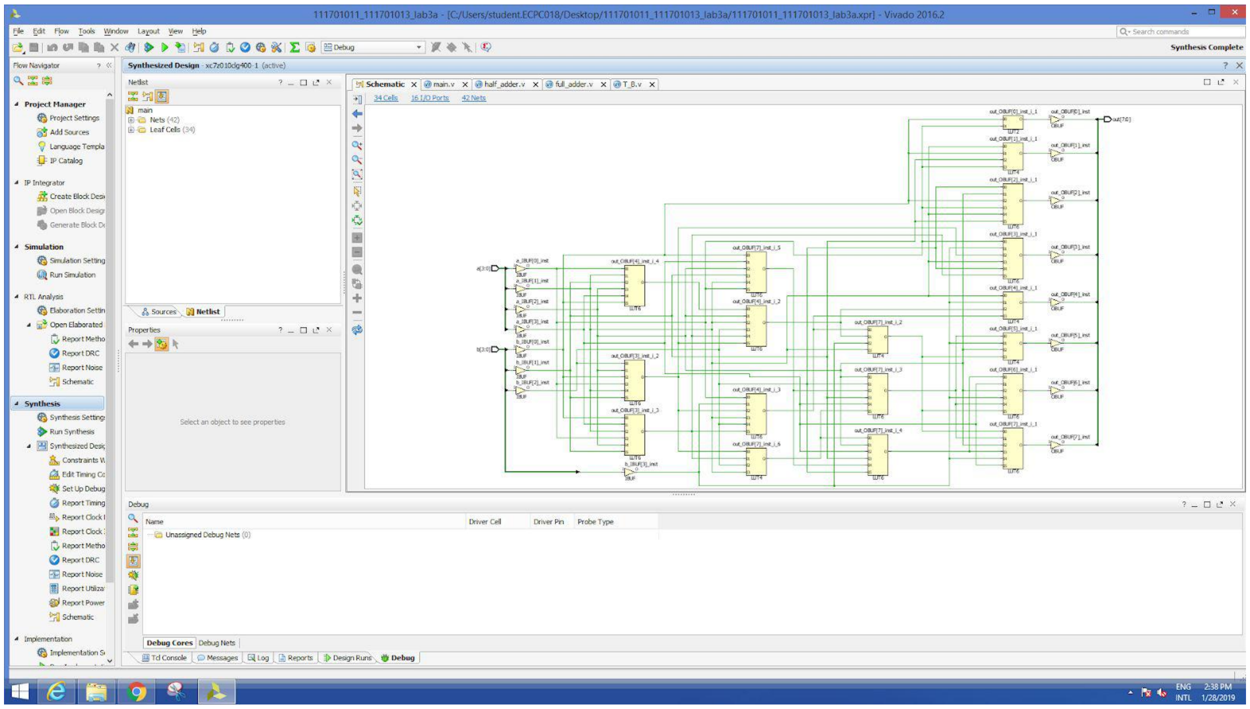
111701011_111701013_lab3a - [C:/Users/student.ECPC018/Desktop/111701011_111701013_lab3a/111701011_111701013_lab3a.xpr] - Vivado 2016.2

Elaborated Design - xc7z010clg400-1 (active)

28 Cells 16 I/O Ports 48 Nets

Design Runs

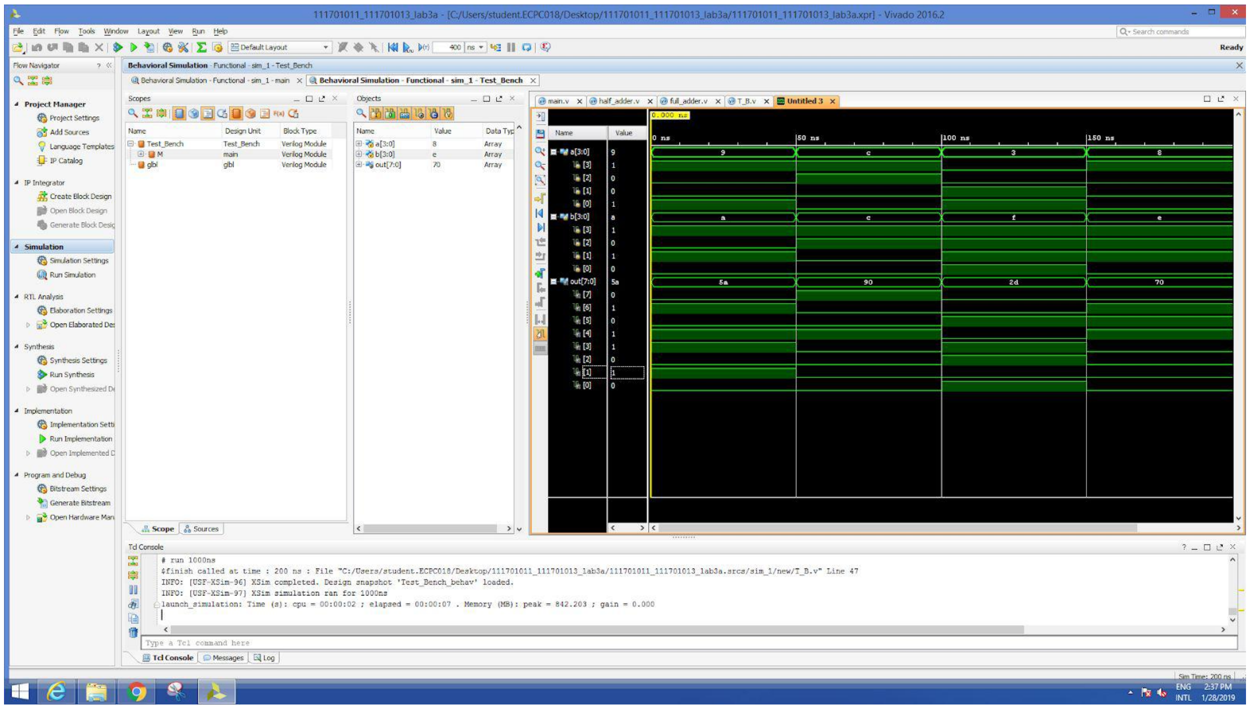
Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Failed Routes	LUT	FF	BRAM	URAM	DSP	Start	Elapsed	Strategy	Part	Description
synth_1	constrs_1	Not started														Vivado Synthesis Defaults (Vivado Synthesis 2016)	xc7z010clg400-1	Vivado Synthesis Defaults
impl_1	constrs_1	Not started														Vivado Implementation Defaults (Vivado Implementation 2016)	xc7z010clg400-1	Default settings for Implementation.



111701011_111701013_lab3a - [C:/Users/student.ECPC018/Desktop/111701011_111701013_lab3a/111701011_111701013_lab3a.xpr] - Vivado 2016.2

Synthesized Design - xc7z010clg400-1 (active)

Simulation Results



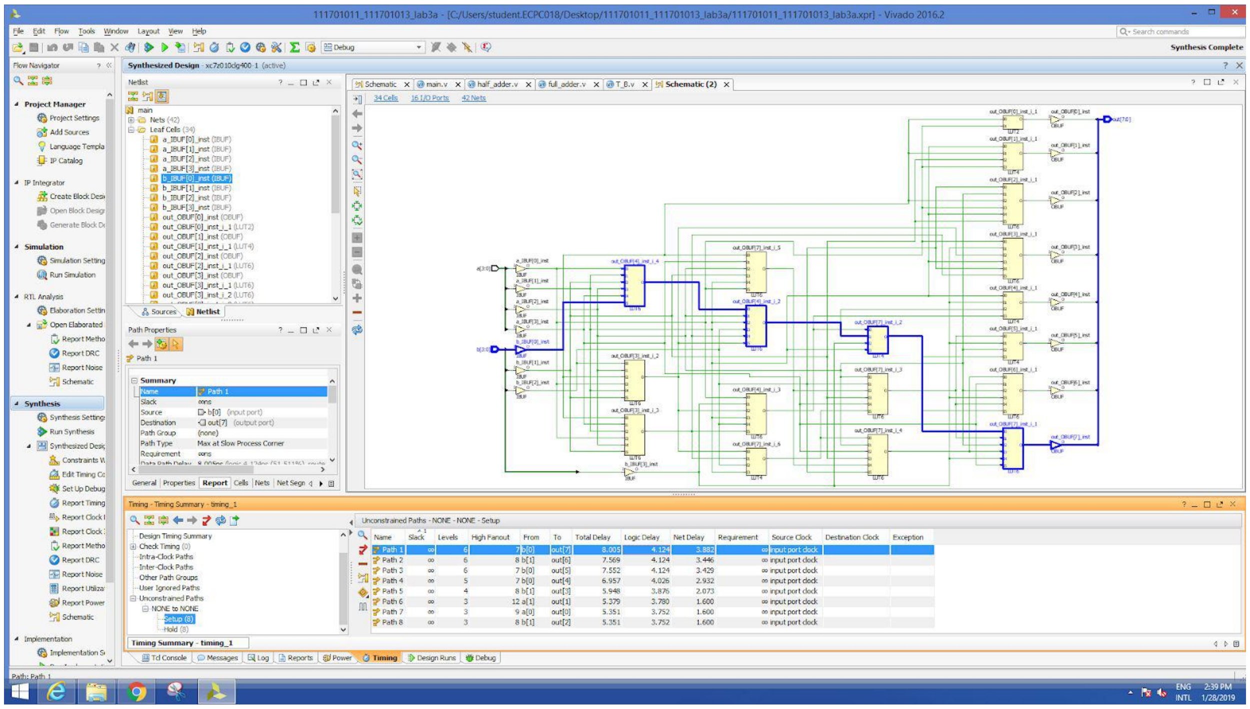
111701011_111701013_lab3a - [C:/Users/student.ECPC018/Desktop/111701011_111701013_lab3a/111701011_111701013_lab3a.xpr] - Vivado 2016.2

Behavioral Simulation - Functional - sim_1 - Test_Bench

Tcl Console

```
# run 1000ns
$finish called at time : 200 ns : File "C:/Users/student.ECPC018/Desktop/111701011_111701013_lab3a/111701011_111701013_lab3a.srcs/sim_1/new/T_B.v" Line 47
INFO: [USF-XSim-96] XSim completed. Design snapshot 'Test_Bench_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:02 ; elapsed = 00:00:07 . Memory (MB): peak = 842.203 ; gain = 0.000
```

Timing Results



111701011_111701013_lab3a - [C:/Users/student.ECPC018/Desktop/111701011_111701013_lab3a/111701011_111701013_lab3a.xpr] - Vivado 2016.2

Synthesized Design - xc7z010clg400-1 (active)

Path Properties – Path 1

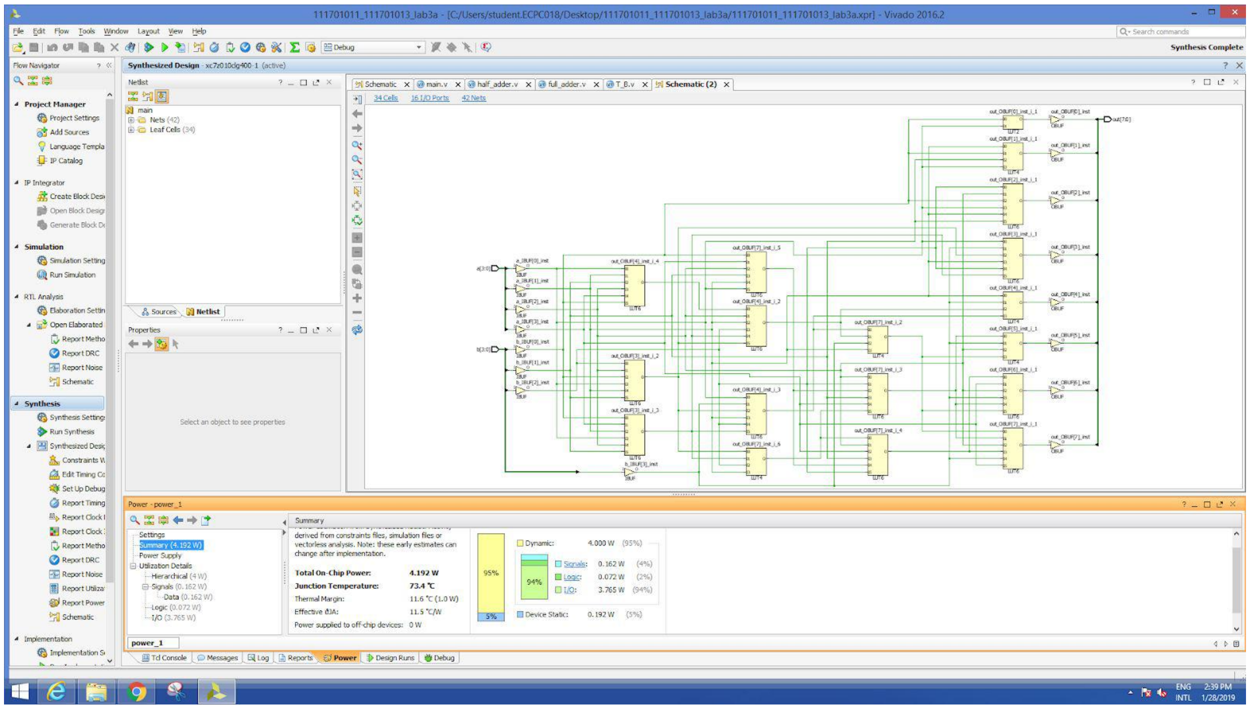
Summary	
Name	Path 1
Slack	∞ns
Source	b[0] (input port)
Destination	out[7] (output port)
Path Group	(none)
Path Type	Max at Slow Process Corner
Requirement	∞ns

Timing - Timing Summary - timing_1

Unconstrained Paths - NONE - NONE - Setup

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception
Path 1	∞	6	7	b[0]	out[7]	8.005	4.124	3.882	∞	input port clock		
Path 2	∞	6	8	b[1]	out[6]	7.569	4.124	3.446	∞	input port clock		
Path 3	∞	6	7	b[0]	out[5]	7.552	4.124	3.429	∞	input port clock		
Path 4	∞	5	7	b[0]	out[4]	6.957	4.026	2.932	∞	input port clock		
Path 5	∞	4	8	b[1]	out[3]	5.948	3.876	2.073	∞	input port clock		
Path 6	∞	3	12	a[1]	out[1]	5.379	3.780	1.600	∞	input port clock		
Path 7	∞	3	9	a[0]	out[0]	5.351	3.752	1.600	∞	input port clock		
Path 8	∞	3	8	b[1]	out[2]	5.351	3.752	1.600	∞	input port clock		

Schematic and On chip power Results



111701011_111701013_lab3a - [C:/Users/student.ECPC018/Desktop/111701011_111701013_lab3a/111701011_111701013_lab3a.xpr] - Vivado 2016.2

Power - power_1

Summary

derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

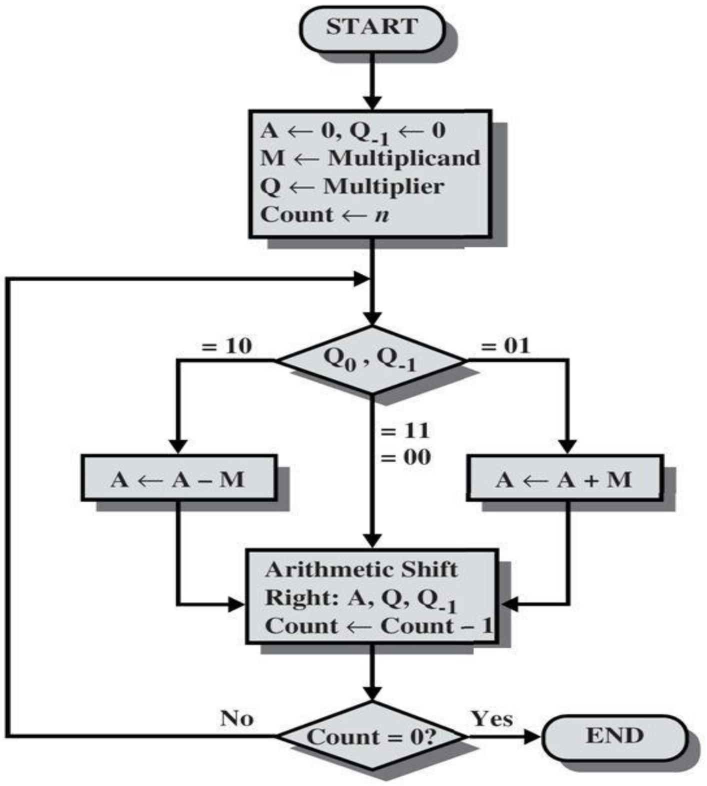
Total On-Chip Power:	**4.192 W**
Junction Temperature:	**73.4 ℃**
Thermal Margin:	11.6 ℃ (1.0 W)
Effective θJA:	11.5 ℃/W
Power supplied to off-chip devices:	0 W

Dynamic:	4.000 W	(95%)
Signals:	0.162 W	(4%)
Logic:	0.072 W	(2%)
I/O:	3.765 W	(94%)
Device Static:	0.192 W	(5%)

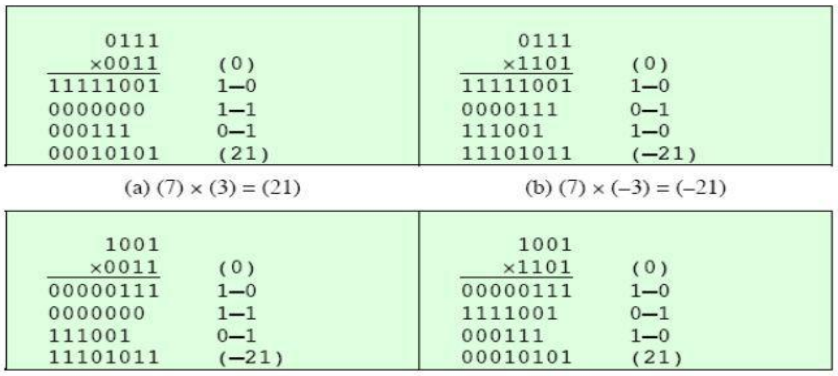
2.) Multiplier using Booth's Algorithm

Booth's multiplication algorithm is a multiplication algorithm that multiplies two signed binary numbers in two's complement notation.
Booth's algorithm examines adjacent pairs of bits of the 'N'-bit multiplier Y in signed two's complement representation, including an implicit bit below the least significant bit, $y_{-1} = 0$. For each bit y_i, for i running from 0 to $N - 1$, the bits y_i and y_{i-1} are considered. Where these two bits are equal, the product accumulator P is left unchanged. Where $y_i = 0$ and $y_{i-1} = 1$, the multiplicand times 2^i is added to P; and where $y_i = 1$ and $y_{i-1} = 0$, the multiplicand times 2^i is subtracted from P. The final value of P is the signed product.
The representations of the multiplicand and product are both also in two's complement representation, like the multiplier. As stated here, the order of the steps is not determined. Typically, it proceeds from LSB to MSB, starting at $i = 0$; the multiplication by 2^i is then typically replaced by incremental shifting of the P accumulator to the right between steps; low bits can be shifted out, and subsequent additions and subtractions can then be done just on the highest N bits of P.

Booth's Algorithm Flowchart



2's Comp Multiplication
Booth's Algorithm



```
    0111
   ×0011       (0)
11111001       1–0
0000000        1–1
000111         0–1
00010101       (21)
```

(a) (7) × (3) = (21)

```
    0111
   ×1101       (0)
11111001       1–0
0000111        0–1
111001         1–0
11101011       (–21)
```

(b) (7) × (–3) = (–21)

```
    1001
   ×0011       (0)
00000111       1–0
0000000        1–1
111001         0–1
11101011       (–21)
```

(c) (–7) × (3) = (–21)

```
    1001
   ×1101       (0)
00000111       1–0
1111001        0–1
000111         1–0
00010101       (21)
```

(d) (–7) × (–3) = (21)

Figure 9.14 Examples Using Booth's Algorithm

Code for Booth's Multiplier

```
module main( input [3:0] M,input [3:0] Q,output [7:0] O);

   wire [7:0] A, B;
   reg [7:0] temp, A1, X, Y, P, C1;

   adder a1(X, Y, A);
   adder a2(A, P, B);
   adder a3(B, temp, O);

   always@*
   begin
      if(M[3] == 0)
      begin
         A1 = 8'b00000000;
      end
      else
      begin
         A1 = 8'b11110000;
      end
      if(Q[0] == 1)
```

```
begin
   A1[0] = M[0];
   A1[1] = M[1];
   A1[2] = M[2];
   A1[3] = M[3];

   C1[0] = ~A1[0];
   C1[1] = ~A1[1];
   C1[2] = ~A1[2];
   C1[3] = ~A1[3];
   C1[4] = ~A1[4];
   C1[5] = ~A1[5];
   C1[6] = ~A1[6];
   C1[7] = ~A1[7];
   C1 = C1 + 1;
   X = C1;
end
else
begin
X = 8'b00000000;
end
if(Q[0] == 1 && Q[1] == 0)
begin
    A1[1] = M[0];
    A1[2] = M[1];
    A1[3] = M[2];
    A1[4] = M[3];
    Y = A1;
end
else if(Q[0] == 0 && Q[1] == 1)
begin
    A1[1] = M[0];
    A1[2] = M[1];
    A1[3] = M[2];
    A1[4] = M[3];
    C1[0] = ~A1[0];
    C1[1] = ~A1[1];
    C1[2] = ~A1[2];
    C1[3] = ~A1[3];
    C1[4] = ~A1[4];
    C1[5] = ~A1[5];
    C1[6] = ~A1[6];
    C1[7] = ~A1[7];
    C1 = C1 + 1;

    Y = C1;
end
else
begin
Y = 8'b00000000;
end

if(Q[1] == 1 && Q[2] == 0)
begin
    A1[0] = 1'b0;
    A1[1] = 1'b0;
    A1[2] = M[0];
    A1[3] = M[1];
    A1[4] = M[2];
    A1[5] = M[3];
    P = A1;
end
else if(Q[1] == 0 && Q[2] == 1)
begin
    A1[0] = 1'b0;
    A1[1] = 1'b0;
    A1[2] = M[0];
    A1[3] = M[1];
    A1[4] = M[2];
    A1[5] = M[3];
    C1[0] = ~A1[0];
    C1[1] = ~A1[1];
    C1[2] = ~A1[2];
    C1[3] = ~A1[3];
    C1[4] = ~A1[4];
    C1[5] = ~A1[5];
    C1[6] = ~A1[6];
    C1[7] = ~A1[7];
    C1 = C1 + 1;

    P = C1;
end
else
begin
P = 8'b00000000;
end
```

```
        if(Q[2] == 1 && Q[3] == 0)
        begin
            A1[0] = 1'b0;
            A1[1] = 1'b0;
            A1[2] = 1'b0;
            A1[3] = M[0];
            A1[4] = M[1];
            A1[5] = M[2];
            A1[6] = M[3];
            temp = A1;
        end
        else if(Q[2] == 0 && Q[3] == 1)
        begin
            A1[0] = 1'b0;
            A1[1] = 1'b0;
            A1[2] = 1'b0;
            A1[3] = M[0];
            A1[4] = M[1];
            A1[5] = M[2];
            A1[6] = M[3];
            C1[0] = ~A1[0];
            C1[1] = ~A1[1];
            C1[2] = ~A1[2];
            C1[3] = ~A1[3];
            C1[4] = ~A1[4];
            C1[5] = ~A1[5];
            C1[6] = ~A1[6];
            C1[7] = ~A1[7];
            C1 = C1 + 1;

            temp = C1;
        end
        else
        begin
        temp = 8'b00000000;
        end
    end


Endmodule
```

Adder: *Adder is 8 bit adder of two 8-bit given numbers. It gives only Sum and carry out (last bit) is neglected.*

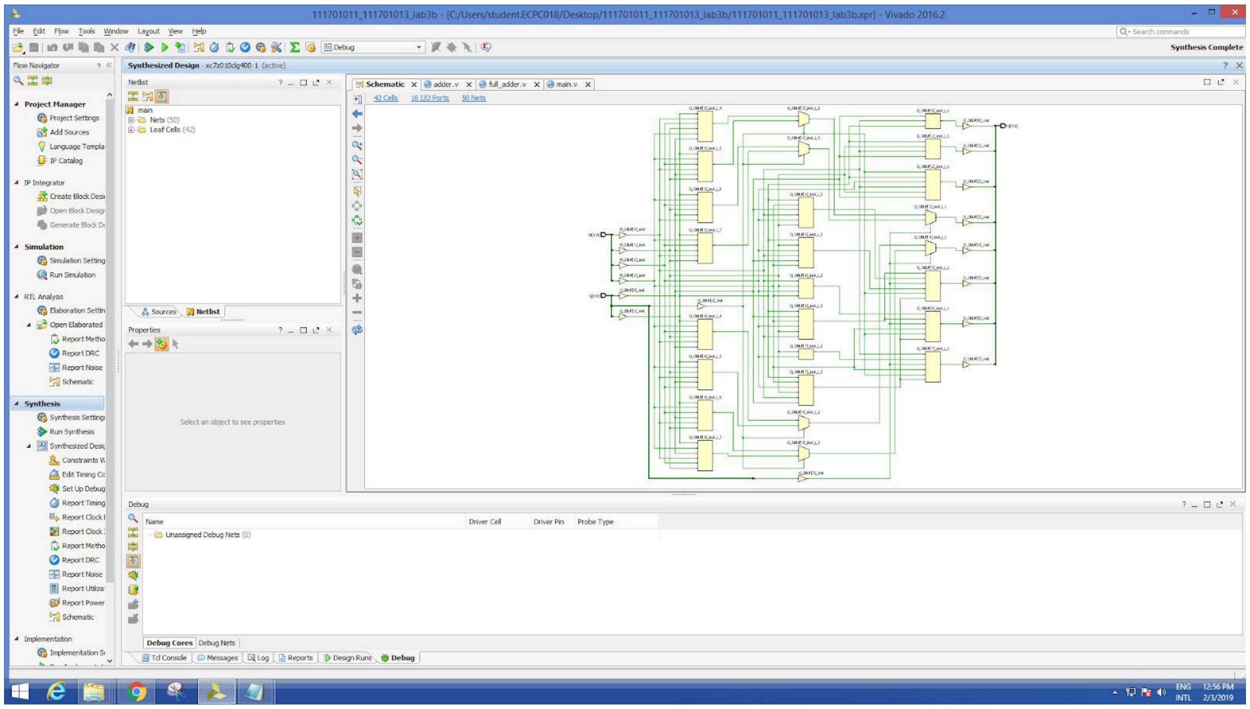
Code for adder

```
   module adder(
 input [7:0] A,
 input [7:0] B,
 output [7:0] S
 );

 wire c1, c2, c3, c4, c5, c6, c7, c8;

 fullAdder fa1(A[0], B[0], 0, S[0], c1);
 fullAdder fa2(A[1], B[1], c1, S[1], c2);
 fullAdder fa3(A[2], B[2], c2, S[2], c3);
 fullAdder fa4(A[3], B[3], c3, S[3], c4);
 fullAdder fa5(A[4], B[4], c4, S[4], c5);
 fullAdder fa6(A[5], B[5], c5, S[5], c6);
 fullAdder fa7(A[6], B[6], c6, S[6], c7);
 fullAdder fa8(A[7], B[7], c7, S[7], c8);
endmodule
```

Elaborated design



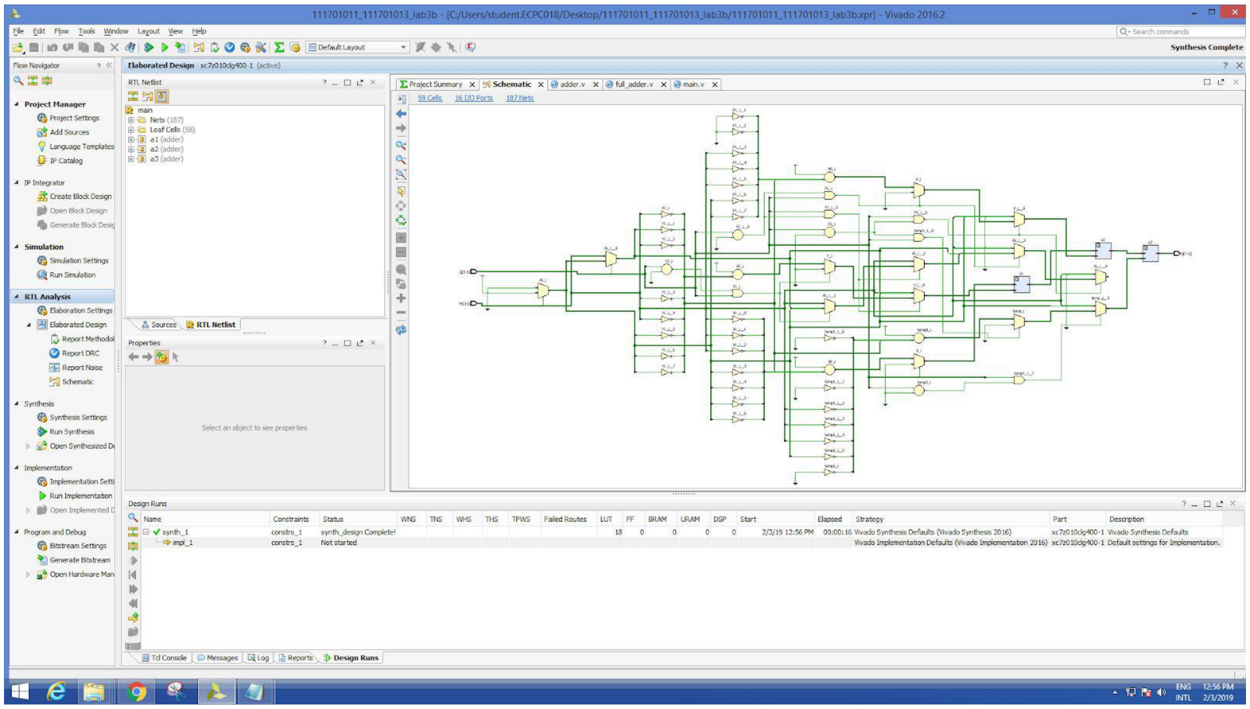
111701011_111701013_lab3b - [C:/Users/student.ECPC018/Desktop/111701011_111701013_lab3b/111701011_111701013_lab3b.xpr] - Vivado 2016.2

Synthesized Design - xc7z010clg400-1 (active)

Netlist: main — Nets (50), Leaf Cells (42)

Schematic: 42 Cells, 16 I/O Ports, 50 Nets

Debug: Unassigned Debug Nets (0)



111701011_111701013_lab3b - [C:/Users/student.ECPC018/Desktop/111701011_111701013_lab3b/111701011_111701013_lab3b.xpr] - Vivado 2016.2

Elaborated Design - xc7z010clg400-1 (active)

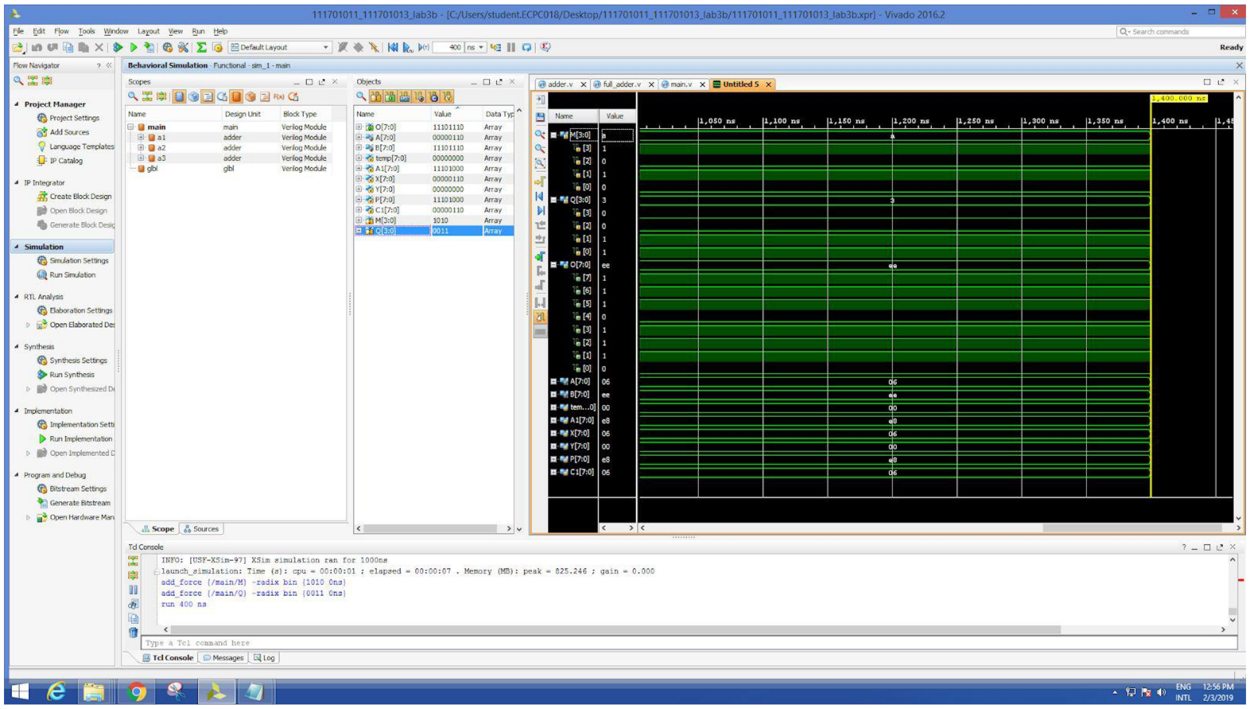
RTL Netlist: main — Nets (187), Leaf Cells (58), a1 (adder), a2 (adder), a3 (adder)

Schematic: 59 Cells, 16 I/O Ports, 187 Nets

Design Runs

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Failed Routes	LUT	FF	BRAM	URAM	DSP	Start	Elapsed	Strategy	Part	Description
✓ synth_1	constrs_1	synth_design Complete!							18	0	0	0	0	2/3/19 12:56 PM	00:00:16	Vivado Synthesis Defaults (Vivado Synthesis 2016)	xc7z010clg400-1	Vivado Synthesis Defaults
impl_1	constrs_1	Not started														Vivado Implementation Defaults (Vivado Implementation 2016)	xc7z010clg400-1	Default settings for Implementation.

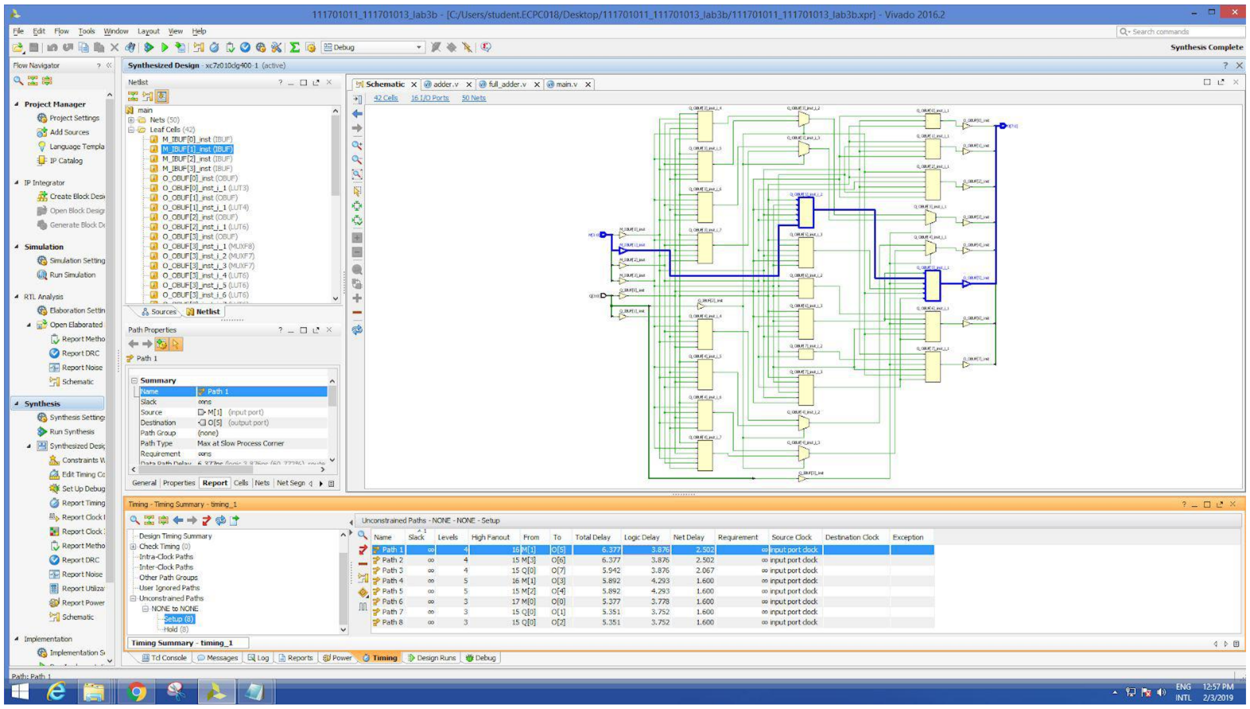
Simulation Results



Name	Value	Data Type
O[7:0]	11101110	Array
A[7:0]	00000110	Array
B[7:0]	11101110	Array
temp[7:0]	00000000	Array
A1[7:0]	11101000	Array
X[7:0]	00000110	Array
Y[7:0]	00000000	Array
P[7:0]	11101000	Array
C1[7:0]	00000110	Array
M[3:0]	1010	Array
Q[3:0]	0011	Array

```
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:01 ; elapsed = 00:00:07 . Memory (MB): peak = 825.246 ; gain = 0.000
add_force {/main/M} -radix bin {1010 0ns}
add_force {/main/Q} -radix bin {0011 0ns}
run 400 ns
```

Timing Results

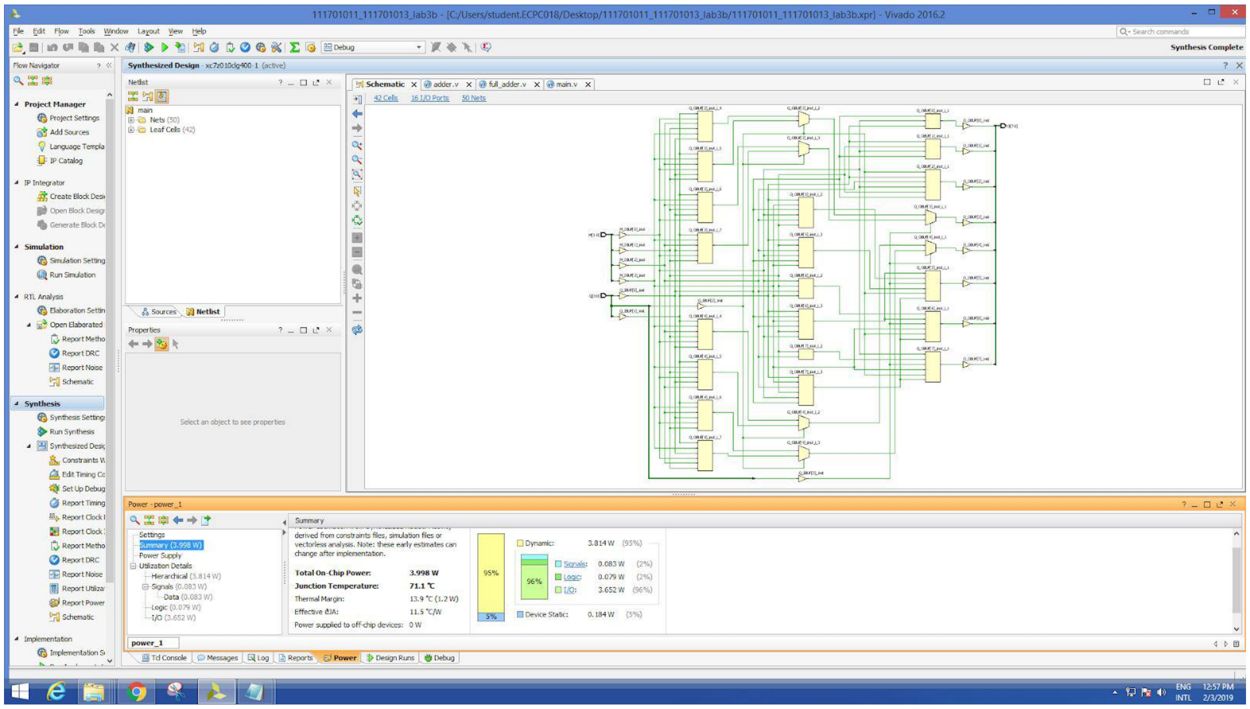


Summary	
Name	Path 1
Slack	∞ns
Source	M[1] (input port)
Destination	O[5] (output port)
Path Group	(none)
Path Type	Max at Slow Process Corner
Requirement	∞ns

Unconstrained Paths - NONE - NONE - Setup

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception
Path 1	∞	4	16	M[1]	O[5]	6.377	3.876	2.502	∞	input port clock		
Path 2	∞	4	15	M[3]	O[6]	6.377	3.876	2.502	∞	input port clock		
Path 3	∞	4	15	Q[0]	O[7]	5.942	3.876	2.067	∞	input port clock		
Path 4	∞	5	16	M[1]	O[3]	5.892	4.293	1.600	∞	input port clock		
Path 5	∞	5	15	M[2]	O[4]	5.892	4.293	1.600	∞	input port clock		
Path 6	∞	3	17	M[0]	O[0]	5.377	3.778	1.600	∞	input port clock		
Path 7	∞	3	15	Q[0]	O[1]	5.351	3.752	1.600	∞	input port clock		
Path 8	∞	3	15	Q[0]	O[2]	5.351	3.752	1.600	∞	input port clock		

Schematic and Power Results



111701011_111701013_lab3b - [C:/Users/student.ECPC018/Desktop/111701011_111701013_lab3b/111701011_111701013_lab3b.xpr] - Vivado 2016.2

Synthesized Design - xc7z010clg400-1 (active)

Power - power_1

Summary

derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

```
Total On-Chip Power:               3.998 W
Junction Temperature:              71.1 °C
Thermal Margin:                    13.9 °C (1.2 W)
Effective ϑJA:                     11.5 °C/W
Power supplied to off-chip devices: 0 W
```

```
Dynamic:        3.814 W  (95%)
  Signals:      0.083 W  (2%)
  Logic:        0.079 W  (2%)
  I/O:          3.652 W  (96%)
Device Static:  0.184 W  (5%)
```

Observations:

Implement an array multiplier that would multiply two unsigned numbers A and B:

Dynamic Power used : 4.000W

Static Power used : 0.192W

Total On Chip Power used : 4.192W

Static Power is 5% of the total power while dynamic power is 95%.

In dynamic power,

94% is used for I/O operations = 3.765W

2% is used in logic = 0.072W

4% is used in signals = 0.162W

Junction Temperature was 73.4°C

Implement Booth's algorithm that would multiply two 2's complement numbers A and B:

Dynamic Power used : 3.814W

Static Power used : 0.184W

Total On Chip Power used : 3.998W

Static Power is 5% of the total power while dynamic power is 95%.

In dynamic power,

96% is used for I/O operations = 3.652W

2% is used in logic = 0.079W

2% is used in signals = 0.083W

Junction Temperature was 71.1°C

1.) *In the first question, power consumption was more as compared to Multiplier circuit developed in second question .*
2.) *Junction Temperature for the first question is also higher than the Junction temperatures in the second question.*
3.) *In both the cases, dynamic power percentages are the same.*
4.) *Most of the dynamic power is consumed by the I/O.*
5.) *Rest of the dynamic power is used in logic and signals which is very less than input output dynamic power.*
6.) *More buffers and logic was there in the first logic as compared to the second one.*

Results:

1.) *Power consumption is less when there are less number of components are used at the same time.*
2.) *Power consumption reduces by a great factor when the number of components working parallely is decreased.*
3.) *But the time delay increases as the parallel logic is transformed into the pipeline logic.*
4.) *But the static power remains constant in both the logic.*
5.) *Time delay is lesser in second program than in the first program.*
6.) *Booth's algorithm is more efficient as compared to the array multiplier logic as it takes lesser power consumption as well as lesser total delay.*
7.) *Booth's Algorithm involve more dynamic processing as compared to the Array*

multiplier algorithm.

8.) *It takes 8 Full adder and 4 Half adder to implement 4x4 bit array multiplier.*

Conclusions:

1.) *More power is consumed in the parallel programming as compared to sequential (pipeline) programming.*

2.) *Static power consumed remains same as it is consumed when there is no activity on the circuit so it is independent on the circuit.*

3.) *Processing delay is more in case of pipeline fashion while it is less in parallel fashion.*