Name:____
Perm #:____

Lab Section TA:________________________________

ECE 152A-Winter 2009

Prof. Volkan Rodoplu

MIDTERM EXAMINATION

INSTRUCTIONS:

1. READ THIS PAGE THOROUGHLY WHEN YOU RECEIVE IT, BUT DO NOT START TURNING TO THE OTHER PAGES UNTIL YOU ARE INSTRUCTED TO DO SO.

2. WHENEVER INDICATED, YOU MUST WRITE YOUR ANSWERS ON THE ANSWER LINES PROVIDED IN CERTAIN PROBLEMS. NO PARTIAL CREDIT WILL BE GIVEN ON THESE PROBLEMS. (We will check only the answer on that line.)

3. On other problems, PARTIAL CREDIT will be given only to true statements that make progress towards the correct answer. Partial credit may be given to correct reasoning in developing structures such as K-maps and truth tables in which the variables are clearly labeled. NO partial credit will be given for incorrect statements or statements to which no truth value can be assigned (such as a bunch of numbers or algebraic expressions). NO partial credit will be given for statements that use symbols that the problem statement or you have not defined. NO credit will be given for any work that is not clearly labeled with the part and problem number to which this work provides an answer.

4. All the exam rules in the course syllabus apply to this exam.

5. You may remove the staple from the exam pages, if is more convenient. We will provide a stapler at the end of the exam.

STUDENTS MUST LEAVE THIS PAGE BLANK

Problem Number	Points	Out of
1		15
2		25
3		12
4		14
5		12
6		21
7		15
TOTAL		114

PROBLEMS:

DIFFICULTY LEVEL 0

Problem # 1 KARNAUGH MAPS [15 points]

(a) **(4 points)** The Karnaugh map ("K-map") of the Boolean function F is below.

B \ A	0	1
0	1	x
1	x	0

Write down ALL of the minimum sum of products (SOP) expressions for F:

(b) **(11 points)** The K-map of the Boolean function G is below.

C \ AB	00	01	11	10
0	x	1	0	x
1	1	0	x	x

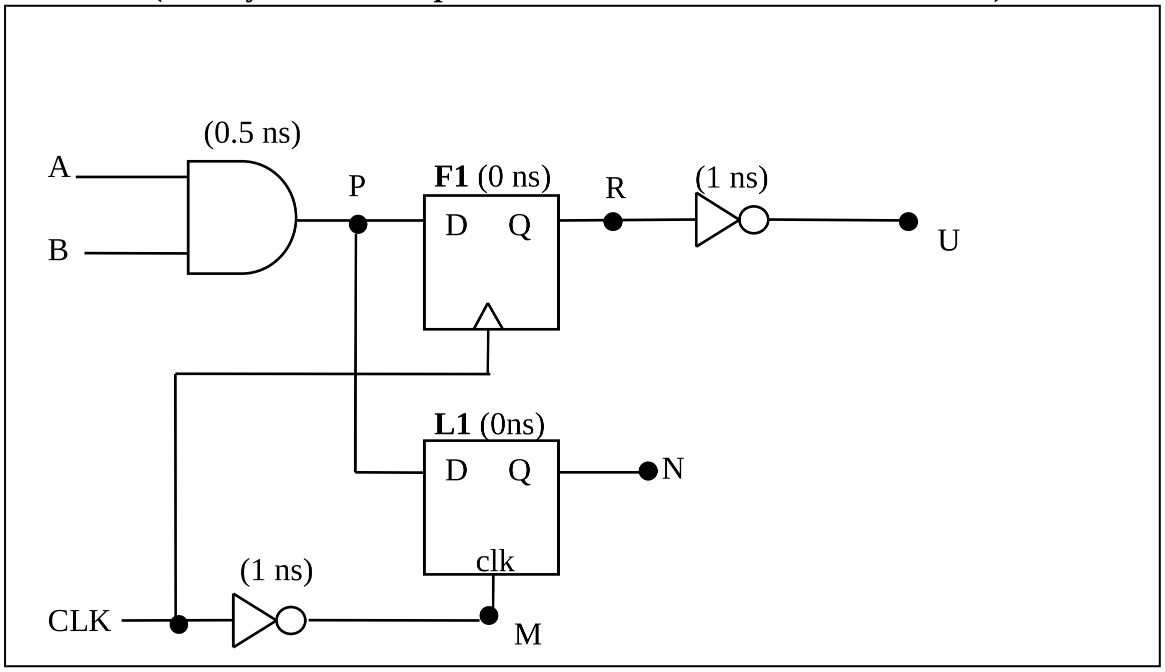
(b.1) Mark <u>all</u> prime implicants and <u>all</u> essential prime implicants on the K-map above.

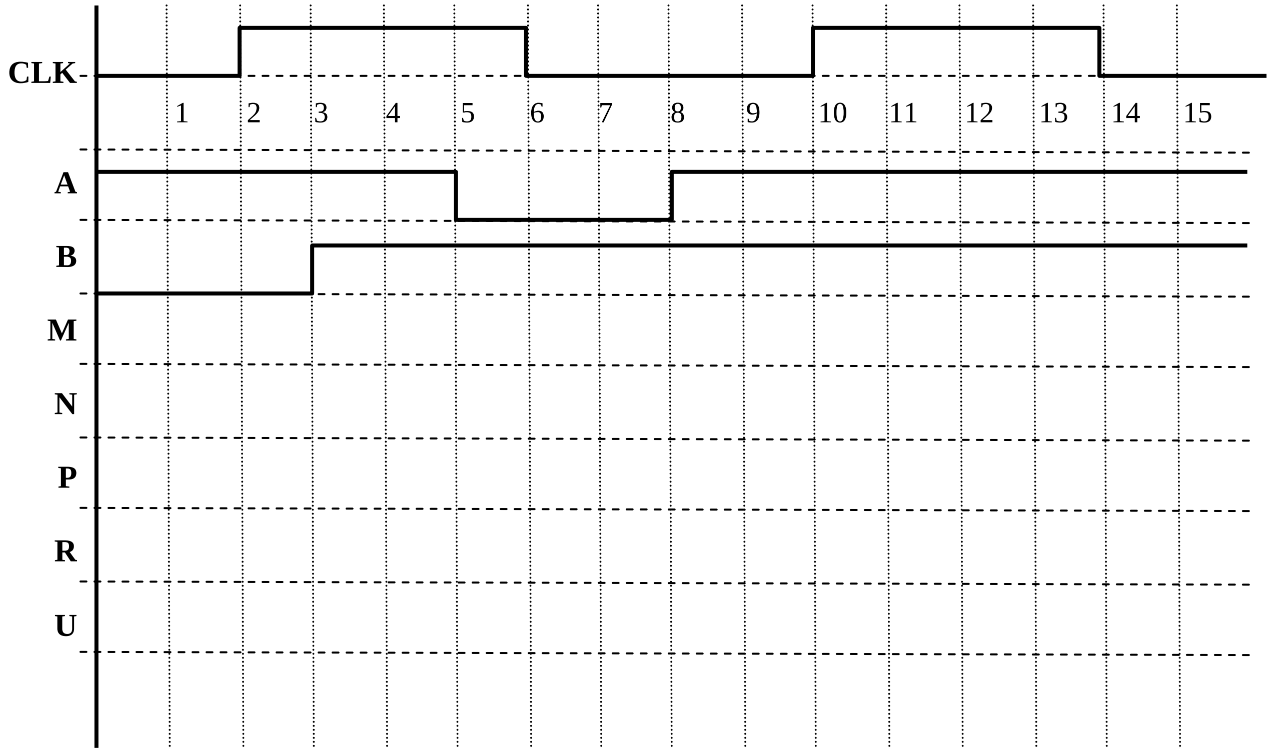
(b.2) Find ALL of the minimum SOP expressions for G. (Your answer must be in algebraic form.)

DIFFICULTY LEVEL 1

Problem # 2 TIMING DIAGRAMS [25 points]

The circuit for this problem is given below. **See the next page for the problem statement. (Show your results up to 15.5 nanoseconds of simulation time.)**





In this circuit, L1 is a positive level-sensitive D latch.

F1 is **positive** edge-triggered D flip-flop.

CLK is the clock input signal.

A and B are input signals.

M, N, P, R, and U are nodes in the circuit.

The gate delays have been shown in the figure (and are listed below):

Inverter:	1 ns
AND gate:	0.5 ns

The propagation delays of latch L1 and flip-flop F1 are negligible (0 ns).

External wires have negligible (i.e. zero) delay.

Assume that the inputs A and B have been stable for a very long time before time t = 0, at their initial values shown: A = 1, B = 0.

Assume that the CLK has been running for a long time before t = 0, in the pattern shown in the figure. (The clock period is 8 ns.)

Complete the timing diagram **that appears beneath the circuit on the previous page**. The waveforms for the inputs A, B, and CLK are given.

Fill in the waveforms for M, N, P, R and U.

Each waveform in your answer MUST be shown up to 15.5 nanoseconds of the simulation. (See the time axis on the diagrams on the previous page.)

Grading: Each waveform is worth 5 points. For each waveform, no partial credit will be given.

Problem # 3 VERILOG & ADDERS [12 points]

In Lab # 2, by writing the Verilog code, you have built a 4-bit Ripple Carry Adder (RCA) using four Full Adders.

Assume that the interface of this 4-bit RCA is given as follows:

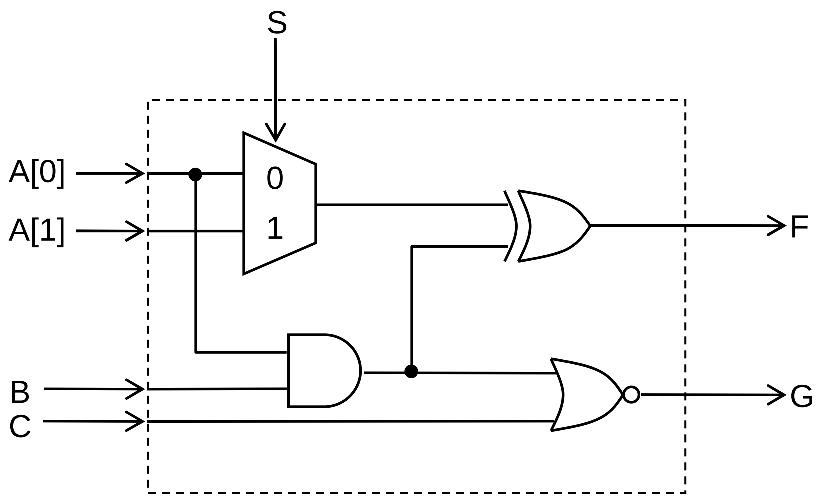
```
module RCA_4(sum, overflow, A, B, C_in);
        input[3:0] A, B;
        input C_in;
        output[3:0] sum;
        output overflow;
```

Throughout this problem, assume that addition is performed over unsigned binary numbers.

Using the above 4-bit RCA module, write a complete Verilog module to implement an 8-bit RCA.

Note: You must instantiate the 4-bit RCA module, and aim for a design that uses the 4-bit RCA module as maximally as possible, with as little additional hardware as possible.

Problem # 4 VERILOG [14 points]



The above circuit has a 2-bit vector input A; 1-bit control input S for the mux; 1-bit inputs B, and C; and 1-bit outputs F and G.

Write a **complete Verilog module** that implements the above circuit using ONLY **PROCEDURAL** assignments.

- You may NOT assume that a 2:1 mux is available as a module. You must express the 2:1 mux using only procedural assignments.
- You must use the vector notation in Verilog to represent the input A.

(Continue your work here.)

DIFFICULTY LEVEL 2

Problem # 5 SHIFT REGISTER DESIGN [12 points]

By drawing the schematic, implement a 2-bit Shift Register using only J-K flip flops as memory elements.

(You may use additional combinational logic elements such as muxes.)

The shift register has a Serial Input “SI”, and outputs the contents of its registers.

In addition to the Serial Input mode, the 2-bit Shift Register must be able to perform in “Parallel Load” mode, in which an outside 2-bit vector can be input simultaneously into the flip-flops in parallel.

(For your reference: The J-K flip flop works as follows: J is the “set” terminal, and K is the “reset” terminal; that is, if J == 1, and K == 0, then Q = 1. If J == 0, and K == 1, then Q = 0. If J == 0, and K == 0, it holds. If J == 1, and K == 1, it toggles.)

(You may continue your work here.)

Problem # 6 COUNTER DESIGN [21 points]

(a) **(5 points)** By drawing the state diagram, design a counter that counts down as follows:

5, 4, 3, 2, 1, 0, 5, 4, 3, 2, 1, 0, …

Design this as a Moore machine. You have to make sure that you show (on your state diagram) exactly what the output of each state is, separately from any state names you might use.

The counter has a synchronous Reset, which resets to the “5” state. In your state diagram, you may show this with a single arrow to the “5” state.)

(b) **(16 points)** Assume that you have built the counter in Part (a), and you have this as a hardware block.

By using only combinational logic as additional logic, show us how you can use this hardware block to build a counter that counts as:

0, 1, 2, 3, 2, 1, 0, 1, 2, 3, 2, 1, 0, …

You are NOT allowed to use any feedback loops in this part. (In particular, you are not allowed to design the above counter from scratch.) Your design MUST utilize the counter you built in Part (a) as a hardware block, and build ONLY combinational logic around it.

Show us the schematic of your circuit. (You have to define clearly ANY variables that you use.)

(You may continue your work here.)

DIFFICULTY LEVEL 3

Problem # 7 DECODERS [15 points]

We have already defined a 2:4 decoder in the lectures.

A 2:4 decoder "with Enable" works as follows: If Enable == 1, then the 2:4 decoder works as we described in class. If Enable == 0, then all of the 4 output wires are set to zero.

By drawing the schematic, **build a 4:16 decoder using ONLY 2:4 decoders with Enable, using the minimum number of 2:4 decoders possible.**

Note: You are NOT allowed to use any additional combinational logic blocks. Only 2:4 decoders are allowed.

(You may continue your work here.)