Name _____

Computer Organization
EE 3755
Midterm Examination

Wednesday, 30 October 2013, 8:30–9:20 CDT

Problem 1	_____	(21 pts)
Problem 2	_____	(15 pts)
Problem 3	_____	(25 pts)
Problem 4	_____	(11 pts)
Problem 5	_____	(11 pts)
Problem 6	_____	(11 pts)
Problem 7	_____	(6 pts)
Exam Total	_____	(100 pts)

Alias _____

Good Luck!

Problem 1: [21 pts] Consider the module below:

```
module prob1is(x,y,a,b,c);
   input a, b, c;   output x, y;
   wire  a, b, c, x, y;          <-- HINT: Declare types for parts b and c.
   assign x = !a ^ y;
   assign y =  b || c;
endmodule
```

(*a*) Draw a logic diagram corresponding to the module. Don't optimize.

☐ Logic diagram of `prob1is`.

(*b*) Complete the module below so that it performs the same operation as `prob1is` but is in explicit structural form.

☐ Explicit structural form of `prob1is`. ☐ Don't forget to declare types.

```
module prob1es(x,y,a,b,c);
   input a, b, c;    output x, y;
```

```
endmodule
```

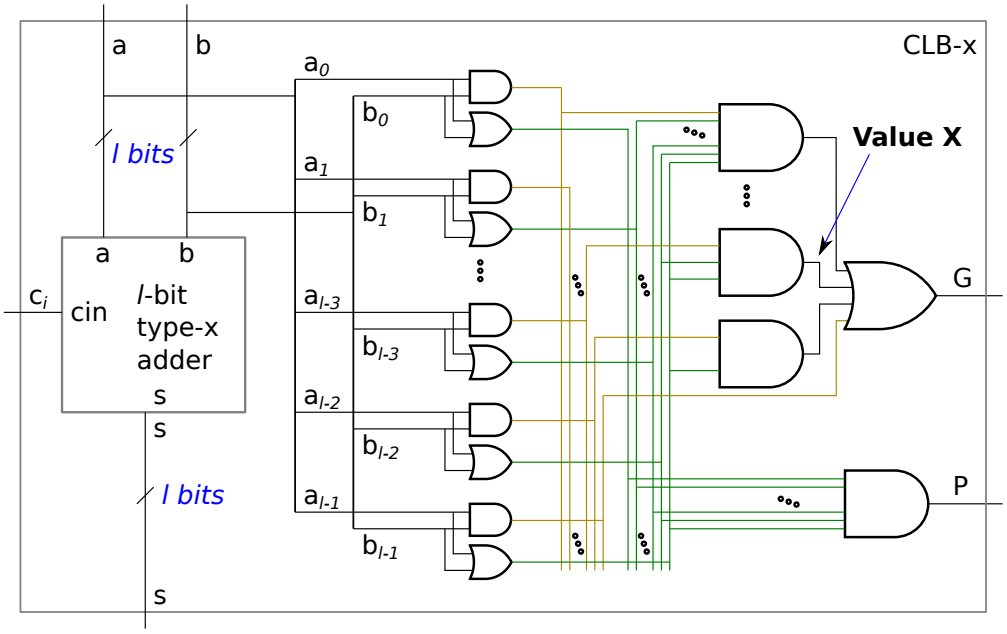
(*c*) Complete the module below so that it performs the same operation as `prob1is` but is in synthesizable behavioral form.

☐ Synthesizable behavioral form of `prob1is`. ☐ Don't forget to declare types.

```
module prob1sb(x,y,a,b,c);
   input a, b, c;    output x, y;
```

```
endmodule
```

Problem 2: [15 pts] Appearing below is the generic carry lookahead block from the CLA lecture slides. Remember that it is part of a hierarchical carry lookahead adder.



(*a*) The output of one of the AND gates is labeled *Value X*. Suppose $l = 16$. Find inputs to the module such that Value X is `1` but all of the other inputs to the generate OR gate are `0`.

☐ Value of module inputs needed so that Value X will be `1`.

(*b*) Notice that the module does not have a carry out signal. Add logic to the module above to compute a carry out signal. (See next part.)

☐ Logic for carry out. ☐ Do not add new inputs to the module.

(*c*) The carry out signal from the previous part isn't needed in real life. Why not? What if it were used?

☐ Why doesn't the CLB need a carry out signal?

☐ What would be the problem with using the carry out signal?

Problem 3: [25 pts] A population count module based on Homework 3 Problem 2 appears below.

```
module red_pop(p,a);
   parameter N = 20;
   parameter M = 4;   // Number of parts.
   parameter N_PER_PART = N/M;
   input wire [N-1:0] a;        output reg [8:0] p;
   reg [8:0]    pa;             integer      i, j;

   always @( a ) begin
      for ( j=0; j<M; j=j+1 ) begin
         pa = a[ j*N_PER_PART ];
         for ( i=1; i<N_PER_PART; i=i+1 ) pa = pa + a[ j*N_PER_PART + i ];
         if ( j == 0 ) p = pa; else p = p + pa;
      end
   end
endmodule
```

(*a*) Sketch the logic that would be inferred for this module (the logic that would be synthesized without optimization). Show adders as a box with a + inside.

☐ Show synthesized logic without optimization.

Problem 3, continued: The module below is the same as the one on the previous page.

```
module red_pop(p,a);
   parameter N = 20;
   parameter M = 4;   // Number of parts.
   parameter N_PER_PART = N/M;
   input wire [N-1:0] a;        output reg [8:0] p;
   reg [8:0]    pa;             integer      i, j;

   always @( a ) begin
      for ( j=0; j<M; j=j+1 ) begin
         pa = a[ j*N_PER_PART ];
         for ( i=1; i<N_PER_PART; i=i+1 ) pa = pa + a[ j*N_PER_PART + i ];
         if ( j == 0 ) p = pa; else p = p + pa;
      end
   end
endmodule
```

(*b*) Compute the cost of the logic from the previous part assuming that all adders are ripple adders and that the cost of a BFA is 7 units. The size of the adders has been made as small as possible. Assume that the gates outside of the BFAs have a cost of one unit each.

☐ Cost of circuit. Show work for partial credit.

☐ Account for size of adders, the sizes have been minimized.

(*c*) Suppose the delay of each ripple adder is just 1 time unit. (Yes, just 1 time unit.) Find the delay of the circuit in terms of `N` and `M`. That is, don't assume `N=20` and `M=4`.

☐ Delay in terms of `N` and `M` assuming 1-time-unit ripple adders.

(*d*) In the module above `M` was set to 4 and `N` to 20. Consider the case where `N` is some arbitrary value, and we want to compute a value for `M` that minimizes delay for this `N`. Using the delay model from the previous part, find an expression for `M` in terms of `N` that will minimize delay.

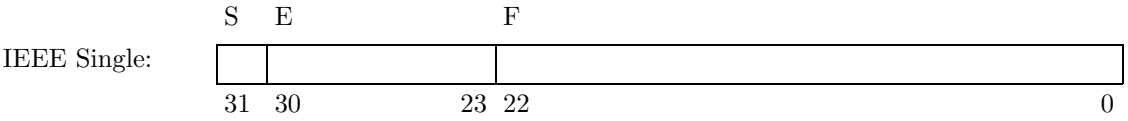
☐ Best value of `M` in terms of `N`.

Problem 4: [11 pts] Answer the computer arithmetic questions below.

(*a*) Show the longhand steps needed to multiply $1234_{16} \times 1203_{16}$ using a radix-16 (four bit) multiplication algorithm, but do not add together the partial products. Show the work in binary or hexadecimal. This is **without** Booth recoding. (The product is $147de9c_{16}$, but remember there is no need to add the partial products.)

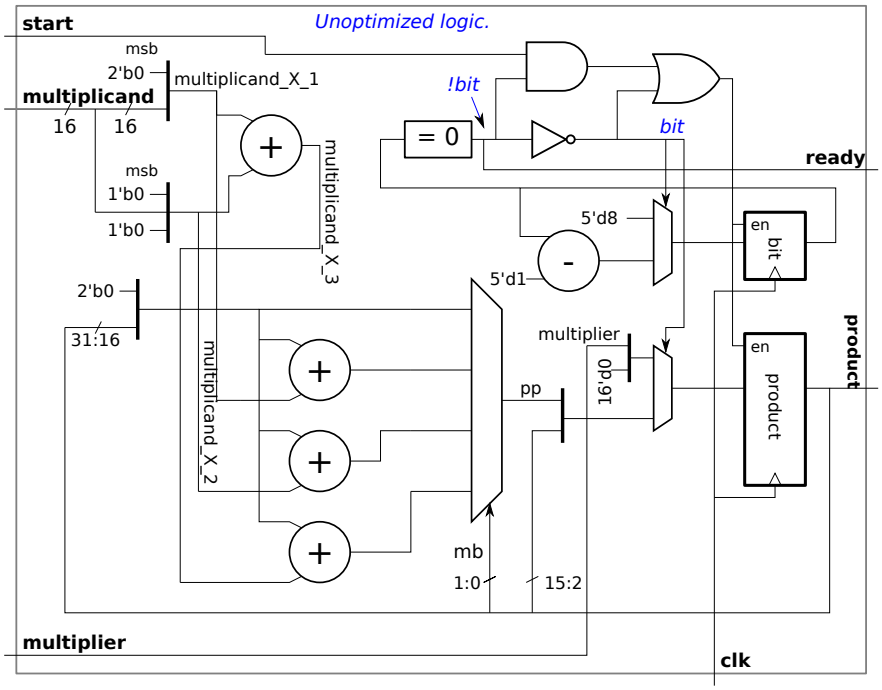
☐ Longhand steps for radix-16 (4 bit) $1234_{16} \times 1203_{16}$

(*b*) Show the value of IEEE 754 single-precision floating point number `0x42fa0000`. For your convenience the layout of an IEEE 754 single is shown below.



☐ Value of number (in decimal or as a formula to compute value) is:

Problem 5: [11 pts] Appearing below is unoptimized logic for an ordinary radix-4 multiplier.



(*a*) As described in class, the logic around the four-input multiplexor can be reduced in cost. Show how.

☐ Sketch of lower-cost logic around the 4-input mux.

(*b*) One of the adders in the illustrated multiplier would not be necessary in a radix-4 multiplier using Booth recoding. Show which adder would not be necessary, and explain what would be done instead.

☐ Indicate which adder not necessary.

☐ Explain what would be done instead.

Problem 6: [11 pts] Answer the following MIPS questions.

(*a*) Show the values in register `$s0` after the execution of each instruction below in the spaces indicated.

☐ Fill in the `s0 =` blanks below.

```
 # Initial register values:  $s1 = 0x18,  $s2 = 0x30

or $s0, $s1, $s2
                    # $s0 =

sll $s0, $s2, 2
                    # $s0 =

slt $s0, $s1, $s2
                    # $s0 =

addi $s0, $s1, 0xffff
                    # $s0 =

ori $s0, $s1, 0xffff
                    # $s0 =
```

(*b*) There is a problem with the instruction below. Describe what the problem is and show replacement instruction(s) that do what was intended.

```
addi $s0, $s0, 0x12345
```

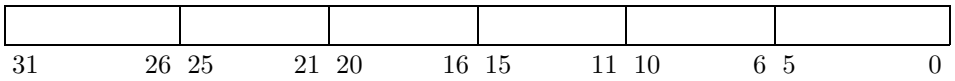
☐ The problem with the instruction is:

☐ Replacement that does the same thing:

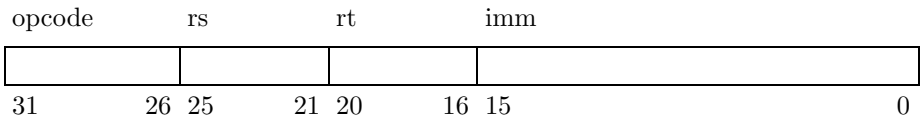
Problem 7: [6 pts] Answer each MIPS encoding question below.

(*a*) Show the encoding of each assembly language instruction below. Some field values would have to be looked up in a table, for those write "look up" instead of the value. For your convenience the layout of the MIPS R and I formats are shown, answers can be written in these or they can be copied.

Format R:



Format I:



☐ Encoding for `sub $20, $21, $22`

☐ Encoding for `lui $8, 0x1234`

☐ Encoding for `sra $9, $2, 31`