EEL2020 Digital Design Lab Report

Sem II AY 2023-24

Experiment No.	**:**	**06**
Name	**:**	**Lavangi Parihar**
Roll No.	**:**	**B22EE044**
Partner Name (Roll Number)	**:**	**Mansi Chaoudhary (B22EE045)**

Objective

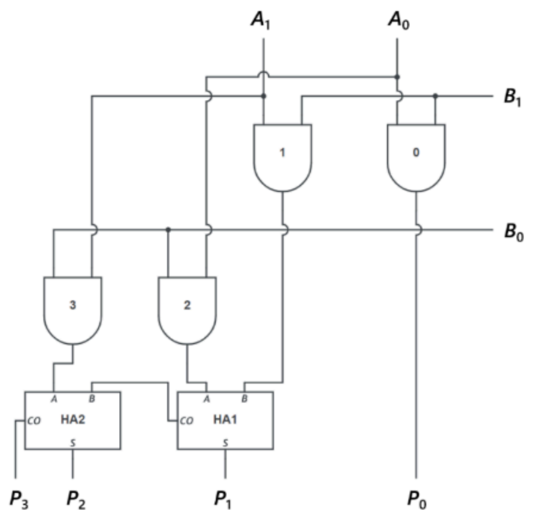
(i) Design, simulate and implement a 2-bit multiplier using half adders and AND gates

(ii) Design and simulate an Arithmetic Logic Unit (ALU) that performs eight operations selected using operation codes

PART I TWO BIT MULTIPLIER

Logic Design

The 2-bit multiplication output be a 4-bit product



Source Description

- Design source

```
module half_adder(
    input A,
    input B,
    output Sum,
    output Cout
);
    assign Sum=A^B;
    assign Cout=A&B;
endmodule


module two_bit_multiplier(input [1:0] A,B, output[3:0] P);
wire C;
assign P[0]=A[0]&B[0];

half_adder HA1((A[1] & B[0]), (A[0] & B[1]),P[1],C);
half_adder HA2(C,(A[1] & B[1]), P[2], P[3]);

endmodule
```

- Constraint file

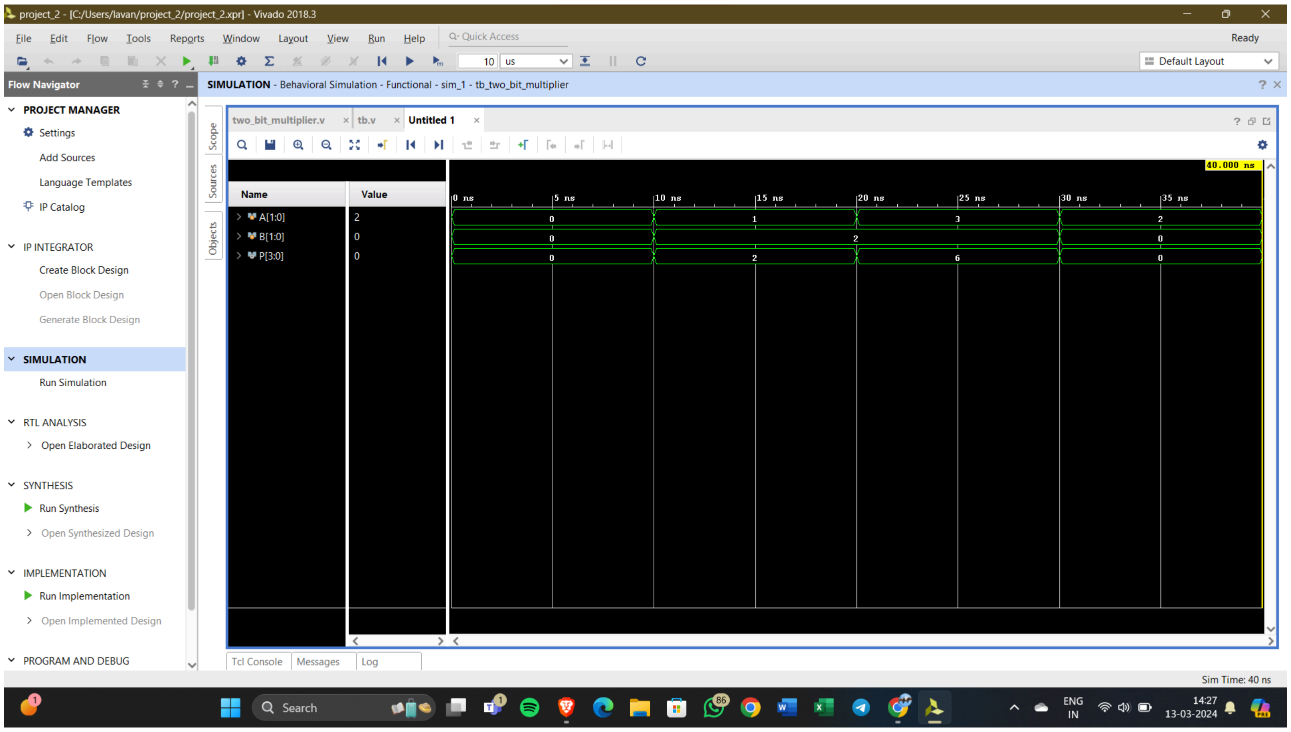
The PYNQ_XDC file was updated with the following changes:

Variable	Designation	Port
P[0]	Output	LED3
P[1]	Output	LED2
P[2]	Output	LED1
P[3]	Output	LED0

- Simulation source (if any)

```
module tb_two_bit_multiplier;
  reg [1:0] A, B;
  wire [3:0] P;

  two_bit_multiplier uut (.A(A),.B(B),.P(P));

  initial begin
    A = 2'b00;
    B = 2'b00;
    #10;
    A= 2'b01;
    B= 2'b10;
    #10
    A= 2'b11;
    B= 2'b10;
    #10;
    A= 2'b10;
    B= 2'b00;
    #10 $finish;
  end

endmodule
```

Simulation Results (Timing diagram)



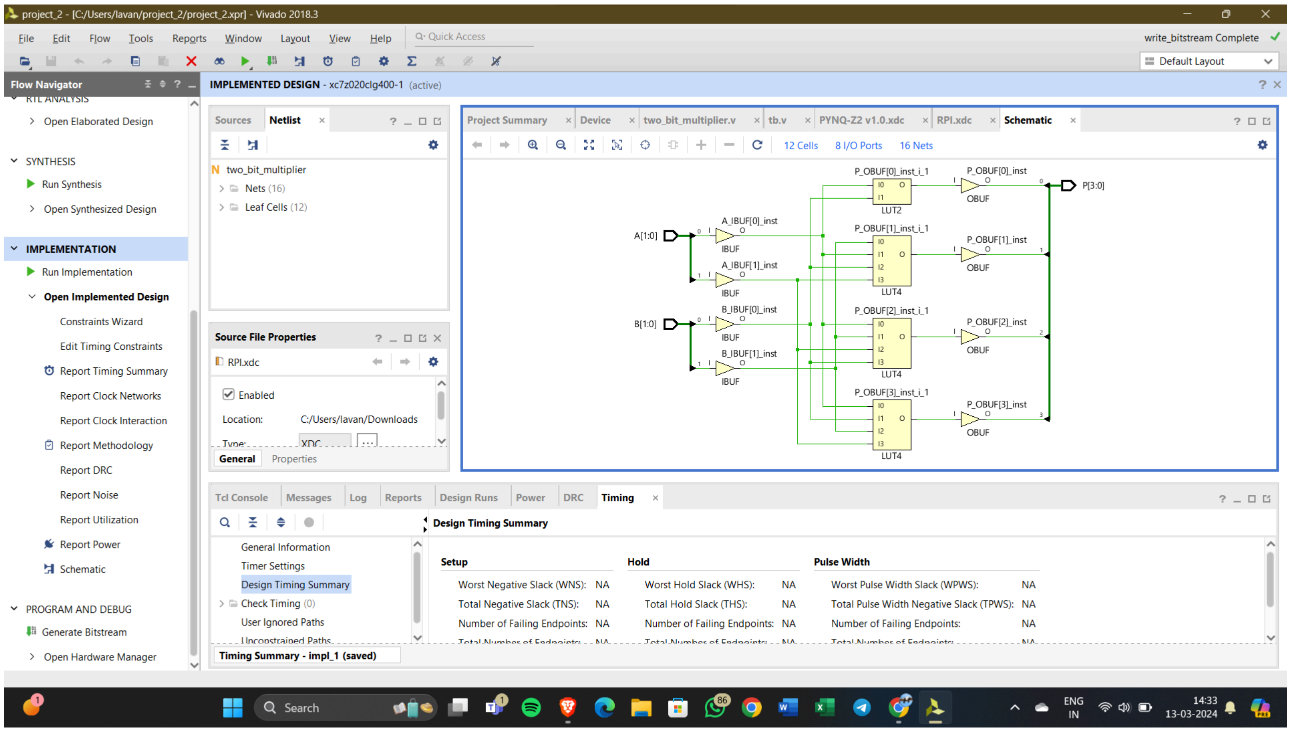
project_2 - [C:/Users/lavan/project_2/project_2.xpr] - Vivado 2018.3

SIMULATION - Behavioral Simulation - Functional - sim_1 - tb_two_bit_multiplier

Name	Value	0 ns	10 ns	20 ns	30 ns
A[1:0]	2	0	1	3	2
B[1:0]	0	0	2	2	0
P[3:0]	0	0	2	6	0

Sim Time: 40 ns

Elaborated Design:



Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	NA	Worst Hold Slack (WHS):	NA	Worst Pulse Width Slack (WPWS):	NA
Total Negative Slack (TNS):	NA	Total Hold Slack (THS):	NA	Total Pulse Width Negative Slack (TPWS):	NA
Number of Failing Endpoints:	NA	Number of Failing Endpoints:	NA	Number of Failing Endpoints:	NA
Total Number of Endpoints:	NA	Total Number of Endpoints:	NA	Total Number of Endpoints:	NA

PYNQ Working Video (to be recorded during lab session)

(with voiceover briefly explaining the working, not exceeding 1-2 minutes)

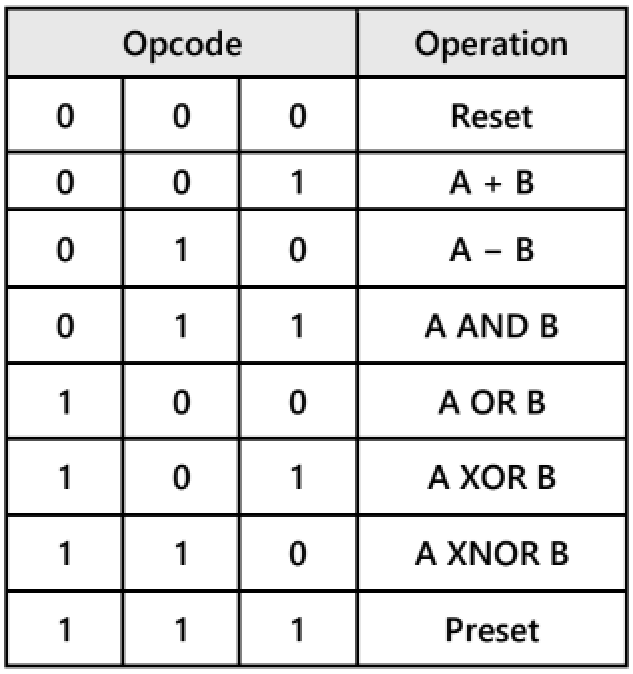
Video Link

PART II ARITHMETIC LOGIC UNIT

Logic Design

The ALU, a vital CPU component, executes arithmetic and logical operations (e.g., addition, subtraction, AND, OR) based on control unit instructions.

Truth Table:



Opcode			Operation
0	0	0	Reset
0	0	1	A + B
0	1	0	A – B
0	1	1	A AND B
1	0	0	A OR B
1	0	1	A XOR B
1	1	0	A XNOR B
1	1	1	Preset

Source Description

- Design source

```
module ALU ( input [2:0] opcode,  input [3:0] A,  input [3:0] B,  output reg [3:0] result );

always @(*)
```

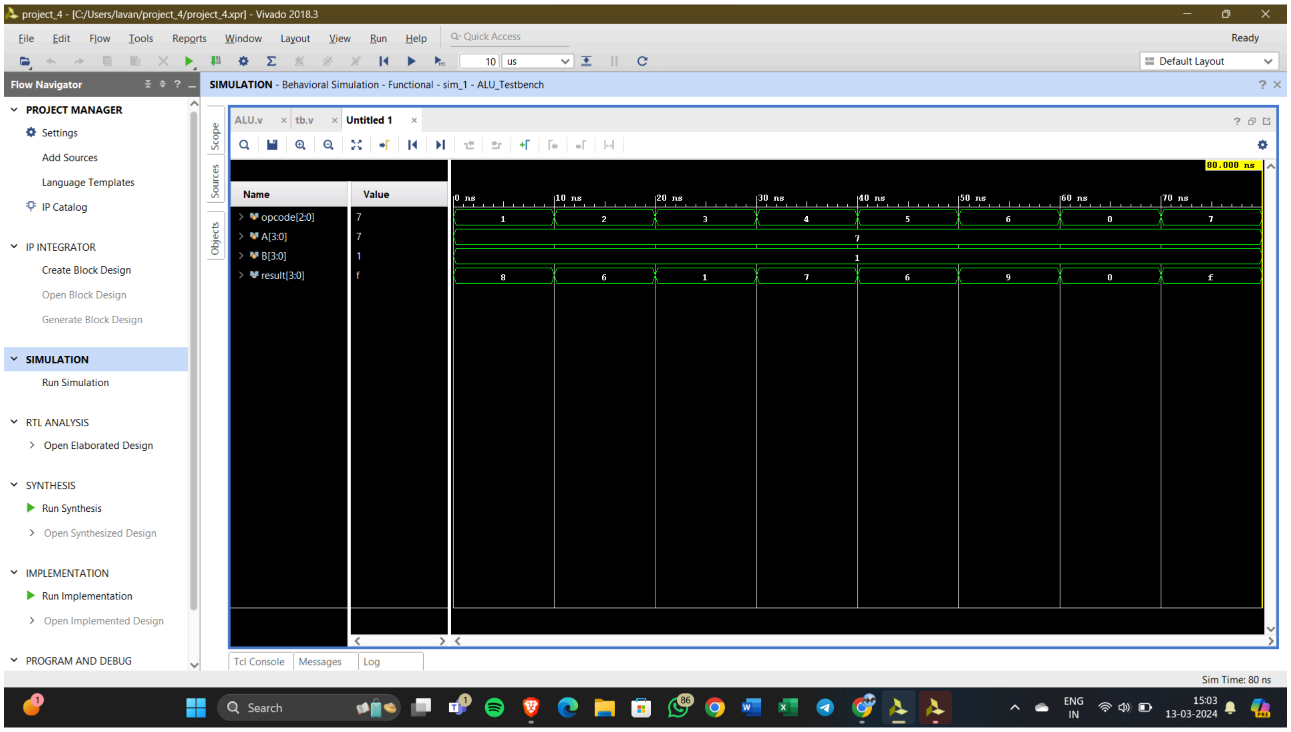
```
begin
   case(opcode)
      3'b000: result = 4'b0000;
      3'b001: result = A + B;
      3'b010: result = A - B;
      3'b011: result = A & B;
      3'b100: result = A | B;
      3'b101: result = A ^ B;
      3'b110: result = ~(A ^ B);
      3'b111: result = 4'b1111;
      default: result = 4'bxxxx; // Undefined
   endcase
end

endmodule
```

- Simulation source (if any)

```
module ALU_Testbench;

// Inputs
reg [2:0] opcode;
reg [3:0] A;
reg [3:0] B;

// Outputs
wire [3:0] result;

// Instantiate the ALU
ALU dut (
   .opcode(opcode),
   .A(A),
   .B(B),
   .result(result)
   );

// Stimulus
initial begin
   // Test case 1
   opcode = 3'b001; // Addition
   A = 4'b0111;
   B = 4'b0001;
   #10;

   // Test case 2
```

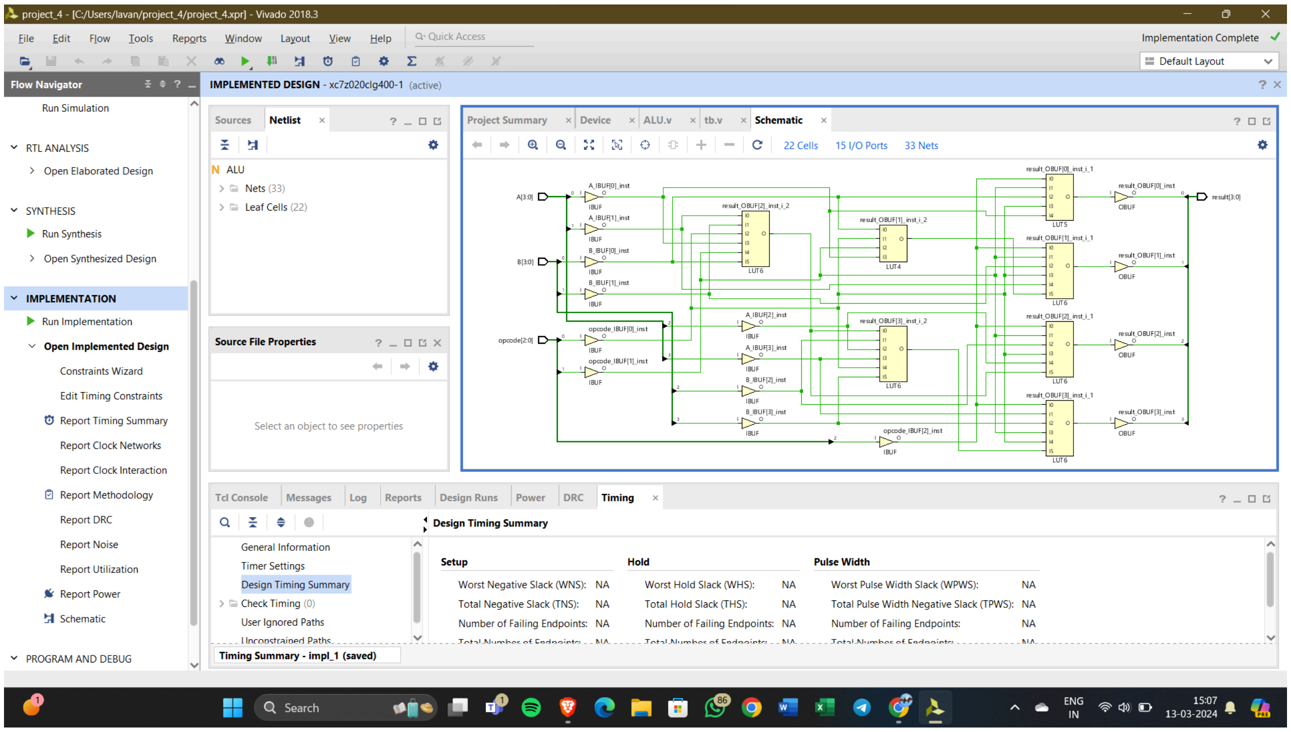
```
    opcode = 3'b010; // Subtraction
    A = 4'b0111;
    B = 4'b0001;
    #10;

    // Test case 3
    opcode = 3'b011; // AND
    A = 4'b0111;
    B = 4'b0001;
    #10;

    // Test case 4
    opcode = 3'b100; // OR
    A = 4'b0111;
    B = 4'b0001;
    #10;

    // Test case 5
    opcode = 3'b101; // XOR
    A = 4'b0111;
    B = 4'b0001;
    #10;

    // Test case 6
    opcode = 3'b110; // XNOR
    A = 4'b0111;
    B = 4'b0001;
    #10;

    // Test case 7
    opcode = 3'b000; // Reset
    A = 4'b0111;
    B = 4'b0001;
    #10;

    // Test case 8
    opcode = 3'b111; // Preset
    A = 4'b0111;
    B = 4'b0001;
    #10;

    $stop;
  end

  endmodule
```

Simulation Results (Timing diagram)


project_4 - [C:/Users/lavan/project_4/project_4.xpr] - Vivado 2018.3
SIMULATION - Behavioral Simulation - Functional - sim_1 - ALU_Testbench
ALU.v
tb.v
Untitled 1
Name
Value
opcode[2:0]
A[3:0]
B[3:0]
result[3:0]
80.000 ns
0 ns
10 ns
20 ns
30 ns
40 ns
50 ns
60 ns
70 ns
Sim Time: 80 ns

Name	Value	0 ns	10 ns	20 ns	30 ns	40 ns	50 ns	60 ns	70 ns
opcode[2:0]	7	1	2	3	4	5	6	0	7
A[3:0]	7	7							
B[3:0]	1	1							
result[3:0]	f	8	6	1	7	6	9	0	f

Elaborated Design:



Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	NA	Worst Hold Slack (WHS):	NA	Worst Pulse Width Slack (WPWS):	NA
Total Negative Slack (TNS):	NA	Total Hold Slack (THS):	NA	Total Pulse Width Negative Slack (TPWS):	NA
Number of Failing Endpoints:	NA	Number of Failing Endpoints:	NA	Number of Failing Endpoints:	NA

Timing Summary - impl_1 (saved)