Name:_____

EECS150: Components and Design Techniques for Digital Systems

University of California
Dept. of Electrical Engineering and Computer Sciences

Mid Term 2 Fall 2007

Last name: ____Solutions_____ First name_____
Student ID: _____ Login: _____

Lab meeting time: _____ TA's name: _____

You may use a single 8.5x11 sheet of notes.. No calculators! This booklet contains 7 numbered pages, including room to show your work. Please, no extra stray pieces of paper. Put your name on every page. The exam contains 7substantive questions and 100 points, so just over 1 point per minute. Browse through the questions before you start. You have 1.5 hours, so relax, work thoughtfully and give clear answers. Good luck!

I certify that my answers to this exam are my own work. If I am taking this exam early, I certify that I shall not discuss the exam questions, the exam answers, or the content of the exam with anyone until after the scheduled exam time. If I am taking this exam in scheduled time, I certify that I have not discussed the exam with anyone who took it early.

Signature: _____

	Score	Count
Problem 1 [10]		
Problem 2 [10]		
Problem 3 [15]		
Problem 4 [15]		
Problem 5 [15]		
Problem 6 [15]		
Problem 7 [20]		
Total [100]		

Name:_____

Problem 1. (10) Number Representation

For each of the following, what is the value represented by the 8-bit binary pattern 11111000? Also, give the binary representation for its negative.

Number System	*Decimal Value of 11111000*	*Bit representation of its negative*
Two's complement	-8	00001000
Unsigned	248	None
One's complement	-7	00000111
Excess 127	121	00000110
Two's complement fixed point with the binary point in the middle of the 8 bits	-0.5	00001000

Problem 2. (10) Floating Point

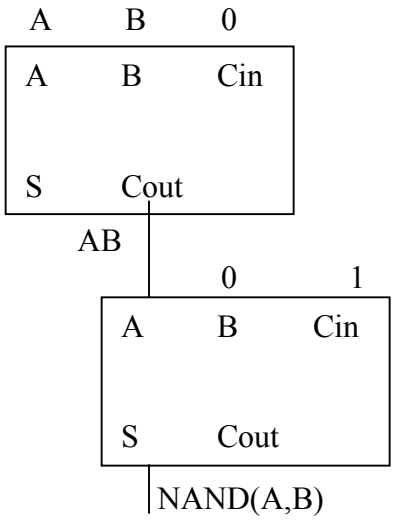
Draw a line matching each occurrence in the left column to the conditions in the right column that best describes when it can occur.

Occurrence	*Conditions*
addition of two normalized floating point numbers results in a DENORM?	Two numbers
multiplication of two normalized floating point numbers results in a DENORM	Two numbers of small magnitude
addition of two normalized floating point numbers result in an overflow	Two numbers of large magnitude
multiplication of two normalized floating point numbers result in an overflow	Two numbers of small magnitude and opposite sign
	Two numbers of large magnitude and opposite sign
	Two numbers of opposite sign and a difference of small magnitude
	Two numbers of opposite sign and a difference of large magnitude

Name:_____

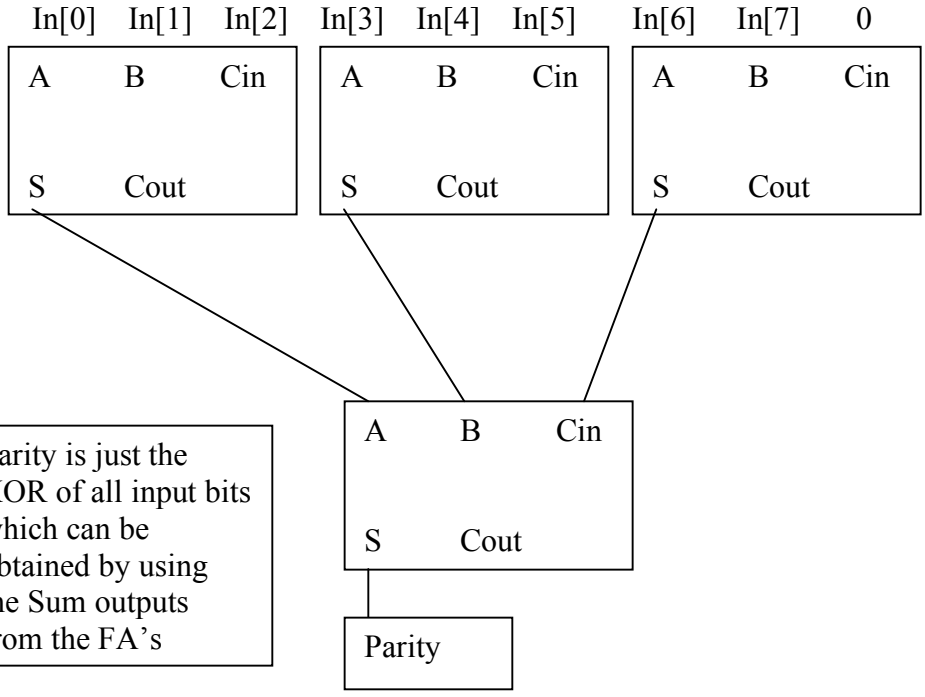
Problem 3. (15) Logic

1. Prove (by drawing a simple diagram) that any logic function can be implemented using full adder cells.



Since all combinational logic can be made out of NAND gates, showing a NAND constructed of FAs is sufficient. Answers that constructed all 3 primitives AND, OR and NOT were also acceptable.

2. How would you build an 8-bit parity circuit using a small number of FAs?



Parity is just the XOR of all input bits which can be obtained by using the Sum outputs from the FA's

Name:_____

Problem 4 (15). Circuit Delays

Determine the maximum clock rate for the circuit shown below. Assume the following:

(1) The primitive inverter delay is 100 ps. All wire delays are 0.

(2) The primitive delay of each gate is the number specified inside the gate. It is in units of primitive inverter delays.

(3) The actual delay of a gate in the circuit is a linear function of its primitive gate delay and its fanout: *Actual delay = primitive delay + 0.25 * (# of fanouts more than 1)*

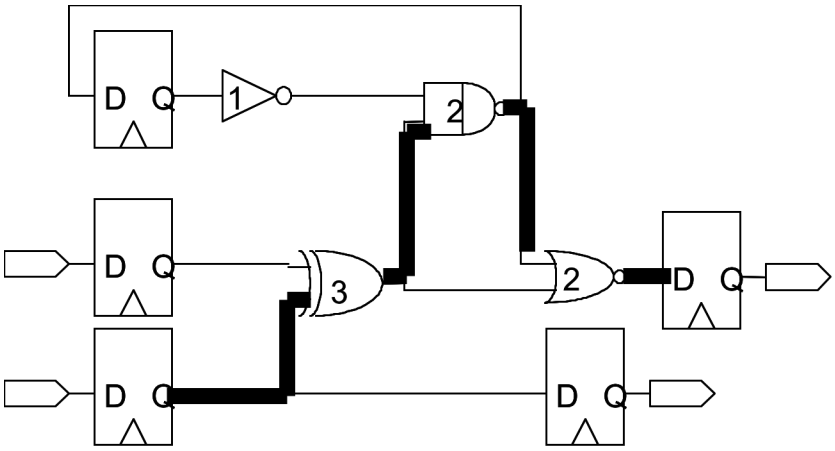
a. Any logic gate, flip-flop or output port counts as one fanout of a gate. For example, the XOR gate in the circuit below has a fanout of 2.

b. Flip-flop outputs incur fanout-related delays, just like gates.

(4) Flip-flop setup time and clock-to-Q time are each 2 inverter delays.

(5) The maximum skew between any two clock inputs is 100 ps.

State clearly any other assumptions you make. **Show your work.**



The critical path is shown on the diagram above

Min Period $= \tau_{\text{Clk-to-Q}} + \tau_{\text{FF-fanout}} + \tau_{\text{XOR}} + \tau_{\text{NAND}} + \tau_{\text{NOR}} + \tau_{\text{Setup}} + \tau_{\text{max-skew}} =$

$= 200\text{ps} + 25\text{ps} + (3 + 0.25(2-1))*100\text{ps} +$
$(2 + 0.25(2-1))*100\text{ps} + 2*100\text{ps} + 200\text{ps} + 100\text{ps}$

$= 225\text{ps} + 325\text{ps} + 225\text{ps} + 200\text{ps} + 200\text{ps} + 100\text{ps} = 1275\text{ps}$

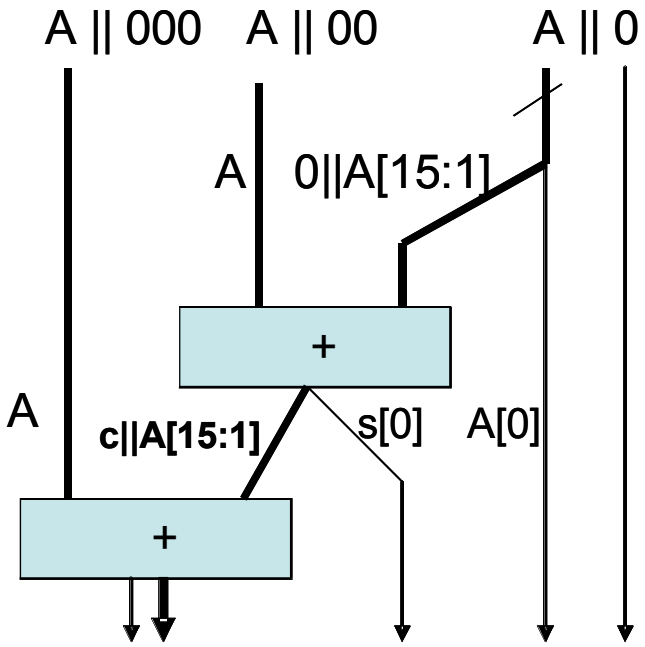
max frequency = 1/min period = 784 MHz

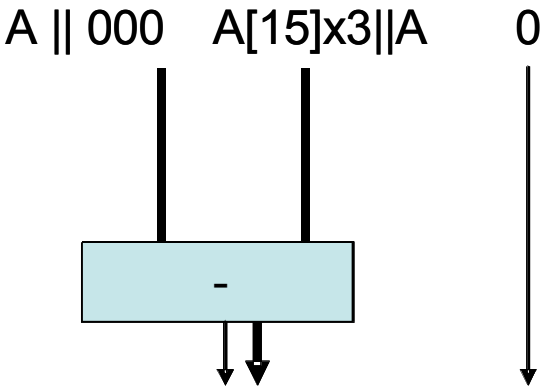
Name:_____

Problem 5 (15). Often in hardware design it is possible to produce optimized circuits for special cases that are faster or more compact than that for the general case.

1. Construct a compact, low delay circuit for a 16-bit "times 14" using adders and/or carry-save adders.
2. Show how to improve your circuit by using one or more subtractor.

1. **Result = A*8 + A*4 + A*2 = A||000 + A||00 + A||0**
2. **Result = A*16 – A*2 = A||0000 – sx(3)||A||0**





Name:_____

Problem 6. (15). Memory

Describe each of the steps involved in a DRAM write operation. (We are looking for a single brief sentence or two for each major step.)

- Prior to the write, `OE_l` and `WE_l` are disasserted (`OE_l = 1, WE_l = 1`) and the DRAM chip is essentially inactive.
- Drive Row Address; then assert RAS
- Drive Data; then asset WE_L
- Drive Col address; then assert CAS (for late, reverse these)
- Deassert WE_L, CAS, RAS

Name:_____

Problem 7 (20). Debugging verilog

Find and correct 6 errors in the verilog modules below. Additional errors count for 1 point of extra credit each. The exact function of the FSM is not important – the errors are basic and common.

```
module ShiftRegister(Clock, Reset, SIn, POut);
       input        Clock, SIn, Reset;
       output reg   [7:0]  POut;


       always @ ( posedge Clock )
              if (Reset)
                     POut <= 8'b0;
              POut <=      {POut[6:0], SIn};
endmodule

module FSM(In, Out, OutValid, Clock, Reset);
       input               In, Clock, Reset;
       output reg   [7:0]  Out;
       output       OutValid;

       reg                 CurrentState, NextState;
       reg                 ShiftReset, OutValid;

       ShiftRegister Shifter(Clock, ShiftReset, Out, In,);
       parameter    STATE_Idle =              1'b0,
                    STATE_A =                       1'b1,

       always @ ( posedge Clock ) begin
              if (Reset)   CurrentState <=           STATE_Idle;
              else         CurrentState <=           NextState;
       end

       always @ ( CurrentState, Sin, Out ) begin
              case ( CurrentState )
                     STATE_Idle: begin
                            if ( In ) begin
                                   NextState =         STATE_A;
                                   ShiftReset = 1'b1;
                            end
                            else begin
                                   NextState =  CurrentState;
                                   ShiftReset = 1'b0;
                            end
                            OutValid =          1'b0;
                     end
                     STATE_A: begin
                            if ( ~Out[7]) begin
                                   NextState =         STATE_A;
                                   OutValid =          1'b1;
                            end
                        else begin
                                   NextState =         STATE_Idle;
                                   OutValid =          1'b0;
                        End
                            ShiftReset = 1'b0;
                     end
              endcase
       end
endmodule
```