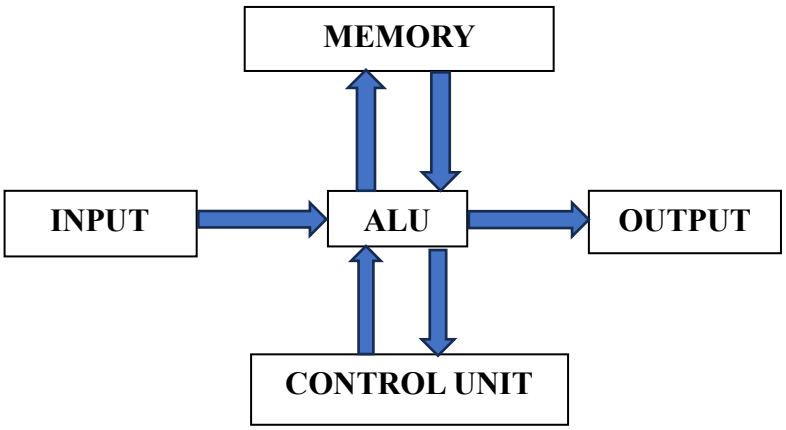
# DESIGN OF 8-BIT ALU IN XILINK VIVADO

1.Aim: To design a 8bit ALU which performs 8 different functions using Verilog.

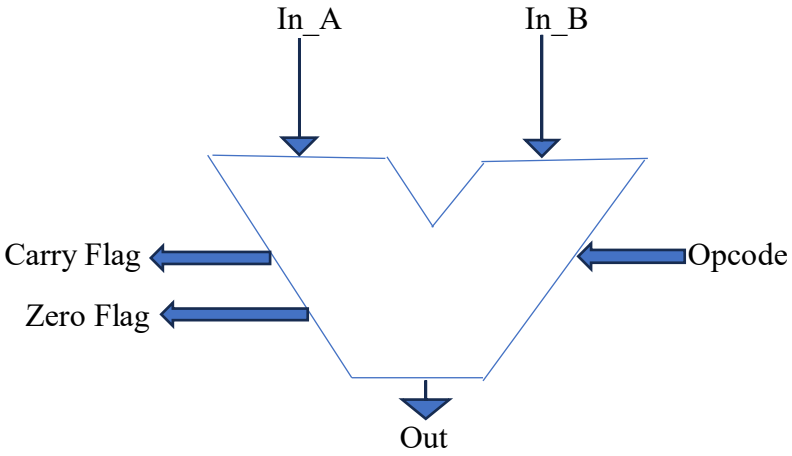
2.Software used: Xilink Vivado.

3.Project Description:

One of the main block in the processor is Arithmetic Logic Unit (ALU). If we take any processor there will be a input block , ALU, Control unit, Output block, then there will be some memory. ALU is the main important block in any processor.




In this project ALU takes two inputs In_A and In_B along with an opcode and generates an output with carry flag as well as zero flag.




Functions performed by ALU:

There are totally eight functions performed by an ALU among them first three functions are arithmetic operations such as addition, subtraction and multiplication, remaining five functions comes under logical operations like AND,OR,NAND,NOR,XOR.

| S.No | Opcode | Operation |
|---|---|---|
| 1 | 000 | result=operand1&operand2 |
| 2 | 001 | result=operand1-operand2 |
| 3 | 010 | result=operand1*operand2 |
| 4 | 011 | result=operand1&operand2 |
| 5 | 100 | result=operand1\|operand2 |
| 6 | 101 | result=~(operand1&operand2) |
| 7 | 110 | result=~(operand1\|operand2) |
| 8 | 111 | result=operand1^operand2 |

- Carryflag will be set whenever there is a carry out of the MSB bit.
- Zeroflag will be set if the result of the operation is zero i.e, when result is equal to zero then the zero flag will be set, if the result is not equal to zero then this will be reset.

**Example:**

Let In_A=10001100(8CH in hexadecimal), In_B=10011100(9CH in hexadecimal), and let the opcode be 000. Hence according to the table mentioned opcode 000 performs addition operation.

$$\begin{array}{r} 10001100 \\ +\ 10011100 \\ \hline \text{result=}\ 100101000 \end{array}$$

End result obtained as 28H and exceeds 8bit, MSB bit is a carry bit hence carry flag will be set.

4.Code:

Design Code:

```
// module declaration
module ALU8bit(opcode,operand1,operand2,result,flagc,flagz);
//input declaration
input[2:0]opcode;
input[7:0]operand1,operand2;
//output declaration
output reg[15:0] result=16'b0;
output reg flagc=1'b0,flagz=1'b0;
```

```
//parameter declaration which decided what operation is to be performed
parameter [2:0] ADD=3'b000,
SUB=3'b001,
MUL=3'b010,
AND=3'b011,
OR=3'b100,
NAND=3'b101,
NOR=3'b110,
XOR=3'b111;
//always block
always@(opcode or operand1 or operand2)
begin
case(opcode)
ADD:begin
result=operand1+operand2;
//The MSB bit of the result gets stored in carry flag, based on the MSB bit it will be decided whether carry flag is to be set or not
flagc=result[8];
//we are using comparison equation to check whether result is equal to zero or not if it is zero zero flag will be set
flagz=(result==16'b0);
end
SUB:begin
result=operand1-operand2;
flagc=result[8];
flagz=(result==16'b0);
end
MUL:begin
result=operand1*operand2;
flagc=result[8];
flagz=(result==16'b0);
```

```
end
AND:begin
result=operand1&operand2;
flagc=result[8];
flagz=(result==16'b0);
end
OR:begin
result=operand1|operand2;
flagc=result[8];
flagz=(result==16'b0);
end
NAND:begin
result=~(operand1&operand2);
flagc=result[8];
flagz=(result==16'b0);
end
NOR:begin
result=~(operand1|operand2);
flagc=result[8];
flagz=(result==16'b0);
end
XOR:begin
result=operand1^operand2;
flagc=result[8];
flagz=(result==16'b0);
end
//If none of the opcodes were selected(i.e from 000 to 111) will go for default statement
default:begin
result=16'b0;
flagc=1'b0;
```

```
flagz=1'b0;
end
endcase
end
endmodule
```

Testbench Code:

```
// represents time unit and time precision
`timescale 1ns / 1ps
//module declaration
module ALU8bit_tb;
//reg and wires declaration
reg [2:0]opcode;
reg [7:0]operand1;
reg [7:0]operand2;
wire [15:0] result;
wire flagc;
wire flagz;
reg [2:0] count=3'd0;
//ALU instantiation
ALU8bit
uut(.opcode(opcode),.operand1(operand1),.operand2(operand2),.result(result),.flagc(flagc),.fl
agz(flagz));
//initial block
initial begin
//initialization
opcode=3'b0;
operand1=8'd0;
operand2=8'd0;
#100;
operand1=8'hAA;
```

```
operand2=8'h55;
//opcode iteration
for(count=0;count<8;count=count+1'b1)
begin
opcode=count;
#20;
end
end
endmodule
```

## 5. Schematic View

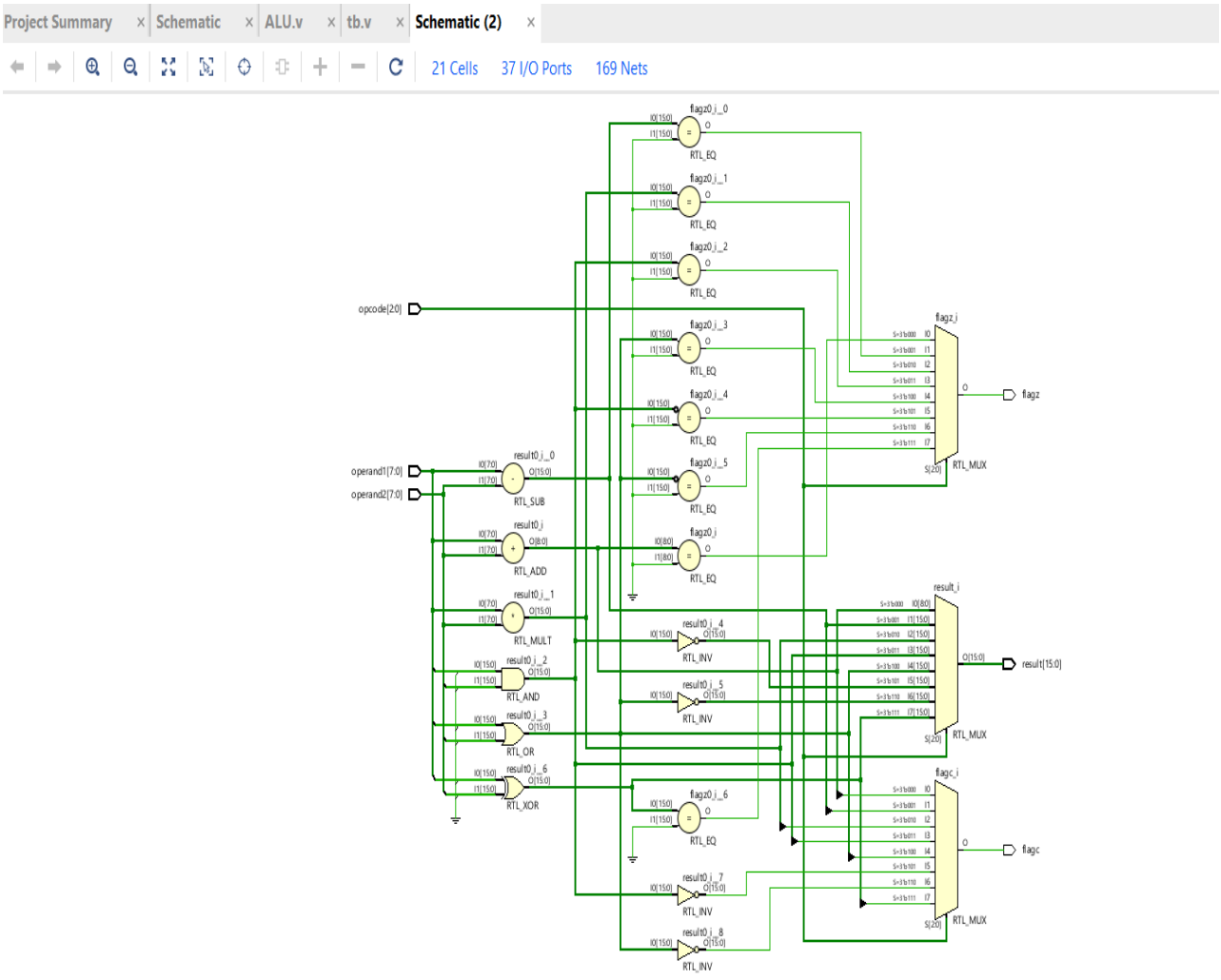



Fig-1: Schematic view of a 8 bit ALU

## 6. Simulation Results

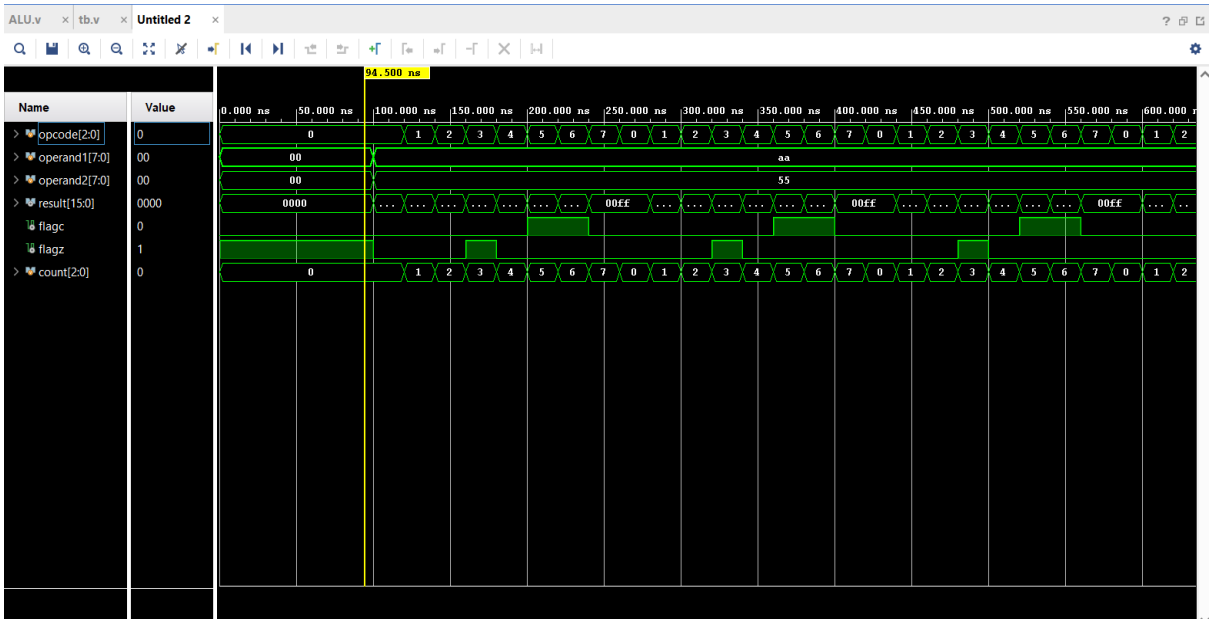



Fig-2: Simulation results when opcode is 0

Fig-2 denotes that operand1 and operand2 are zeroes, opcode chosen zero which performs addition operation hence result will be zero so that zero flag will be set.

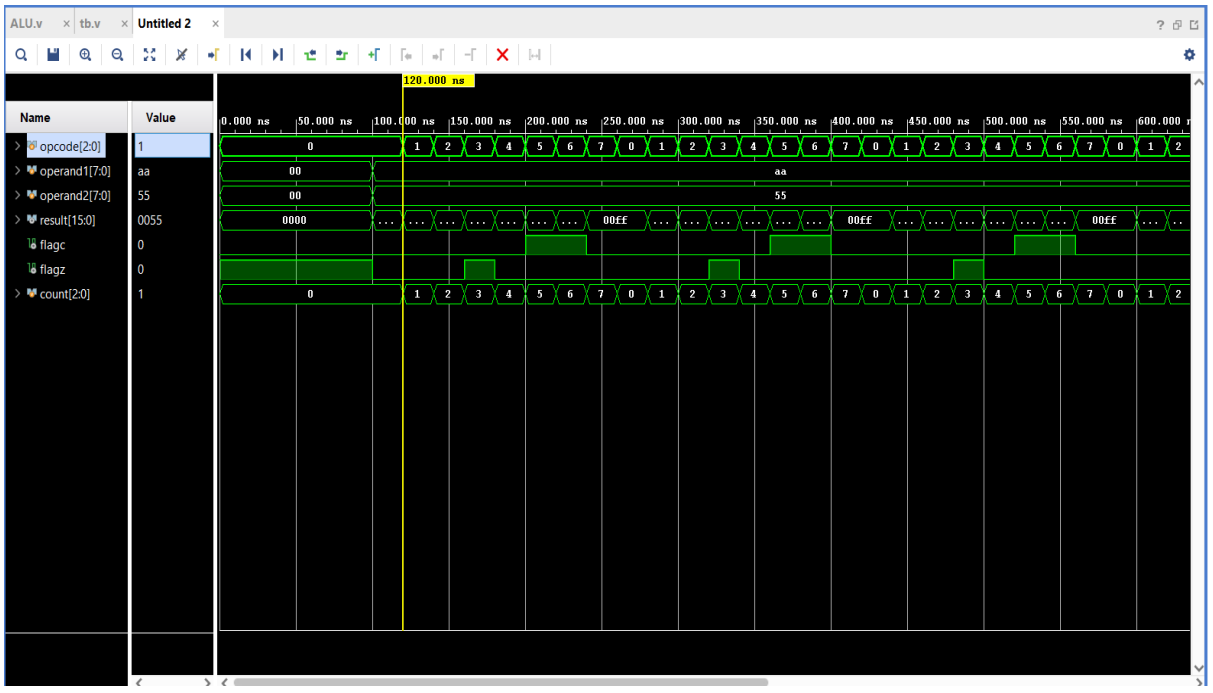



Fig-3: Simulation results when opcode is 1

Here operand1 is aa and operand2 is 55, whereas opcode is 1 which performs subtraction as we know value of a is 10, aa-55 result is 0055.As MSB bit is zero and end result is not equal to zero both carry flag and zero flag will not be set.

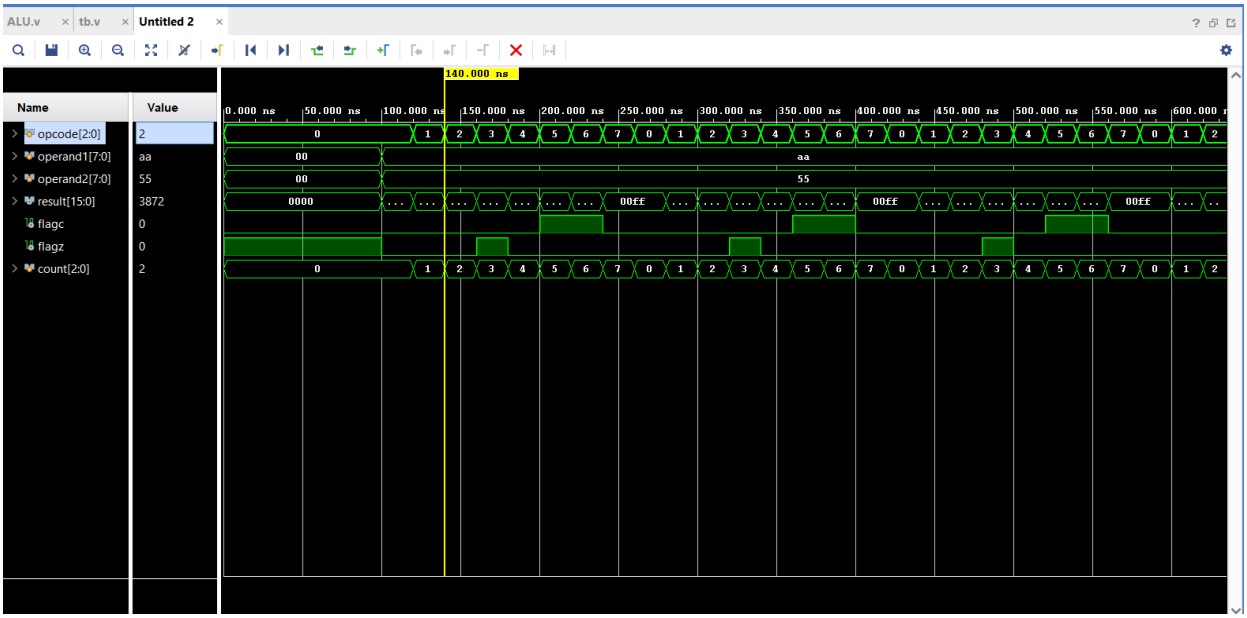



Fig-4: Simulation results when opcode is 2

If we consider the same operands i.e operand1 as aa and operand2 as 55, choosing opcode as 2 performs multiplication. Firstly, both the operands are converted into binary numbers as 10101010 and 01010101 and performs multiplication and gives end result as 0011100001110010 which is equivalent to 3872. Since MSB bit is not 1 carry flag will not be set and as result obtained is not equal to zero, zero flag will not be set.

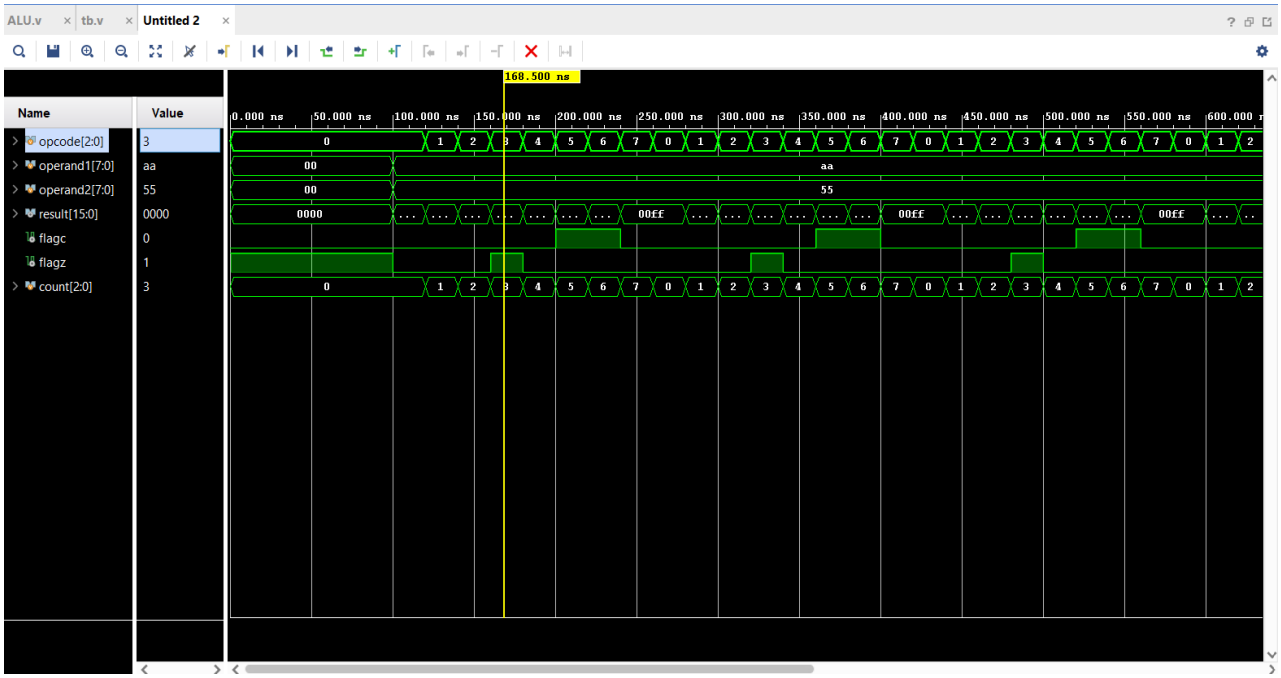



Fig-5: Simulation results when opcode is 3

Fig-5 shows that when opcode is chosen as 3 it performs bitwise AND operation. The result of the AND operation would be 1 when both the bits are high (1) otherwise result would be zero. In this case AND operation between operand1(aa) and operand2(55) gives 0. Carry flag will not be set as MSB bit is not high and zero flag will be set as end result is equal to zero.

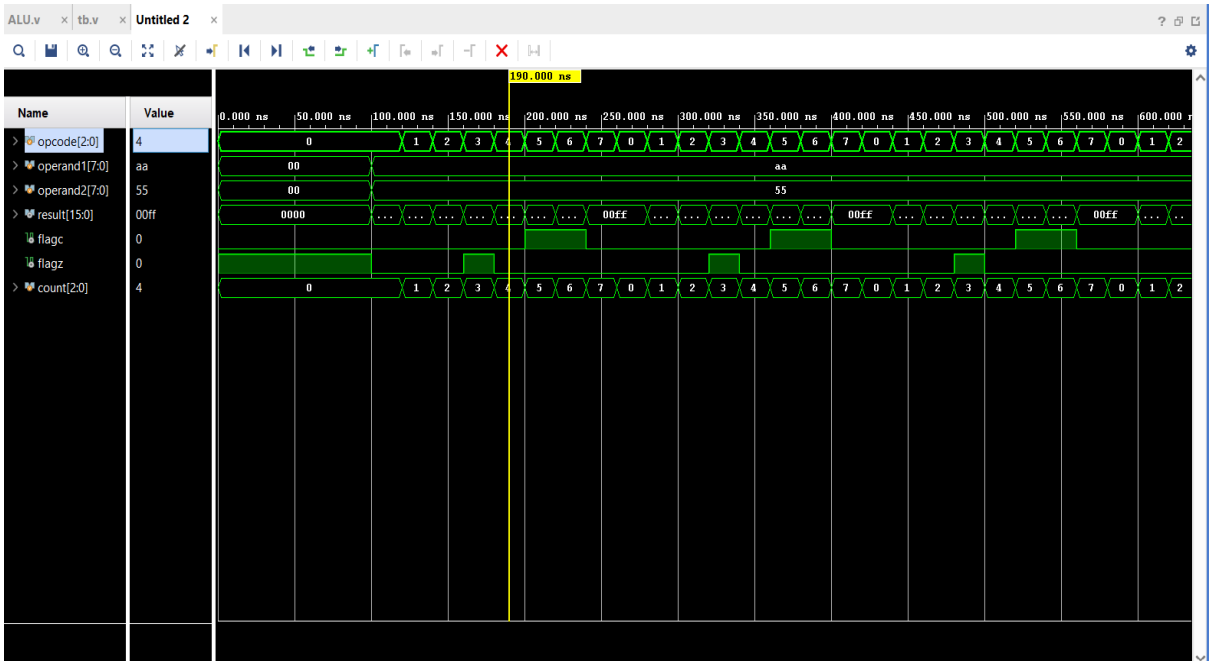



Fig-6: Simulation results when opcode is 4

OR operation between two operands are performed when opcode is 4. The result of the OR gate is high if any one bit is high. If two bits are zero then only the result would be zero. As operand1 is aa and operand two is 55 OR operation between this two gives 00ff (0000000011111111) as result. Both carry flag and zero flag will not be set as MSB bit is zero and total result is not equal to zero.

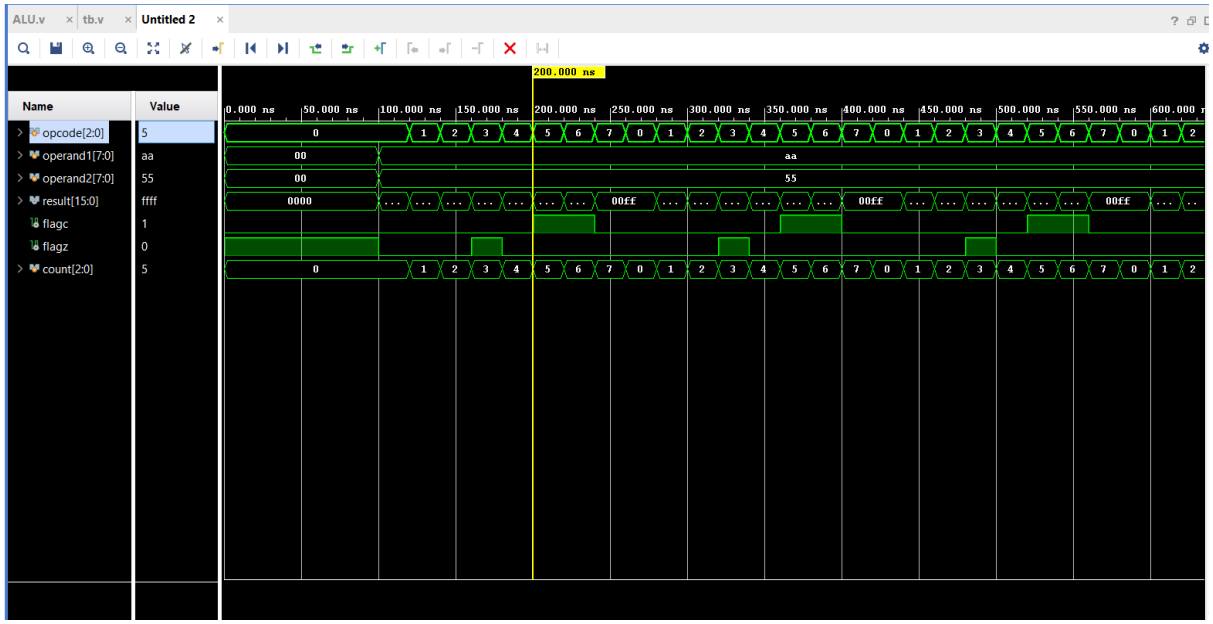



Fig-7: Simulation results when opcode is 5

Fig-7 represents that selecting opcode as 5 performs NAND operation which is negation of AND operation. When two bits are low(0) then the result of a NAND will be high(1) else result will be low(0). In this case NAND operation between aa and 55 gives ffff (1111111111111111). As MSB bit is 1, carry flag will be set but zero flag will not be set as end result is not equal to zero.

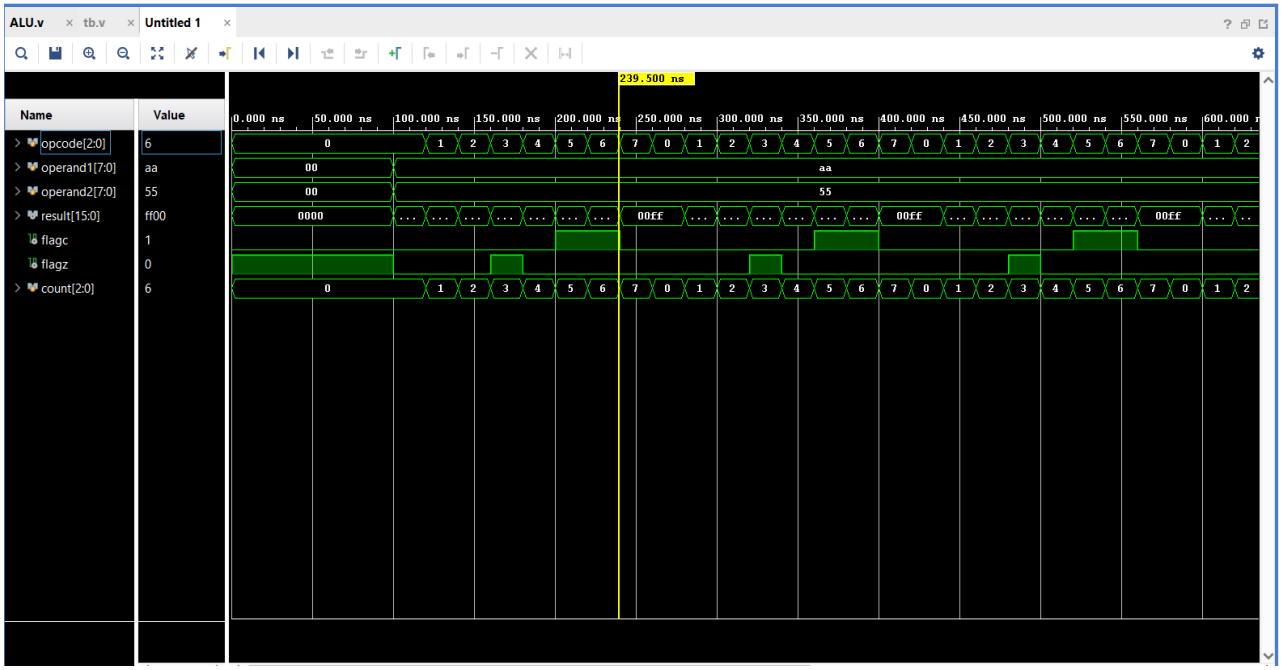



Fig-8: Simulation results when opcode is 6

As NAND is negation of AND similarly NOR operation is negation of OR operation. The NOR operation is performed when opcode is selected as 6. The result obtained when opcode is 4 as 00ff (0000000011111111) negation of this gives ff00(1111111100000000).

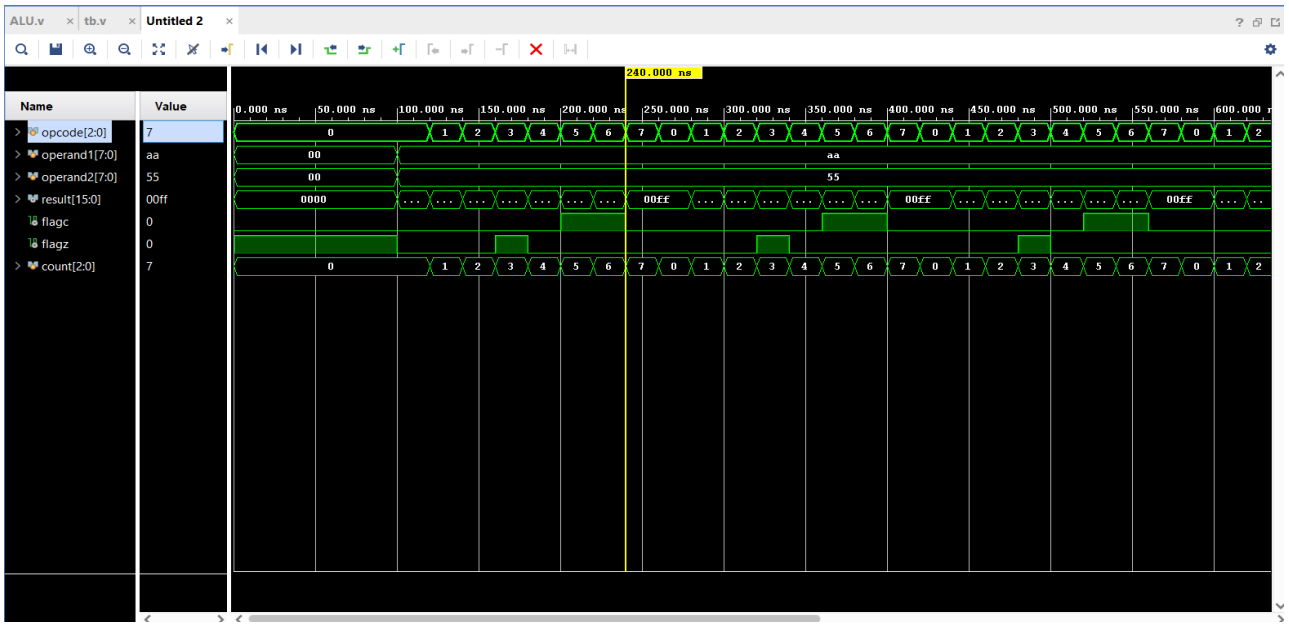



Fig-9: Simulation results when opcode is 7

Fig-9 denotes that when the opcode is 7 then it performs XOR(Exclusive-OR) operation. The result of a XOR operation is 0 when both the bits are same (00,11) otherwise result would be high (01,10). Here operand1 is aa and operand 2 is 55, XOR between this two gives 00ff.

## 7. Power Utilization

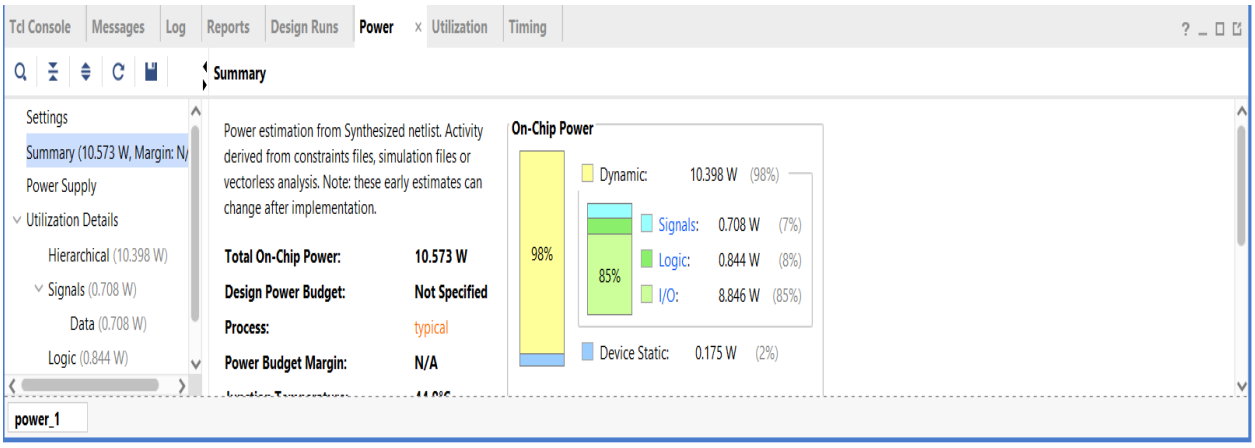



Fig-10: Power Utilization Report

Fig-10 represents that the total power consumed by both dynamic and static components are 10.573W. Most of the power is dynamic (98%). Dynamic power arises due to switching activity as well as due to charging and discharging nature of a capacitor in chip. Static power arises when the device is in idle state.

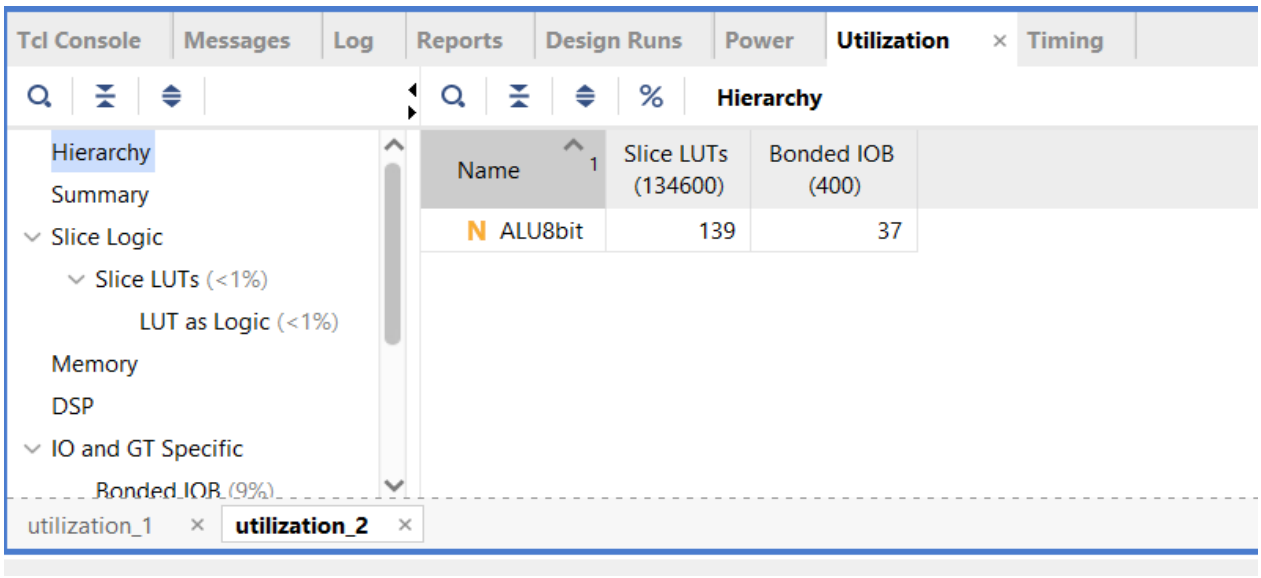



Fig-11: Resource Utilization Report

Fig-11 represents the resource utilization report. In this ALU design 139 slice LUTs are used and 37 bonded IOB are utilized.