Name:____
Perm #:____

Lab Section TA:______________________________________

ECE 152A-Winter 2008

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		14
4		16
5		25
6		22
7		16
TOTAL		133

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	x

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	1	x

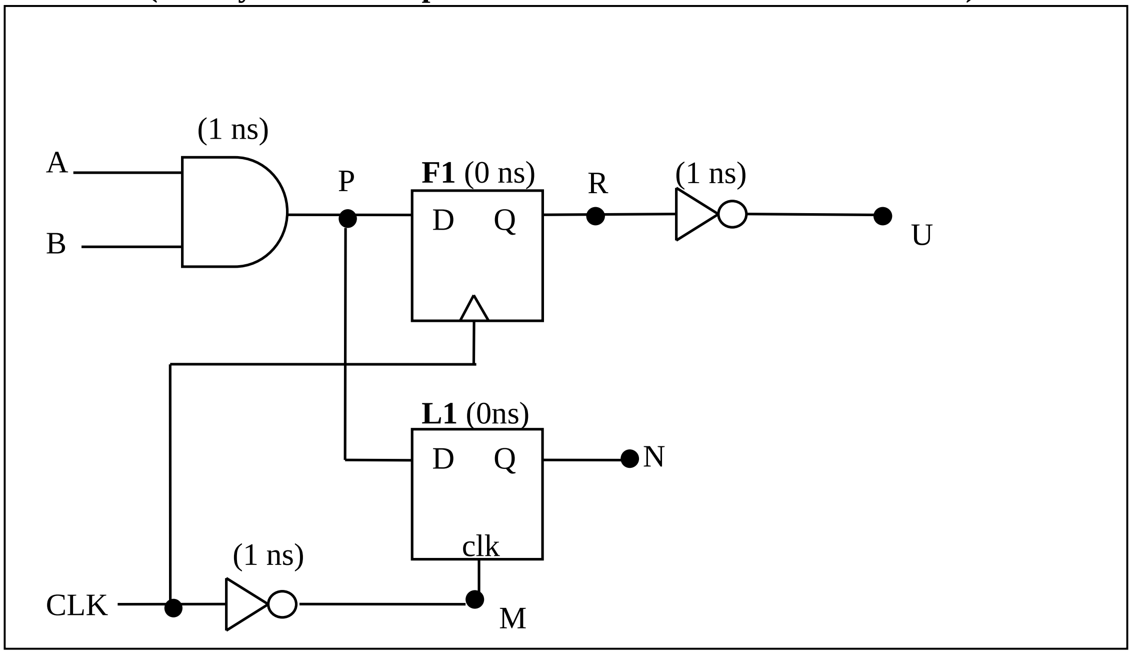
(b.1) Mark <u>all</u> prime implicants and <u>all</u> essential prime implicants on the K-map above.

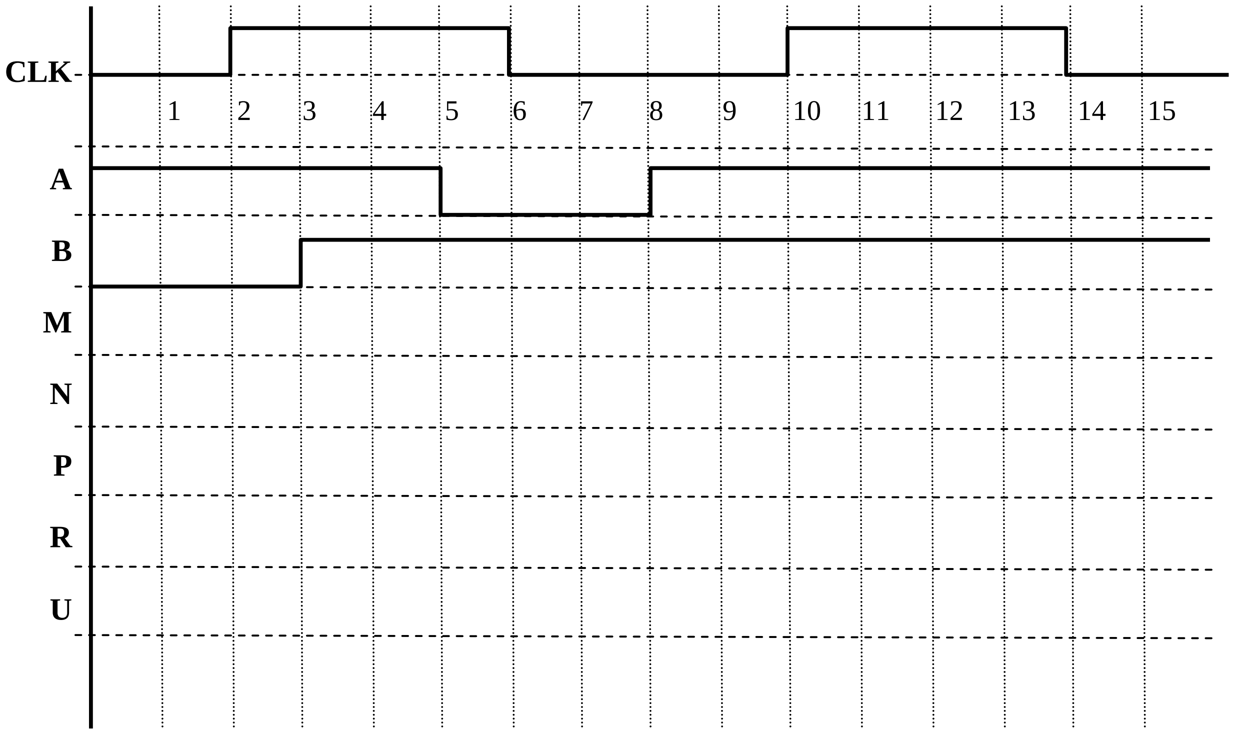
(b.2) Find ALL of the minimum SOP expressions for G. (Your answer must be in algebraic form.)

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:	1 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.)

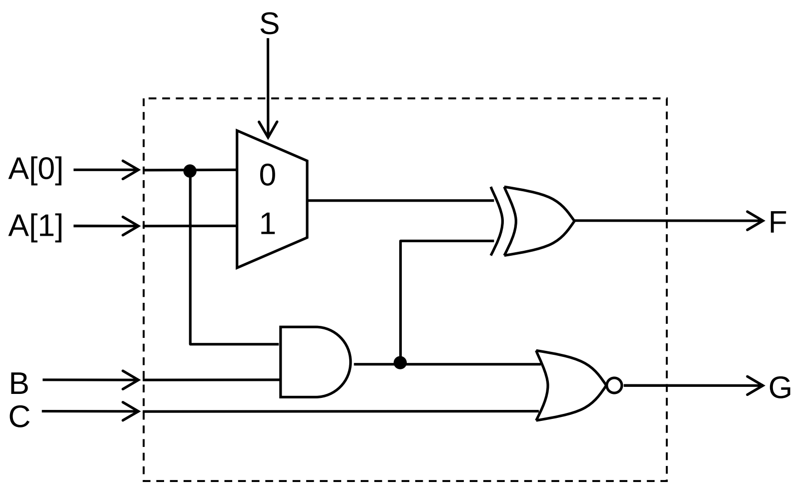
Complete the timing diagram **that appears beneath the circuit on the previous page**. The waveforms for the inputs A, B, and CLK are given.

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** **[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 continuous assignments **that use the assign keyword.**

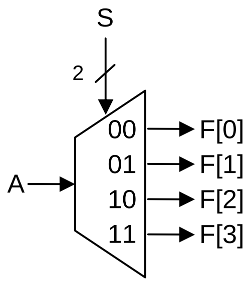
- You may NOT assume that a 2:1 mux is available as a module. You must express the 2:1 mux using only continuous assignments.
- You must use the vector notation in Verilog to represent the input A.

(Continue your work here.)

DIFFICULTY LEVEL 2

Problem # 4 1:4 DEMUX [16 points]

A 1:4 DEMUX ("demultiplexor") is defined as follows: It has a 2-bit control input S; a 1-bit data input A; and a 4-bit vector output F. The 1:4 demux routes the 1-bit data input A to the output 'F' specified by the address in S, and sets the remainder of the output bits to don't cares.



Examples:

- If S = 2`b00, then F[0] = A, and F[1], F[2], and F[3] are all don't cares.
- If S = 2`b11, then F[3] = A, and F[0], F[1], and F[2] are all don't cares.

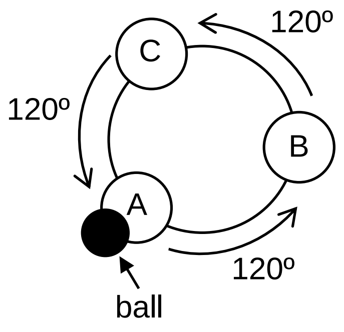
Write a complete Verilog module that implements a 1:4 DEMUX using **only procedural assignments.** (Implementations that use other techniques will get zero credit).

You MUST use the vector notation in Verilog. Implementations that do not use the vector notation will be penalized.

(You may continue your work here.)

Problem # 5 ROTATOR DESIGN [25 points]

Design a **combinational logic circuit** that will be used to control a rotator. The physical set up of the Rotator system is shown in the figure. There is a ball, and there are 3 positions (A, B or C) in which the ball can be located.



The controller will work as follows: Given the current position of the ball (in either position A, B, or C), and an input signal that specifies the degree of rotation (counterclockwise, in multiples of 120º), the controller produces the next location of the ball (A, B, or C). Assume that the only degrees of rotation are 0º, 120º, and 240º (since all others can be expressed as a number of full rotations plus one of these).

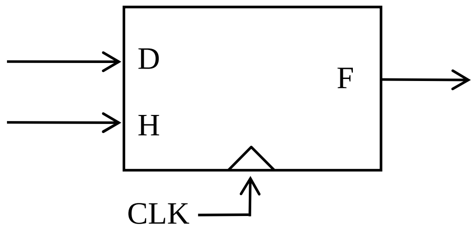
For example, if the current position of the ball is "A" as shown in the figure, and the input signal specifies a rotation of "240 degrees", then the output of the controller will produce the correct position "C".

(a) **(8 points)** Define your input and output variables. You must explain what each variable corresponds to and relate it clearly to the operation of the Rotator. (Note that you need to define Boolean variables. A, B, C are not Boolean variables; they are positions. Do NOT use A, B, C as names for the Boolean variables you define.)

(b) **(3 points)** Draw the interface schematic (also known as the “top-level schematic” or “symbol”). Clearly show all of the inputs and outputs. For vector variables, include their bitwidths.

(c) **(14 points)** Write down the truth table. **(Your design MUST make use of don’t cares where applicable.)**

Problem # 6 SAMPLE-OR-HOLD ("SOH") FLIP-FLOP [22 points]



In this problem, we design a new flip-flop called a **sample-or-hold ("SOH") flip-flop.** The main idea is that a control input H specifies whether the device samples its data input, or holds its current value at the active clock edge. The precise definition of the SOH flip-flop is as follows:

- It has a 1-bit data input D; a 1-bit control input H ("H" stands for "hold"); a clock input CLK; and a 1-bit output F.
- The device is positive edge-triggered; that is, it holds the value of F at all times outside of the rising clock edge.
- At the rising clock edge, if H == 0, the device samples its data input D; that is, F gets D at the rising clock edge in this case.
- At the rising clock edge, if H == 1, the device holds the current value of F at the rising clock edge.

(a) **(6 points)** By drawing the schematic, implement the SOH flip-flop using **only** **D flip-flops** as memory elements. (You may use additional combinational logic, but you may not use other types of flip-flops as memory elements.)

(b) **(16 points)** By drawing the schematic, **implement a J-K flip-flop** using only a **single** SOH flip-flop as a memory element. (You may use additional combinational logic, but you may not use other flip-flops, such as D-, or T- flip-flops.)

In your design, you are NOT allowed to connect the H terminal of the SOH flip-flop to the logic value "0". (You MUST think of a way to utilize the Hold feature of the SOH flip-flop, in order to achieve the Hold operation of a J-K flip-flop.)

(Reminder: A J-K flip-flop works as follows: If J = 1, and K = 0, it sets Q = 1. If J = 0, K = 1, it resets Q = 0. If J = 0, and K = 0, it holds Q. If J = 1, K = 1, it toggles Q to Q'.)

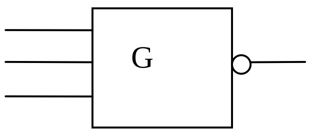
DIFFICULTY LEVEL 3

Problem # 7 BUBBLE-PUSHING ON A NEW GATE [16 points]

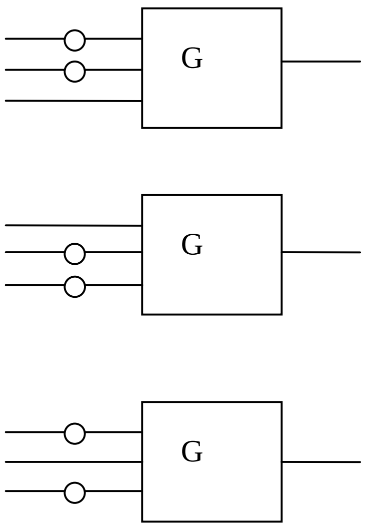
We would like to design a new combinational logic gate, called "G", that has three 1-bit inputs A, B and C, and one 1-bit output F.

(Recall that on a circuit schematic, a "bubble" denotes an inverter.)

The new gate G has the following bubble-pushing law: When a bubble at the output of G is pushed back through G, the gate G remains unchanged, and bubbles are produced at exactly any choice of two of the inputs of G. These bubble-pushing laws are shown in the diagram below:



is equivalent to each of the three on the right



Write down the truth table, and a minimum sum of products (SOP) expression for the new gate G. (If such a logic function G does not exist, prove that it does not exist.)

(You may continue your work here.)

(You may continue your work here.)