Megan McCormick

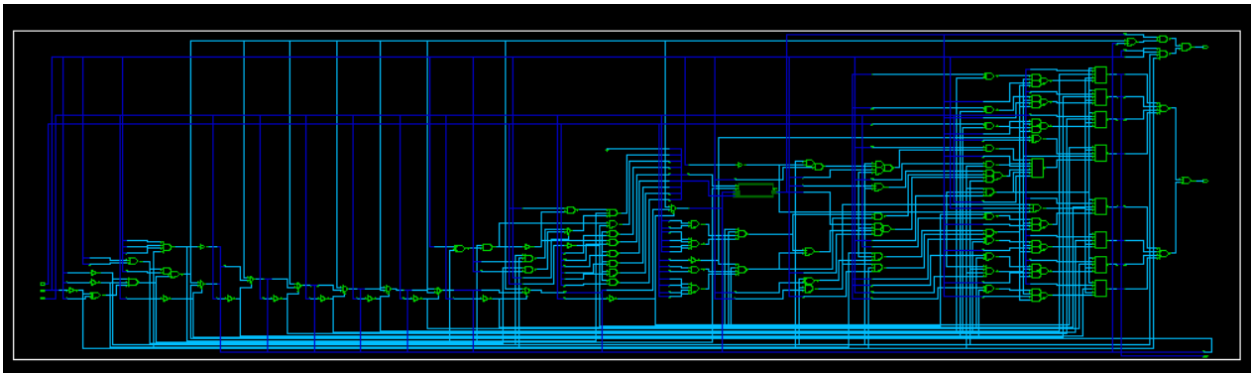
VLSI System Design

Homework 2

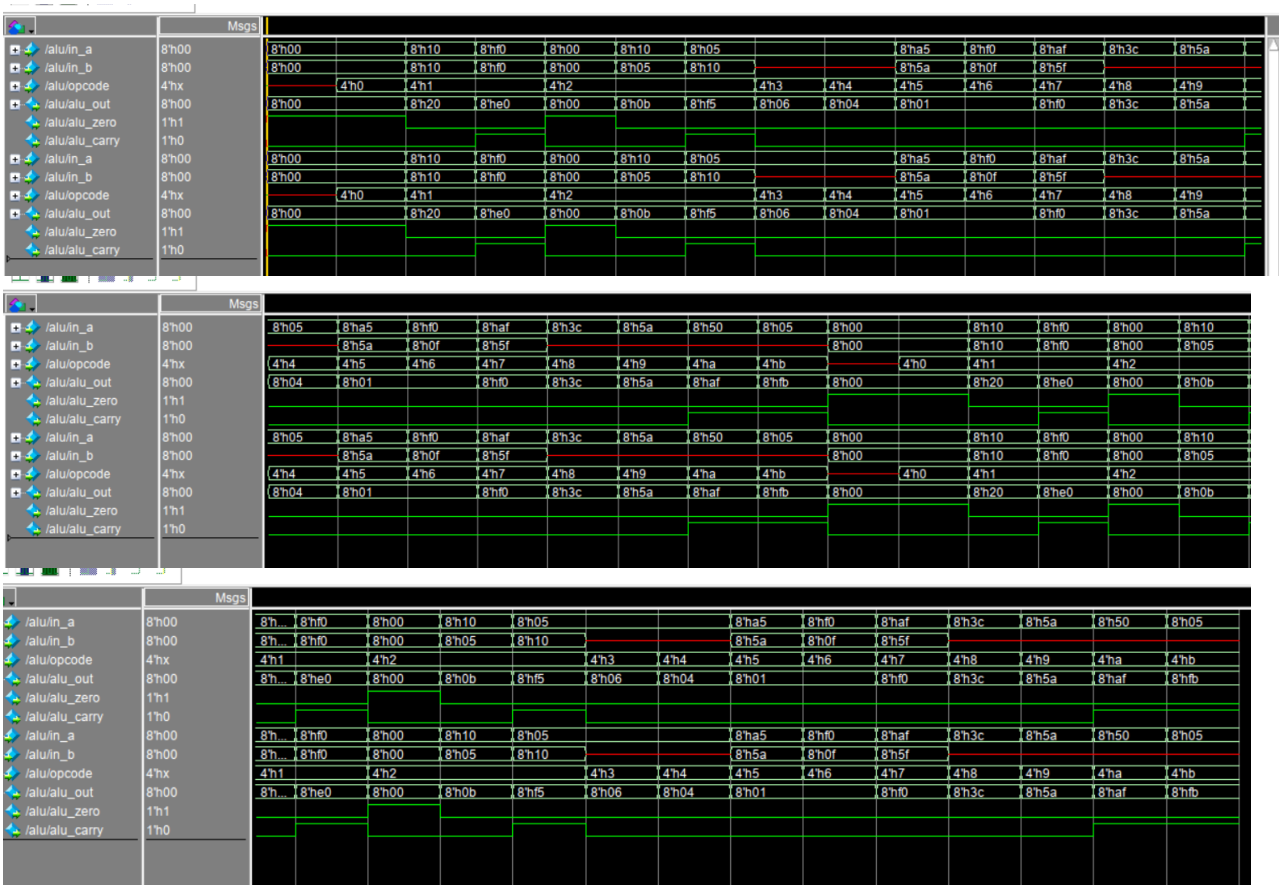
Verilog Code:

```
module alu(
input        [7:0] in_a     ,  //input a
input        [7:0] in_b     ,  //input b
input        [3:0] opcode   ,  //opcode input
output  reg  [7:0] alu_out  ,  //alu output
output  reg        alu_zero ,  //logic '1' when alu_output [7:0] is all zeros
output  reg        alu_carry   //indicates a carry out from ALU
);

  parameter      c_add = 4'h1;                //in_a + in_b
  parameter      c_sub = 4'h2;          //in_a - in_b
  parameter      c_inc = 4'h3;          //in_a + 1
  parameter      c_dec = 4'h4;          //in_a - 1
  parameter      c_or  = 4'h5;          //in_a OR in_b
  parameter      c_and = 4'h6;          //in_a AND in_b
  parameter      c_xor = 4'h7;          //in_a XOR in_b
  parameter      c_shr = 4'h8;          //in_a is shifted one place right, zero shifted in
  parameter      c_shl = 4'h9;          //in_a is shifted one place left, zero shifted in
  parameter      c_onescomp = 4'hA;     //in_a gets "ones complemented"
  parameter      c_twoscomp = 4'hB;     //in_a gets "twos complemented"

  always_comb
  begin
         alu_carry = 0;
         case(opcode)
                c_add:  {alu_carry,alu_out} = in_a + in_b;
                c_sub:  {alu_carry,alu_out} = in_a - in_b;
                c_inc:  {alu_carry,alu_out} = in_a + 1;
                c_dec:  {alu_carry,alu_out} = in_a -1;
                c_or:   alu_out = (in_a || in_b);
                c_and:  alu_out = in_a && in_b;
                c_xor:  alu_out = in_a ^ in_b;
                c_shr:  alu_out = in_a >> 0;
                c_shl:  {alu_carry,alu_out} = in_a << 0;
                c_onescomp:    {alu_carry,alu_out} = ~in_a;
                c_twoscomp:    {alu_carry,alu_out} = ~in_a + 1;
                default:       alu_out = 0;
         endcase
         alu_zero = ~| alu_out; //Check that alu_out is all 0s
  end
endmodule
```

Schematic:



Waveform:



Questions:

a. 1309.048222
b. 24
c. 1249.434007/5.5296um^2
d. It's an add/subtract block. It's the part of the alu that does addition or subtraction(which is addition with a negative)
e. Maximum delay path was from r30/B[0] (alu_DW01_addsub_0) to r30/SUM[7] (alu_DW01_addsub_0)