Verilog

```
//=====================================================
// Title       : Basic ALU
// Description : Performs ADD, SUB, AND, OR, AND NOT
//=====================================================

module ALU (
    input  [7:0] A,           // Operand A
    input  [7:0] B,           // Operand B
    input  [2:0] OP,          // Operation selector
    output reg [7:0] RESULT, // Result
    output reg ZERO,          // Zero flag
    output reg CARRY,         // Carry flag
    output reg NEGATIVE       // Negative flag
);

    parameter ADD    = 3'b000;
    parameter SUB    = 3'b001;
    parameter AND_OP = 3'b010;
    parameter OR_OP  = 3'b011;
    parameter ANDNOT = 3'b100;

    reg [8:0] temp; // for carry

    always @(*) begin
        case (OP)
            ADD: begin
                temp = A + B;
                RESULT = temp[7:0];
                CARRY = temp[8];
            end
            SUB: begin
                temp = A - B;
                RESULT = temp[7:0];
                CARRY = temp[8];
            end
            AND_OP: begin
                RESULT = A & B;
                CARRY = 0;
            end
            OR_OP: begin
                RESULT = A | B;
                CARRY = 0;
            end
```

```
            ANDNOT: begin
                RESULT = A & (~B);
                CARRY = 0;
            end
            default: begin
                RESULT = 8'b00000000;
                CARRY = 0;
            end
        endcase

        ZERO = (RESULT == 8'b00000000);
        NEGATIVE = RESULT[7];
    end

endmodule
```

Verilog

```
//=====================================================
// Testbench for ALU
//=====================================================

module tb_ALU;

    reg [7:0] A, B;
    reg [2:0] OP;
    wire [7:0] RESULT;
    wire ZERO, CARRY, NEGATIVE;

    // Instantiate ALU
    ALU uut (
        .A(A), .B(B), .OP(OP),
        .RESULT(RESULT), .ZERO(ZERO), .CARRY(CARRY), .NEGA
TIVE(NEGATIVE)
    );

    initial begin

$display("-----------------------------------------------
-------------");
        $display("TIME |  A   |   B   | OP | RESULT | ZERO
| CARRY | NEGATIVE ");

$display("-----------------------------------------------
-------------");

        $monitor("%4t | %b | %b | %b | %b |   %b   |   %b
|    %b",
                 $time, A, B, OP, RESULT, ZERO, CARRY,
NEGATIVE);

        // Test 1: ADD
        A = 8'b00001100; B = 8'b00000011; OP = 3'b000;
#10;
        // Test 2: SUB
        A = 8'b00001100; B = 8'b00000010; OP = 3'b001;
#10;
        // Test 3: AND
        A = 8'b10101010; B = 8'b11001100; OP = 3'b010;
#10;
        // Test 4: OR
        A = 8'b10101010; B = 8'b01010101; OP = 3'b011;
#10;
        // Test 5: AND NOT
        A = 8'b11110000; B = 8'b00001111; OP = 3'b100;
```

```
#10;
        // Test 6: ZERO FLAG
        A = 8'b00000000; B = 8'b00000000; OP = 3'b010;
#10;

        $finish;
    end

endmodule
```

```
-------------------------------------------------------------------
---
TIME |  A    |    B    | OP | RESULT | ZERO | CARRY |
NEGATIVE
-------------------------------------------------------------------
---
   0 | 00001100 | 00000011 | 000 | 00001111 |   0    |   0
|     0    <-- 12 + 3 = 15
  10 | 00001100 | 00000010 | 001 | 00001010 |   0    |   1
|     0    <-- 12 - 2 = 10
  20 | 10101010 | 11001100 | 010 | 10001000 |   0    |   0
|     1    <-- AND
  30 | 10101010 | 01010101 | 011 | 11111111 |   0    |   0
|     1    <-- OR
  40 | 11110000 | 00001111 | 100 | 11110000 |   0    |   0
|     1    <-- AND NOT
  50 | 00000000 | 00000000 | 010 | 00000000 |   1    |   0
|     0    <-- ZERO FLAG
```

Operation	A	B	Result	Explanation
ADD	12	3	15	Simple additio
SUB	12	2	10	Subtraction
AND	10101010	11001100	10001000	Bitwise AND
OR	10101010	01010101	11111111	Bitwise OR
AND NOT	11110000	00001111	11110000	A AND (NOT B)
ZERO	00000000	00000000	00000000	Result = 0, ZER

Verilog

```
//=============================================================
// Title       : Basic ALU
// Description : Performs ADD, SUB, AND, OR, AND NOT
//=============================================================

module ALU (
    input  [7:0] A,           // Operand A
    input  [7:0] B,           // Operand B
    input  [2:0] OP,          // Operation selector
    output reg [7:0] RESULT,  // Output result
    output reg ZERO,          // Zero flag
    output reg CARRY,         // Carry flag
    output reg NEGATIVE       // Negative flag
);

    // Operation codes
    parameter ADD    = 3'b000;
    parameter SUB    = 3'b001;
    parameter AND_OP = 3'b010;
    parameter OR_OP  = 3'b011;
    parameter ANDNOT = 3'b100;

    reg [8:0] temp; // extra bit for carry

    always @(*) begin
        case (OP)
            ADD: begin
                temp = A + B;
                RESULT = temp[7:0];
                CARRY = temp[8];
            end
            SUB: begin
                temp = A - B;
                RESULT = temp[7:0];
                CARRY = temp[8];
            end
            AND_OP: begin
                RESULT = A & B;
                CARRY = 0;
            end
            OR_OP: begin
                RESULT = A | B;
                CARRY = 0;
            end
```

```
            ANDNOT: begin
                RESULT = A & (~B);
                CARRY = 0;
            end
            default: begin
                RESULT = 8'b00000000;
                CARRY = 0;
            end
        endcase

        // Flags
        ZERO = (RESULT == 8'b00000000);
        NEGATIVE = RESULT[7];
    end
endmodule
```

Verilog

```
//=======================================================
// Testbench for Basic ALU
//=======================================================
module tb_ALU;

    reg [7:0] A, B;
    reg [2:0] OP;
    wire [7:0] RESULT;
    wire ZERO, CARRY, NEGATIVE;

    // Instantiate the ALU
    ALU uut (
        .A(A), .B(B), .OP(OP),
        .RESULT(RESULT), .ZERO(ZERO),
        .CARRY(CARRY), .NEGATIVE(NEGATIVE)
    );

    initial begin

$display("-------------------------------------------------
-------------");
        $display("TIME |   A   |   B   | OP |  RESULT  |
ZERO | CARRY | NEGATIVE");

$display("-------------------------------------------------
-------------");
        $monitor("%4t | %b | %b | %b | %b |   %b   |   %b
|    %b",
                 $time, A, B, OP, RESULT, ZERO, CARRY,
NEGATIVE);

        // Test 1: ADD
        A = 8'b00001100; B = 8'b00000011; OP = 3'b000;
#10;
        // Test 2: SUB
        A = 8'b00001100; B = 8'b00000010; OP = 3'b001;
#10;
        // Test 3: AND
        A = 8'b10101010; B = 8'b11001100; OP = 3'b010;
#10;
        // Test 4: OR
        A = 8'b10101010; B = 8'b01010101; OP = 3'b011;
#10;
        // Test 5: AND NOT
        A = 8'b11110000; B = 8'b00001111; OP = 3'b100;
#10;
        // Test 6: ZERO flag
        A = 8'b00000000; B = 8'b00000000; OP = 3'b010;
#10;

        $finish;
    end
endmodule
```

```
------------------------------------------------------------------
---
TIME |   A    |   B    | OP |  RESULT  | ZERO | CARRY |
NEGATIVE
------------------------------------------------------------------
---
   0 | 00001100 | 00000011 | 000 | 00001111 |   0   |   0
|     0   <-- ADD 12 + 3 = 15
  10 | 00001100 | 00000010 | 001 | 00001010 |   0   |   1
|     0   <-- SUB 12 - 2 = 10
  20 | 10101010 | 11001100 | 010 | 10001000 |   0   |   0
|     1   <-- AND
  30 | 10101010 | 01010101 | 011 | 11111111 |   0   |   0
|     1   <-- OR
  40 | 11110000 | 00001111 | 100 | 11110000 |   0   |   0
|     1   <-- AND NOT
  50 | 00000000 | 00000000 | 010 | 00000000 |   1   |   0
|     0   <-- ZERO flag set
```

Operation	OP Code	Result	Explanation
ADD	000	00001111	12 + 3 = 15
SUB	001	00001010	12 - 2 = 10
AND	010	10001000	Bitwise AND
OR	011	11111111	Bitwise OR
AND NOT	100	11110000	A AND (NOT B)
ZERO	-	00000000	Both inputs 0 → ZERO = 1