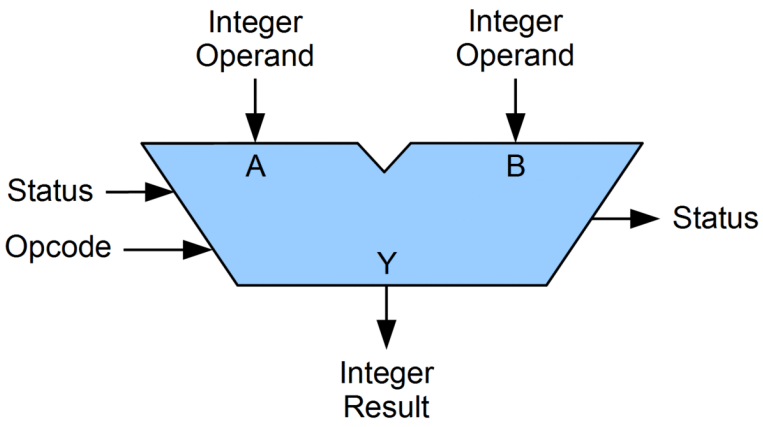
8 Bits Arithmetic Logic Unit



Introduction:

The Arithmetic Logic Unit (ALU) is a vital component of digital computers used to perform arithmetic and logic operations on binary numbers. In this report, we will discuss the design and implementation of an 8-bit ALU using Verilog HDL.

Design:

The 8-bit ALU can be designed by breaking it down into three modules: full-adder, full-adder-subtractor, and ALU. The full-adder module takes three inputs, performs addition, and produces two outputs: the sum and the carry-out.

The full-adder-subtractor module takes two 8-bit inputs, a carry-in bit, and produces an 8-bit output, a carry-out bit, and a c_flag. The c_flag indicates whether there is a carry-out from the most significant bit, which can occur during subtraction. This module uses eight instances of the full-adder module to perform the addition or subtraction.

The ALU module takes two 8-bit inputs, an opcode, a carry-in bit, and produces an 8-bit output, a carry-out bit, a zero flag, and a c_flag. The opcode specifies the operation to be performed, such as addition, subtraction, AND, OR, XOR, shift-left A or B. The zero flag indicates if the result is zero, while the c_flag indicates if there is a carry-out from the most significant bit.

Implementation:

The Verilog HDL code for the full-adder, full-adder-subtractor, and ALU modules are shown below:

- Full Adder Module:

```
/* 1 Bit Full Adder*/
module full_adder(a,b,cin,sum,cout);
input a,b,cin;
output sum, cout;

assign sum = a ^ b ^ cin;
assign cout = (a & b) | (a & cin) | (b & cin);
endmodule
```

- Full Adder-Subtractor Module:

```
/* 8 Bits Full Adder/Subtractor */
module full_adder_subtractor(a, b, carry_in, result, carry_out, flag_c);
input [7:0] a,b;
input carry_in;
output [7:0] result;
output carry_out, flag_c;

wire [7:0] carry;

full_adder fa0(a[0], b[0] ^ carry_in, carry_in, result[0], carry[0]);
full_adder fa1(a[1], b[1] ^ carry_in, carry[0], result[1], carry[1]);
full_adder fa2(a[2], b[2] ^ carry_in, carry[1], result[2], carry[2]);
full_adder fa3(a[3], b[3] ^ carry_in, carry[2], result[3], carry[3]);
full_adder fa4(a[4], b[4] ^ carry_in, carry[3], result[4], carry[4]);
full_adder fa5(a[5], b[5] ^ carry_in, carry[4], result[5], carry[5]);
full_adder fa6(a[6], b[6] ^ carry_in, carry[5], result[6], carry[6]);
full_adder fa7(a[7], b[7] ^ carry_in, carry[6], result[7], carry_out);

// generate the c_flag output by comparing A and B
assign flag_c = (a >= b) ? 1'b1 : 1'b0;
endmodule
```

- ALU Module:

```
/* 8 Bits Arithmetic Logic Unit */
module ALU(a, b, opcode, carry_in, result, carry_out, zero_flag, c_flag);
    input [7:0] a;
    input [7:0] b;
    input [2:0] opcode;
    input carry_in;
    output reg [7:0] result;
    output wire carry_out;
    output reg zero_flag;
    output wire c_flag;

    wire [7:0] sum;// Based on carry_in (if 1 --->difference) and (if 0---> addition)
    wire [7:0] and_res;
    wire [7:0] or_res;
    wire [7:0] xor_res;
    wire [7:0] shift_left_a;
    wire [7:0] shift_left_b;


    full_adder_subtractor add_sub(a, b, carry_in, sum, carry_out, c_flag);
    // generate AND, OR, XOR, and shift-left outputs using intermediate wires
    assign and_res = a & b;
    assign or_res = a | b;
    assign xor_res = a ^ b;
    assign shift_left_a = a << 1;
    assign shift_left_b = b << 1;

    always @(*) begin
        case(opcode)
            3'b000: result = sum;
            3'b001: result = sum;
            3'b010: result = and_res;
            3'b011: result = or_res;
            3'b100: result = xor_res;
            3'b101: result = {8{1'b0}}; // default value
            3'b110: result = shift_left_a;
            3'b111: result = shift_left_b;
        endcase

         zero_flag = (result == 8'h00) ? 1'b1 : 1'b0;
   end
endmodule
```

Conclusion:

The implementation of an 8-bit ALU using Verilog HDL has been presented in this report. The design was broken down into three modules: full-adder, full-adder-subtractor, and ALU. The full-adder module performs addition, while the full-adder-subtractor module performs both addition and subtraction. The ALU module performs various arithmetic and logic operations based on the opcode selected. Verilog HDL code snippets for each module were provided.