1-BIT ALU

1-bit ALU. Build a 1-bit ALU that can perform the following logical and arithmetic operations: AND, OR, NAND, NOR, ADD, SUBTRACT. Include two additional flags: (a) a flag to indicate if the result of the ALU operation is zero (Z), (b) a flag to indicate if the first input is greater than the second input.

OP Codes:

4-bits
bit 1 : neg_A
bit 2 : neg_B
bit 3,4 : indicate component
00 => and
01 => or
10 => add

The main module accepts 4 inputs from the user. The input 1, input 2, carry_in and the opcode. The result is calculated using the following truth tables.

Adder

Input bit for number A	Input bit for number B	Carry bit input C_{IN}	Sum bit output S	Carry bit output C_{OUT}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Subtractor

Inputs			Outputs	
A	B	B_{in}	D	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Table 3.9 Truth table for full-subtractor

The two additional flags which check if the output is 0 or not, and to check if the 1 st input is greater or not are:

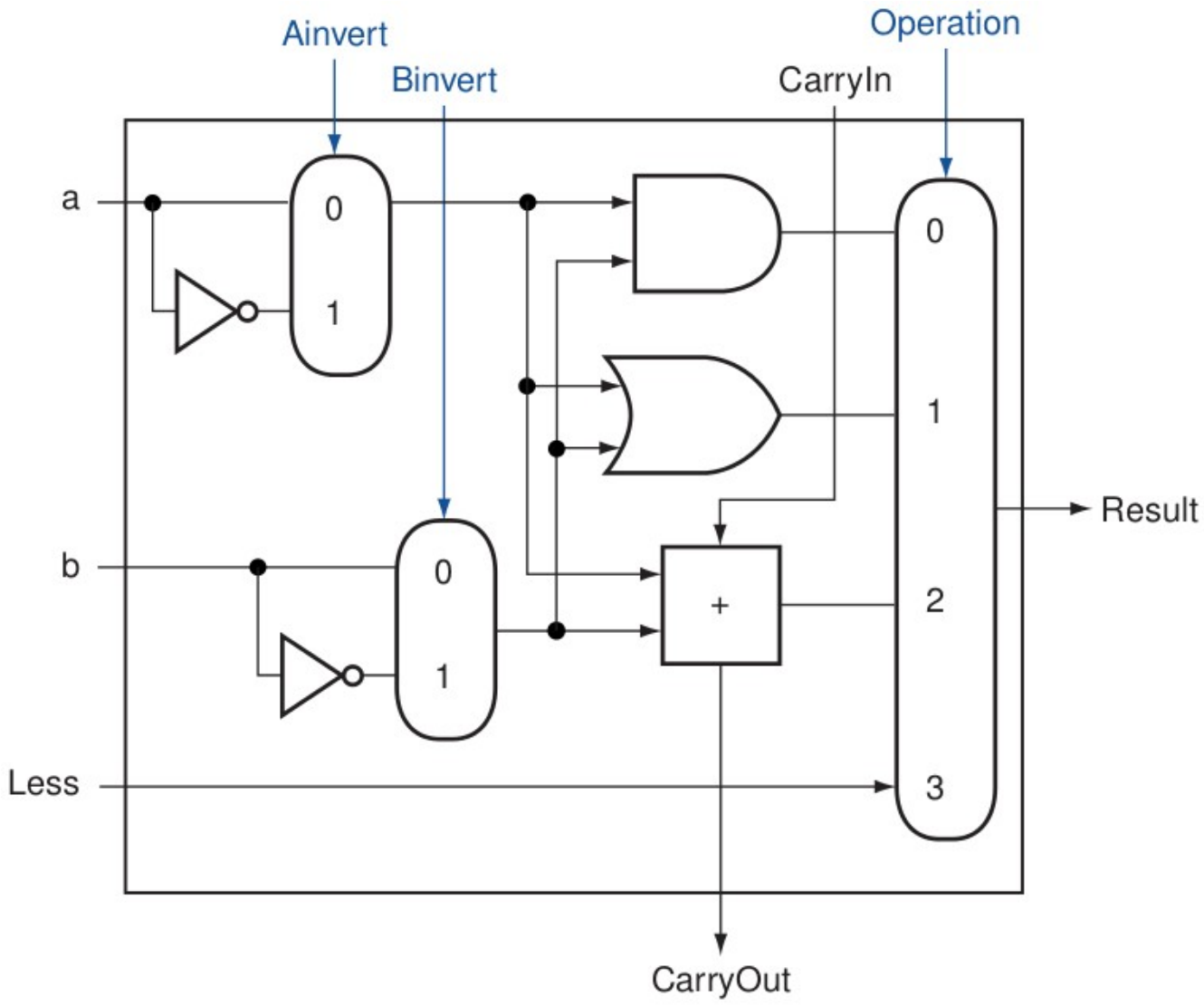
FLAG 1

flag1= not(output_bit)

FLAG 2

flag2= (input_1) and (not(input_2))

BLOCK DIAGRAM



1-Bit ALU

VERILOG CODE :

```
File  Edit  View  Search  Terminal  Help
hackspot@hackspot-inspiron-3521:~/code/Verilog/A1$ bat 1-bit_ALU.v
       File: 1-bit_ALU.v
   1   /*
   2           K Rahul Reddy   17CO119
   3           Sushruth V      17CO148
   4           24 October 2018
   5
   6           Op-Code : 4-bits
   7           bit 1 : neg_A
   8           bit 2 : neg_B
   9           bit 3,4 : indicate component
  10                                   00 => and
  11                                   01 => or
  12                                   10 => add
  13
  14   */
  15
  16   module one_bit_ALU(output reg Result, output reg Zero, output reg CarryOut, input A, input B, input CarryIn, input[3:0] Op);
  17           reg a;
  18           reg b;
  19           always@(A, B, CarryIn, Op) begin
  20                   a = (~A & Op[3]) | (A & ~Op[3]);
  21                   b = (~B & Op[2]) | (B & ~Op[2]);
  22                   if (Op[1:0] == 2'b00) begin
  23                           Result = a & b;
  24                   end
  25                   if (Op[1:0] == 2'b01) begin
  26                           Result = a | b;
  27                   end
  28                   if (Op[1:0] == 2'b10) begin
  29                           {CarryOut, Result} = Op[2] + a + b;
  30                   end
  31                   Zero = ~Result;
  32           end
  33   endmodule
hackspot@hackspot-inspiron-3521:~/code/Verilog/A1$
```

SAMPLE OUTPUT :

```
File  Edit  View  Search  Terminal  Help
hackspot@hackspot-inspiron-3521:~/code/Verilog/A1$ vvp 1-bit_ALU.vvp
VCD info: dumpfile one_bit_ALU.vcd opened for output.

Time =    1
A =   0         B =   1
Op Code = 0000          Carry In =   0

Result =   0            Zero =   1              Carry Out = x


Time =    2
A =   1         B =   0
Op Code = 0000          Carry In =   0

Result =   0            Zero =   1              Carry Out = x


Time =    3
A =   1         B =   1
Op Code = 0000          Carry In =   0

Result =   0            Zero =   1              Carry Out = x


Time =    4
A =   0         B =   0
Op Code = 0000          Carry In =   1

Result =   1            Zero =   0              Carry Out = x


Time =    5
A =   0         B =   1
Op Code = 0000          Carry In =   1

Result =   0            Zero =   1              Carry Out = x


Time =    6
A =   1         B =   0
Op Code = 0000          Carry In =   1
```