CS2610: Computer Organization and Architecture

<u>Lab 5: Report</u>

Devansh Singh Rathore(111701011), Himanshu Jain(111701013)

Objective

To design a simple 4-bit ALU and analyse the efficiency in terms of processing delay and power consumption.

Problem

In the previous labs you had already designed and analysed different 4-bit adder, subtractors and multipliers in Verilog.
a.) By choosing the best candidates from above, design a 4-bit ALU to perform fast addition, subtraction, bitwise AND and bitwise OR operations. There should be a control unit to select the required operation.
b.) Using the designed 4-bit ALU, implement an 8-bit ALU to perform the same operations.

Design

a.) 4-bit ALU

- ❏ We have used the carry look ahead 4-bit adder to add the two given 4-bit numbers as it has given us the least time delay.
- ❏ For subtraction we have calculated the 2’s complement of the second number to be subtracted and then adder to the first one using the above designed carry look ahead adder.
- ❏ For calculating 2’s complement - Starting from the least significant bit, the bit remains same till we encounter 1, after that all 0’s are converted to 1 and all 1’s are converted to 0.
- ❏ For bitwise AND and bitwise OR we have simply calculated it bitwise.
- ❏ For the control unit we have taken a 2-bit input from the user and checked if it is 00-addition, 01-Subtraction, 10-bitwise AND, 11-bitwise OR.

b.) 8-bit ALU using 4-bit ALU

- ❏ For addition we have passed the cout from the first four bits to second four bits and added the second four bit.
- ❏ For subtraction we have passed the complement of the cout from the first four bits to second four bits and then they were subtracted.
- ❏ For bitwise AND and bitwise OR 4-bit ALU was called two times, first for the starting four bits and second for the second set of four bits.
- ❏ For control unit same procedure was followed and the same input is passed to the 4-bit ALU.

Implementation

a.) 4-bit ALU

Code for the main program

```
module main(input [1:0] op, [3:0] a,b, output reg [3:0]out, output reg
cout);

    reg [3:0] a1,a2,b1,b2;
    wire [3:0] o1,o2;
    wire c1,c2;


    adder A(a1,b1,o1,c1);
    subtractor B(a2,b2,o2,c2);

    always@*
    begin
    if(op==2'b00)
    begin
    a1 = a;
    b1 = b;
    out = o1;
    cout=c1;
    end
    else if(op==2'b01)
    begin
    a2= a;
    b2 = b;
    out = o2;
    cout = c2;
    end
    else if(op==2'b10)
    begin
    out[0]=a[0]&b[0];
    out[1]=a[1]&b[1];
    out[2]=a[2]&b[2];
    out[3]=a[3]&b[3];
    cout = 0;
    end
    else
    begin
    out[0]=a[0]|b[0];
    out[1]=a[1]|b[1];
    out[2]=a[2]|b[2];
    out[3]=a[3]|b[3];
    cout = 0;
    end
    end

Endmodule
```

Code for the adder(carry look ahead 4-bit)

```
module adder(input [3:0] a,b, output [3:0] S,cout);

wire [3:0] C;

assign C[0] = a[0]&b[0];
assign C[1] = (a[1]&b[1]) | ((a[1]^b[1])&(a[0]&b[0]));
assign C[2] = (a[2]&b[2]) | ((a[2]^b[2])&(a[1]&b[1])) |
((a[2]^b[2])&(a[1]^b[1])&(a[0]&b[0]));
assign C[3] = (a[3]&b[3]) | ((a[3]^b[3])&(a[2]&b[2])) |
((a[3]^b[3])&(a[2]^b[2])&(a[1]&b[1]))
|((a[3]^b[3])&(a[2]^b[2])&(a[1]^b[1])&(a[0]&b[0]));
assign S[0] = a[0]^b[0];
assign S[1] = a[1]^b[1]^C[0];
assign S[2] = a[2]^b[2]^C[1];
assign S[3] = a[3]^b[3]^C[2];
assign cout = C[3];

endmodule
```

Code for the Subtractor

```
module subtractor (input [3:0] a,b, output [3:0] S,cout);

    wire [3:0] re;

    comp C (b,re);
    adder Ad(a,re,S,cout);


Endmodule
```

Code for the calculating the complement of a 4-bit number

```
module comp(input [3:0] a, output reg [3:0] o);

    always@*
    begin
    if(a[0]==1)
    begin
    o[0] = a[0];
    o[1] = ~a[1];
    o[2] = ~a[2];
    o[3] = ~a[3];
    end
    else if(a[1]==1)
    begin
    o[0] = a[0];
    o[1] = a[1];
    o[2] = ~a[2];
    o[3] = ~a[3];
    end
    else if(a[2]==1)
    begin
    o[0] = a[0];
    o[1] = a[1];
    o[2] = a[2];
    o[3] = ~a[3];
    end
    else
    begin
    o[0] = a[0];
    o[1] = a[1];
    o[2] = a[2];
    o[3] = a[3];
    end
    end

endmodule
```

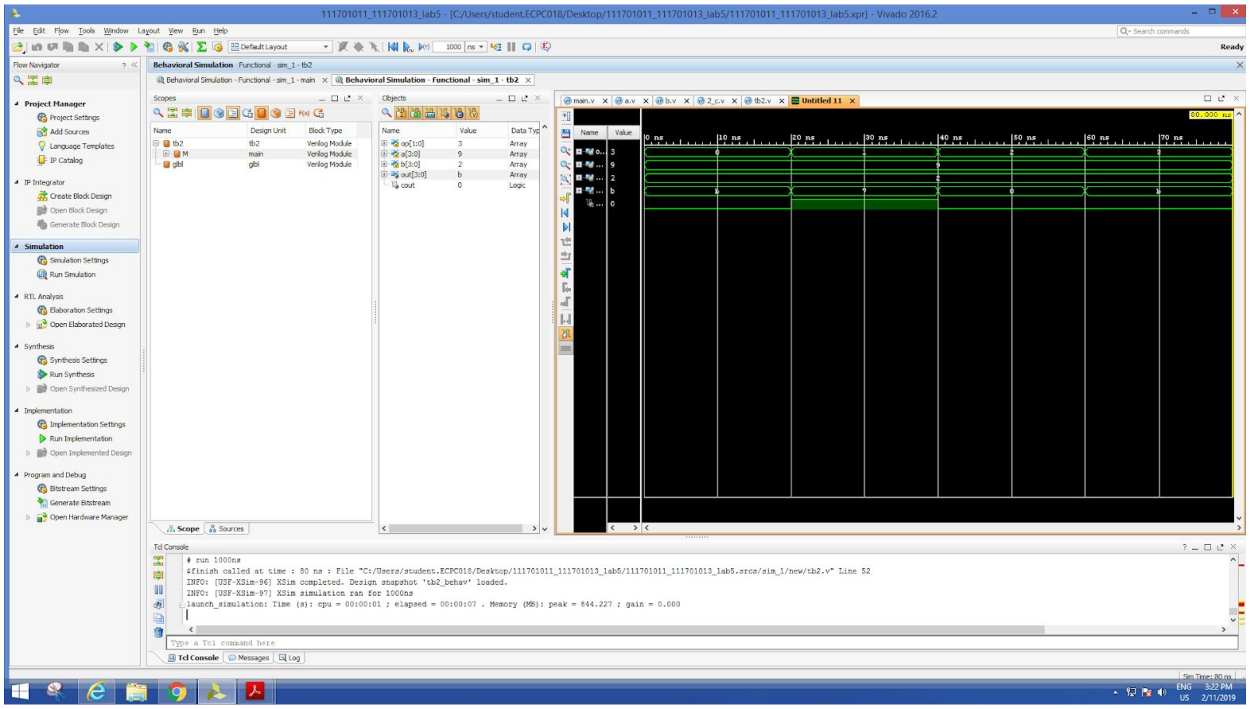
b.) 8-bit ALU using 4-bit ALU

```
module main(input [1:0]op, input [7:0] a,b, output [7:0] out,output cout);

    wire c1;
    reg c2;

    ALU A1 (op,a[3:0],b[3:0],0,out[3:0],c1);
    ALU A2 (op,a[7:4],b[7:4],c2,out[7:4],cout);

    always@*
    begin
    if(op==2'b00)
    begin
    c2 = c1;
    end
    else if(op==2'b01)
    begin
    c2 = ~c1;
    end
    else
    begin
    c2 = 0;
    end
    end

endmodule
```

Experiments

a.) 4-bit ALU

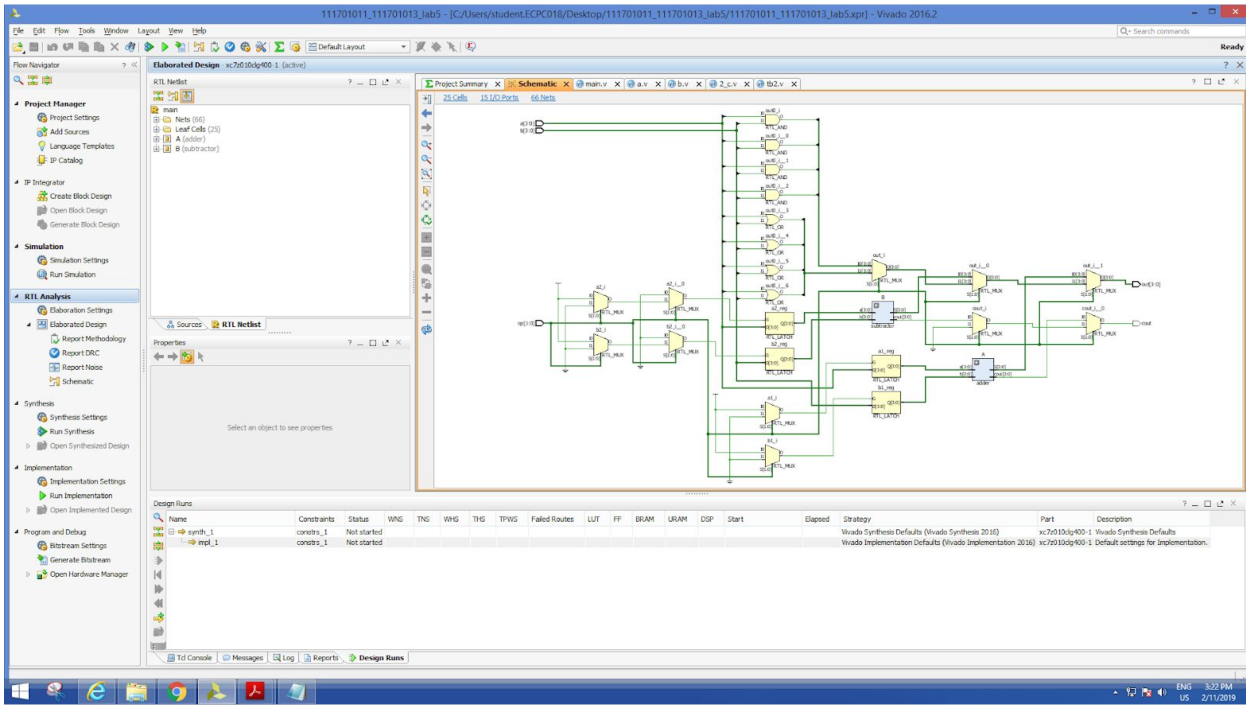
1. Simulation snapshots:



Name	Value	Data Type
op[1:0]	3	Array
a[3:0]	9	Array
b[3:0]	2	Array
out[3:0]	b	Array
cout	0	Logic

```
# run 1000ns
$finish called at time : 80 ns : File "C:/Users/student.ECPC018/Desktop/111701011_111701013_lab5/111701011_111701013_lab5.srcs/sim_1/new/tb2.v" Line 52
INFO: [USF-XSim-96] XSim completed. Design snapshot 'tb2_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:01 ; elapsed = 00:00:07 . Memory (MB): peak = 844.227 ; gain = 0.000
```

2. Schematic diagram:



111701011_111701013_lab5 - [C:/Users/student.ECPC018/Desktop/111701011_111701013_lab5/111701011_111701013_lab5.xpr] - Vivado 2016.2

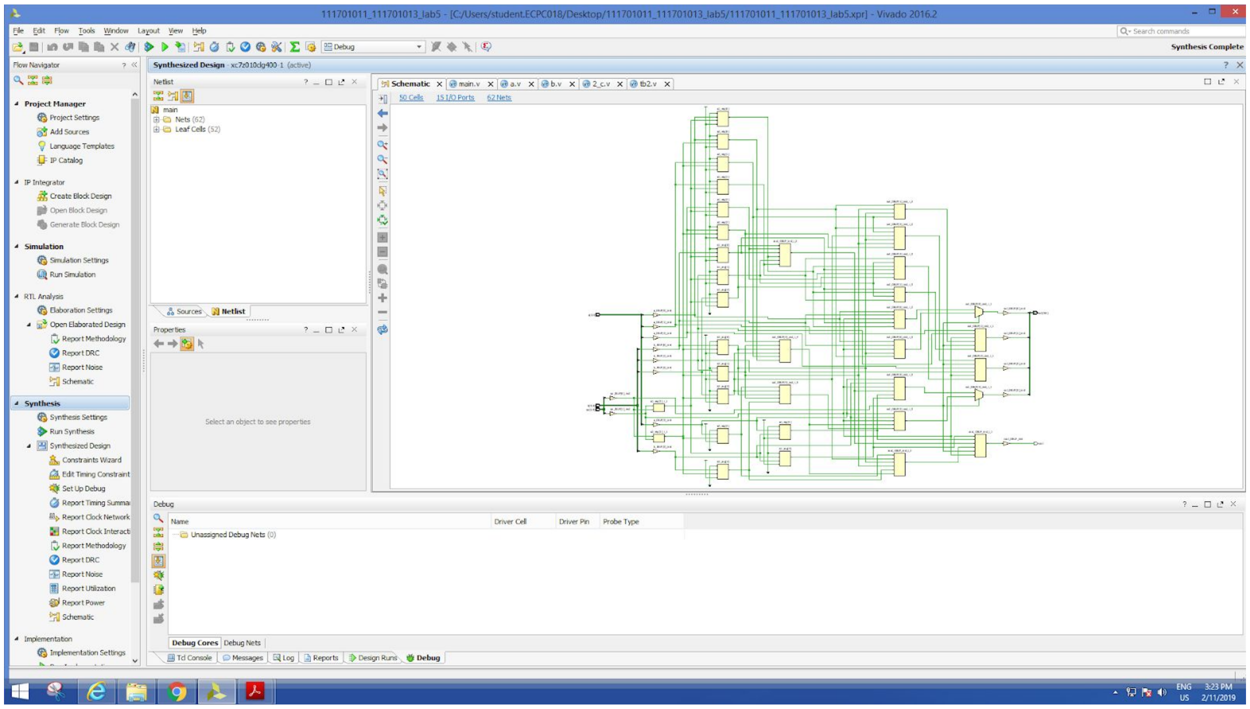
Elaborated Design - xc7z010clg400-1 (active)

RTL Netlist: main — Nets (66), Leaf Cells (25), A (adder), B (subtractor)

25 Cells · 15 I/O Ports · 66 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_lab5 - [C:/Users/student.ECPC018/Desktop/111701011_111701013_lab5/111701011_111701013_lab5.xpr] - Vivado 2016.2

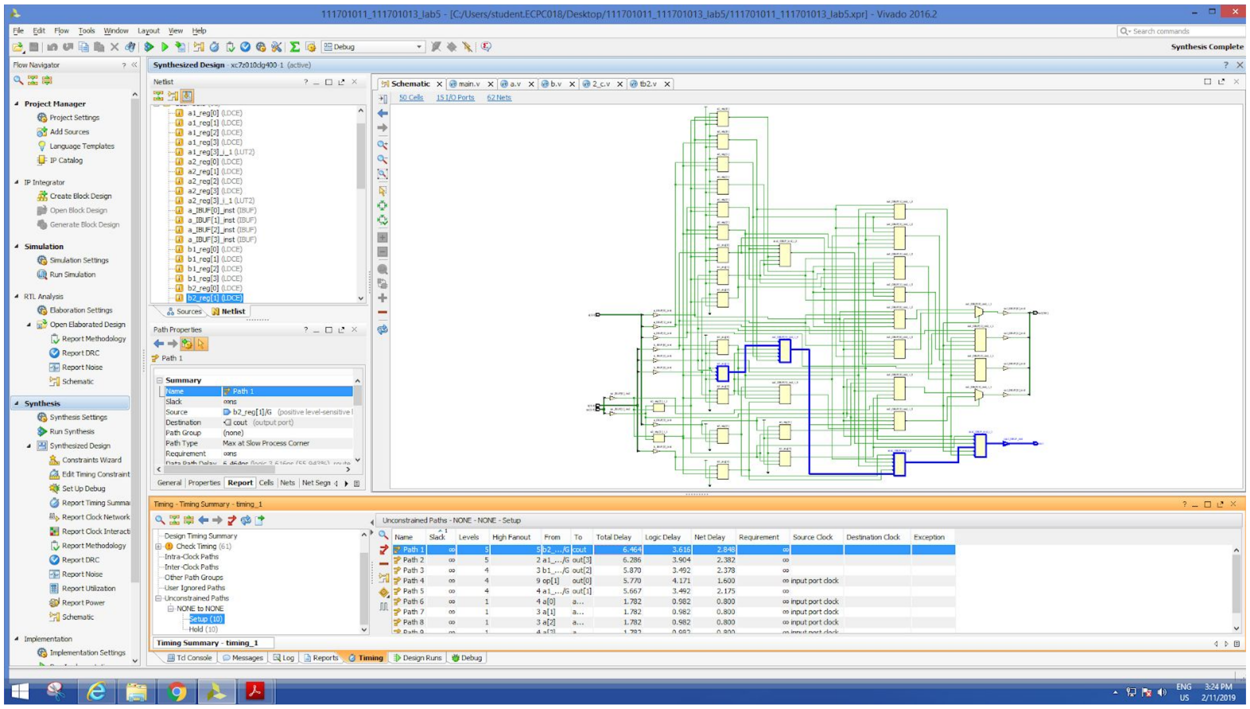
Synthesized Design - xc7z010clg400-1 (active)

Netlist: main — Nets (62), Leaf Cells (52)

50 Cells · 15 I/O Ports · 62 Nets

Debug: Unassigned Debug Nets (0)

3. Synthesis report.


111701011_111701013_lab5 - [C:/Users/student.ECPC018/Desktop/111701011_111701013_lab5/111701011_111701013_lab5.xpr] - Vivado 2016.2

Synthesized Design - xc7z010clg400-1 (active)

Path Properties

Path 1

Summary

Name	Path 1
Slack	∞ns
Source	b2_reg[1]/G (positive level-sensitive
Destination	cout (output port)
Path Group	(none)
Path Type	Max at Slow Process Corner
Requirement	∞ns

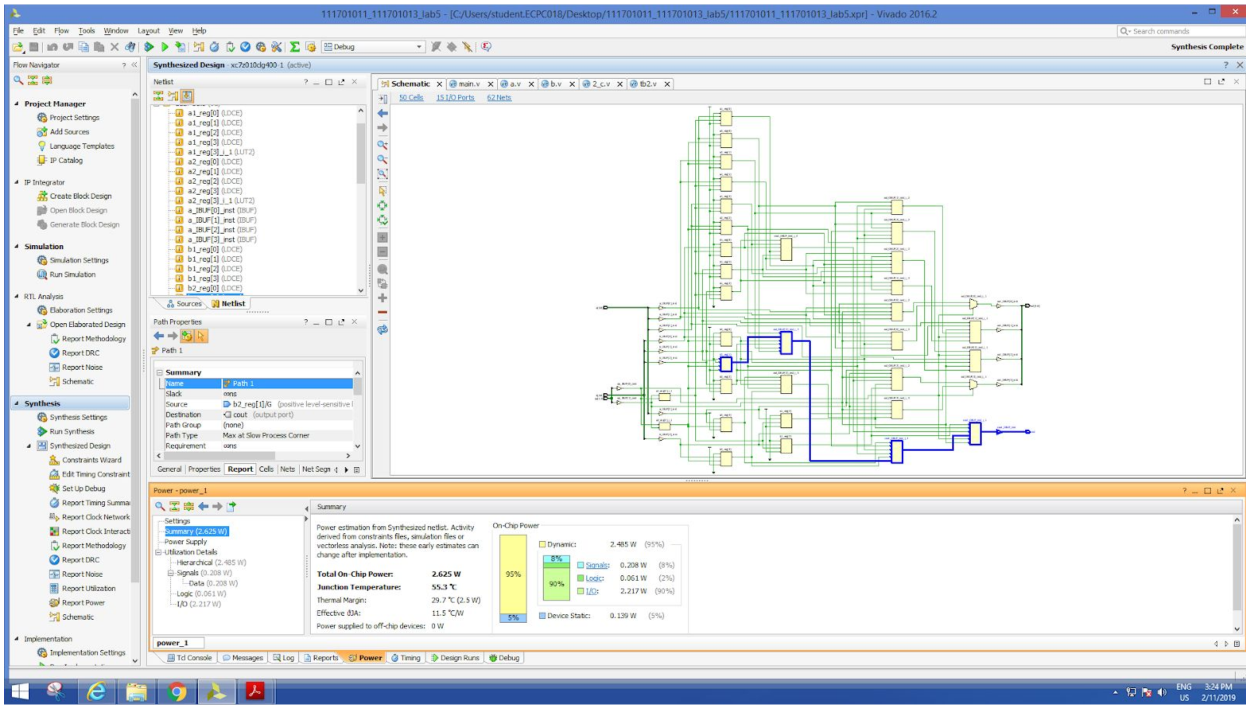
Timing - Timing Summary - timing_1

- Design Timing Summary
- Check Timing (61)
- Intra-Clock Paths
- Inter-Clock Paths
- Other Path Groups
- User Ignored Paths
- Unconstrained Paths
  - NONE to NONE
    - Setup (10)
    - Hold (10)

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	∞	5	5	b2_.../G	cout	6.464	3.616	2.848	∞			
Path 2	∞	5	2	a1_.../G	out[3]	6.286	3.904	2.382	∞			
Path 3	∞	4	3	b1_.../G	out[2]	5.870	3.492	2.378	∞			
Path 4	∞	4	9	op[1]	out[0]	5.770	4.171	1.600	∞	input port clock		
Path 5	∞	4	4	a1_.../G	out[1]	5.667	3.492	2.175	∞			
Path 6	∞	1	4	a[0]	a...	1.782	0.982	0.800	∞	input port clock		
Path 7	∞	1	3	a[1]	a...	1.782	0.982	0.800	∞	input port clock		
Path 8	∞	1	3	a[2]	a...	1.782	0.982	0.800	∞	input port clock		

Timing Diagram


111701011_111701013_lab5 - [C:/Users/student.ECPC018/Desktop/111701011_111701013_lab5/111701011_111701013_lab5.xpr] - Vivado 2016.2

Power - power_1

- Settings
- Summary (2.625 W)
- Power Supply
- Utilization Details
  - Hierarchical (2.485 W)
  - Signals (0.208 W)
    - Data (0.208 W)
  - Logic (0.061 W)
  - I/O (2.217 W)

Summary

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**2.625 W**
Junction Temperature:	**55.3 ℃**
Thermal Margin:	29.7 ℃ (2.5 W)
Effective ϑJA:	11.5 ℃/W
Power supplied to off-chip devices:	0 W

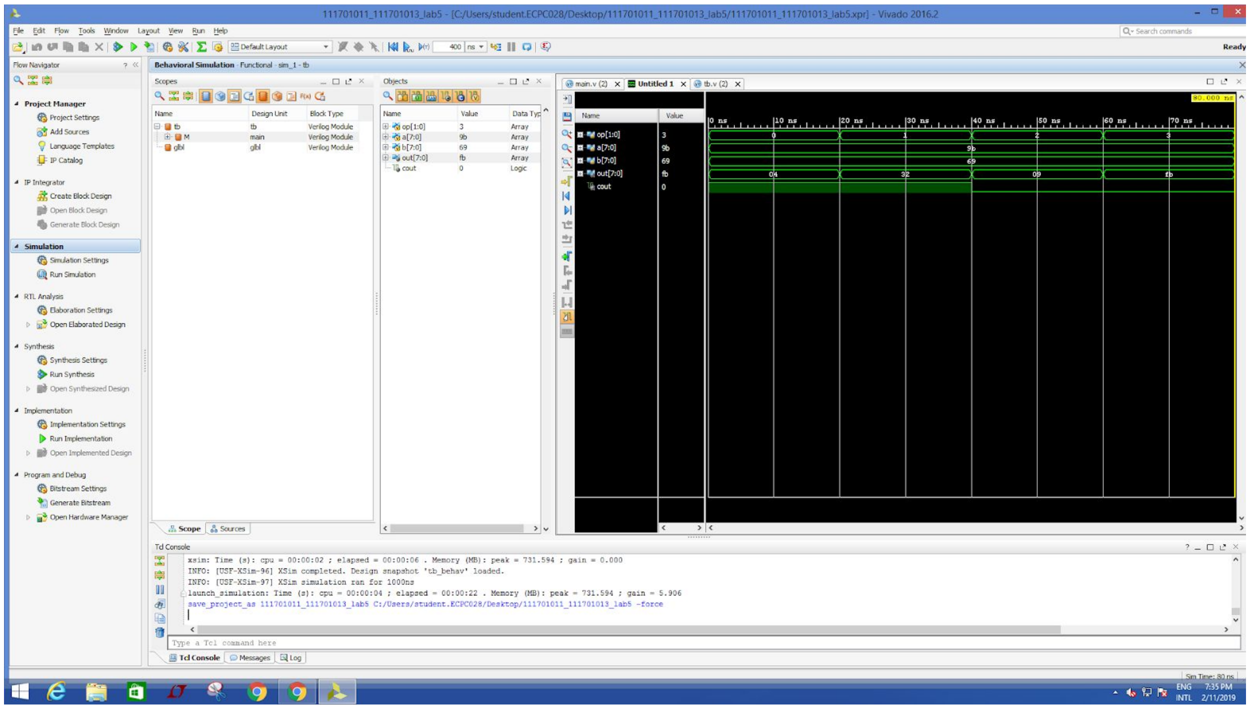
On-Chip Power

Dynamic:	2.485 W	(95%)
Signals:	0.208 W	(8%)
Logic:	0.061 W	(2%)
I/O:	2.217 W	(90%)
Device Static:	0.139 W	(5%)

Power Results

a.) 8-bit ALU using 4-bit ALU

1.) Simulation snapshots:



111701011_111701013_lab5 - [C:/Users/student.ECPC028/Desktop/111701011_111701013_lab5/111701011_111701013_lab5.xpr] - Vivado 2016.2

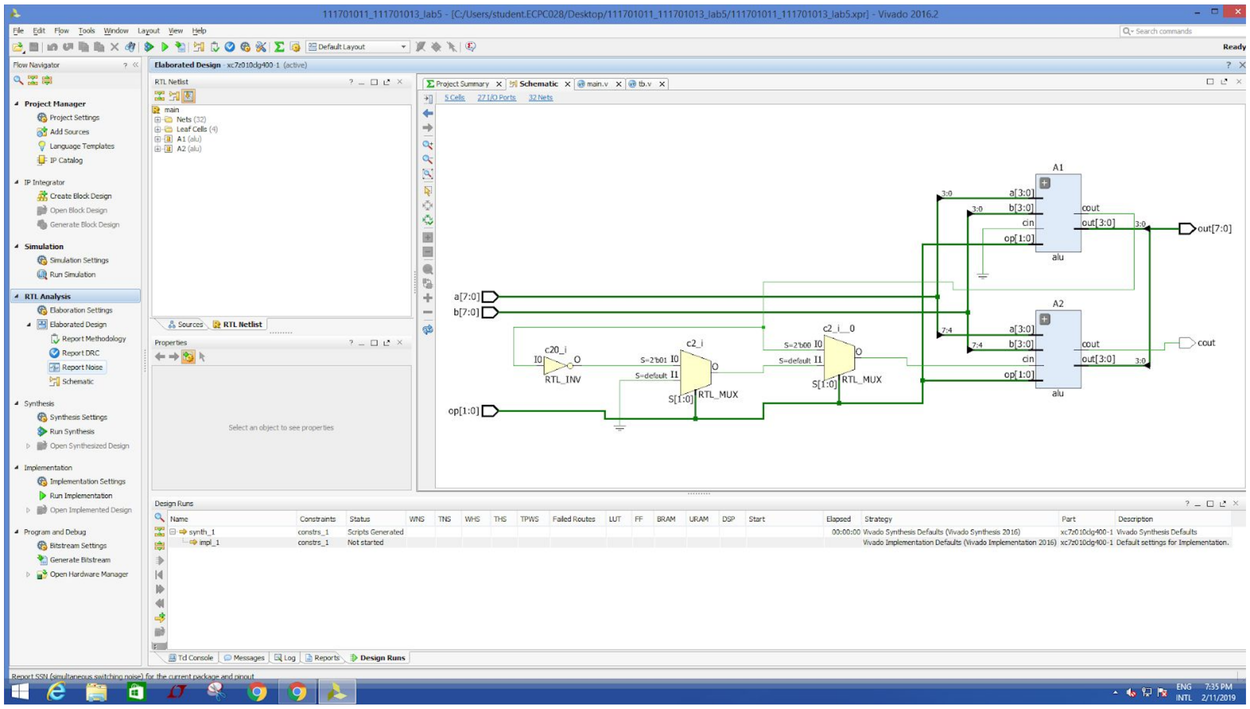
Behavioral Simulation - Functional - sim_1 - tb

Name	Value	Data Type
op[1:0]	3	Array
a[7:0]	9b	Array
b[7:0]	69	Array
out[7:0]	fb	Array
cout	0	Logic

Tcl Console

```
xsim: Time (s): cpu = 00:00:02 ; elapsed = 00:00:06 . Memory (MB): peak = 731.594 ; gain = 0.000
INFO: [USF-XSim-96] XSim completed. Design snapshot 'tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:04 ; elapsed = 00:00:22 . Memory (MB): peak = 731.594 ; gain = 5.906
save_project_as 111701011_111701013_lab5 C:/Users/student.ECPC028/Desktop/111701011_111701013_lab5 -force
```

2.) Schematic diagram:



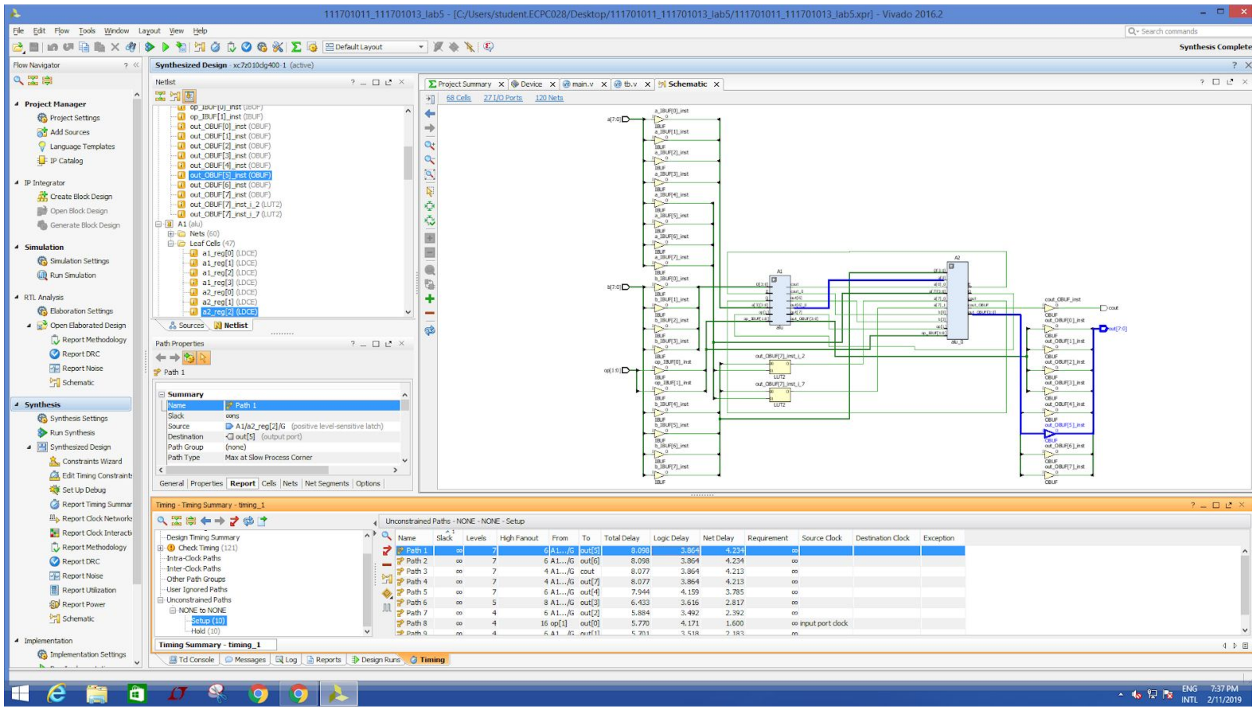
111701011_111701013_lab5 - [C:/Users/student.ECPC028/Desktop/111701011_111701013_lab5/111701011_111701013_lab5.xpr] - Vivado 2016.2

Elaborated Design - xc7z010clg400-1 (active)

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	Scripts Generated													00:00:00	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.

3.) Synthesis report.



111701011_111701013_lab5 - [C:/Users/student.ECPC028/Desktop/111701011_111701013_lab5/111701011_111701013_lab5.xpr] - Vivado 2016.2

Synthesized Design - xc7z010clg400-1 (active)

Path Properties — Path 1

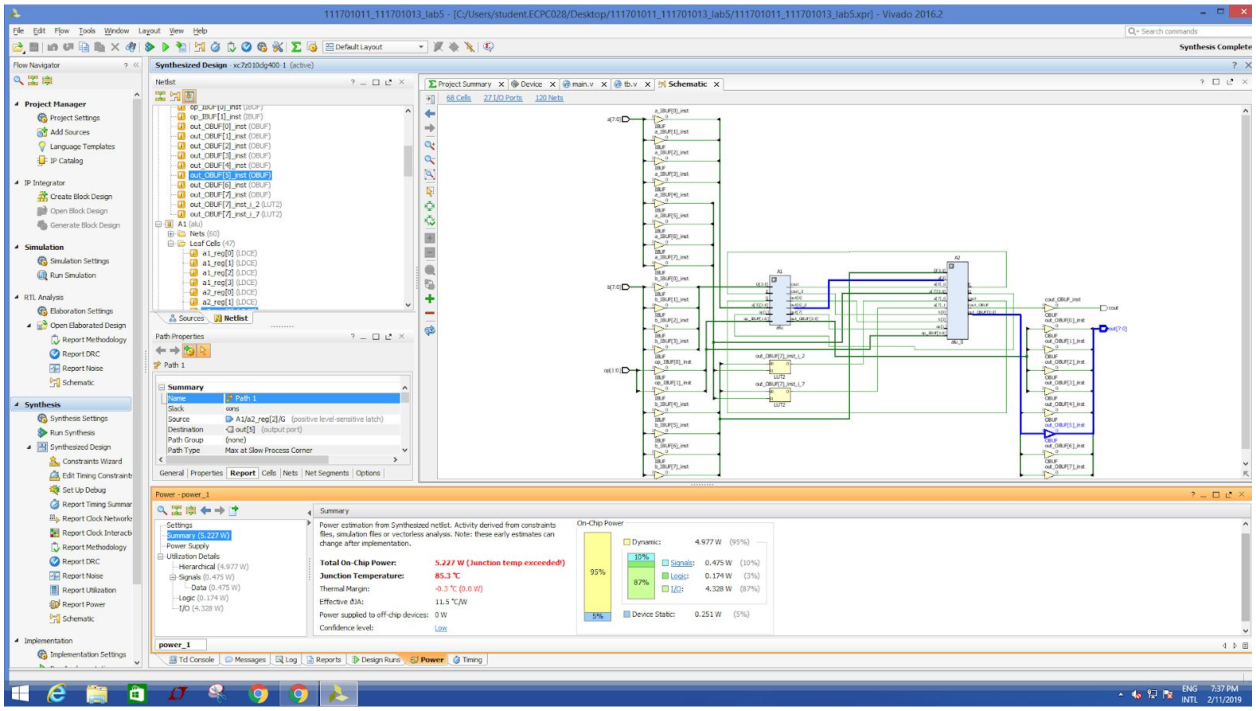
Summary	
Name	Path 1
Slack	∞ns
Source	A1/a2_reg[2]/G (positive level-sensitive latch)
Destination	out[5] (output port)
Path Group	(none)
Path Type	Max at Slow Process Corner

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	∞	7	6	A1.../G	out[5]	8.098	3.864	4.234	∞			
Path 2	∞	7	6	A1.../G	out[6]	8.098	3.864	4.234	∞			
Path 3	∞	7	4	A1.../G	cout	8.077	3.864	4.213	∞			
Path 4	∞	7	4	A1.../G	out[7]	8.077	3.864	4.213	∞			
Path 5	∞	7	6	A1.../G	out[4]	7.944	4.159	3.785	∞			
Path 6	∞	5	8	A1.../G	out[3]	6.433	3.616	2.817	∞			
Path 7	∞	4	6	A1.../G	out[2]	5.884	3.492	2.392	∞			
Path 8	∞	4	16	op[1]	out[0]	5.770	4.171	1.600	∞	input port clock		
Path 9	∞	4	6	A1.../G	out[1]	5.701	3.518	2.183	∞			

Timing Diagram



Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

```
Total On-Chip Power:               5.227 W (Junction temp exceeded!)
Junction Temperature:              85.3 °C
Thermal Margin:                    -0.3 °C (0.0 W)
Effective θJA:                     11.5 °C/W
Power supplied to off-chip devices: 0 W
Confidence level:                  Low
```

```
On-Chip Power
Dynamic:        4.977 W  (95%)
  Signals:      0.475 W  (10%)
  Logic:        0.174 W  (3%)
  I/O:          4.328 W  (87%)
Device Static:  0.251 W  (5%)
```

Power Results

Observations

Implementing 4-bit ALU to perform fast addition, subtraction, bitwise AND and bitwise OR operations:

Dynamic Power used : 2.485W
Static Power used : 0.139W
Total On Chip Power used : 2.625W
Static Power is 5% of the total power while dynamic power is 95%.
In dynamic power,
90% is used for I/O operations = 2.217W
2% is used in logic = 0.061W
8% is used in signals = 0.208W
Junction Temperature was 55.3 ° C
Total Delay = 6.464

Using the designed 4-bit ALU, implementing an 8-bit ALU to perform the same operations:

Dynamic Power used : 4.977W
Static Power used : 0.251W
Total On Chip Power used : 5.227W (Junction temperature exceeded)
Static Power is 5% of the total power while dynamic power is 95%.
In dynamic power,
87% is used for I/O operations = 4.328W

3% is used in logic = 0.174W
10% is used in signals = 0.475W
Junction Temperature was 85.3 °C
Total Delay = 8.098

Key observations in trends of power consumption and timings:

1. Implementation of the 4-bit ALU, consumed lesser power as compared to previous designs of Adder and Subtractor circuits individually.
2. However, total delay of 4-bit ALU is higher as compared to that of individual circuits of adder, subtractor which might be caused due to addition of extra components to switch between adder/subtractor/and operation/or operation.
3. More Dynamic power is consumed, due to higher pipeline structure of circuit design.
4. Designing 8-bit ALU consumed more power as well as has higher total delay than the 4-bit ALU circuits obviously because it has denser component circuit structure and involves double the processes.
5. Power consumption seems to be doubled(approx.) in 8-bit ALU implementation as compared to the 4-bit ALU implementation, which is expected because it involves approximately double the algorithm(and logical) steps and deals with registers and other data types of nearly double the size of that used in the 4-bit ALU implementation.
6. In the case of total delay, total time delay in the 8-bit ALU implementation is lesser than the double of the total time delay in the 4-bit ALU, as:
    - For (bitwise OR/AND) operations: Calculations in first half of 8-bit ALU (0th to 3rd bit) and (4th to 7th bit) goes in parallel(static way) without influencing each other.
    - For Adder/Subtractor operations: Calculations in first half of 8-bit ALU (0th to 3rd bit)
      for carry/borrow affects the other half (4th to 7th bit) and then calculation of rest of the process goes in parallel structure.

7. Inconsistency: Junction temperature exceeded in the implementation of the 8-bit ALU, mainly due to working with registers of large size, and complex logical circuit used.

Conclusions

We analyzed the power efficiency and total time delay of 4 bit ALU, and constructed 8 bit ALu using two such ALU's .
Further, more bit ALU's can be constructed using 4 bit ALU's and 4 bit ALU's can also be constructed using 4 1-bit ALU.
Thus using 1-bit ALU we can design 4-bit ALU and then using 4-bit we can design 8-bit and so on.