Name:____
Perm #:____

Lab Section TA:_____________________________________

ECE 152A-Fall 2005

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.

6. There are 23 pages in this exam. When you are instructed to start, first check that you have all the pages.

7. There is a total of 181 points on this exam.

8. You may use additional blank pages for extra work. (These are available in the front of the room.) However, these will be graded only if you clearly indicate so, label it with the part and problem number, and have them stapled to the main set of pages at the end.

LEAVE THIS PAGE BLANK

PROBLEMS:

DIFFICULTY LEVEL 0

Problem # 1 ALGEBRAIC SIMPLIFICATION [6 points]

Simplify the following Boolean expressions algebraically. (Show ALL your steps. NO credit will be given to other methods such as K-maps.)

(The complement of a variable X is denoted by X ' . The superscript D denotes the dual.)

(a) $X + X'YZ$

(b) $(X + Y')^D$

Problem # 2 BASIC DEFINITIONS [21 points]

Consider the Boolean function: F = A + B

Fill out the following lines. YOUR ANSWERS MUST BE ALGEBRAIC EXPRESSIONS. (No other forms will be accepted.) Clearly separate the expressions by commas, if a list of items is the answer.

(a) Implicants of F:

____

(b) Prime implicants of F:

____

(c) Essential prime implicants of F:

____

(d) Minterm expansion of F:

____

(e) Maxterm expansion of F:

____

(f) Minimum sum of products expression for F:

____

(g) Minimum product of sums expression for F:

____

Problem # 3 WORD PROBLEM [6 points]

By drawing the schematic, implement a Boolean circuit that takes two 4-bit unsigned numbers as inputs, and produces a 1 if both numbers are even, and a 0 otherwise.

CLEARLY DEFINE YOUR INPUT AND OUTPUT VARIABLES.

Problem # 4 NAND-ONLY IMPLEMENTATION [7 points]

By drawing the schematic, implement the function $F = A + B\,C$ using only NAND gates.

(YOU MUST PUT A BOX AROUND YOUR FINAL ANSWER TO INDICATE YOUR FINAL ANSWER.)

Problem # 5 VERILOG [5 points]

By completing the module below, implement the **half adder** in Verilog using **ONLY continuous assignments that use the "assign" statement. (NO credit will be given for explicit gate instantiations).**

```
module halfAdder(A, B, cout, sum);

        input A, B;
        output cout, sum;
```

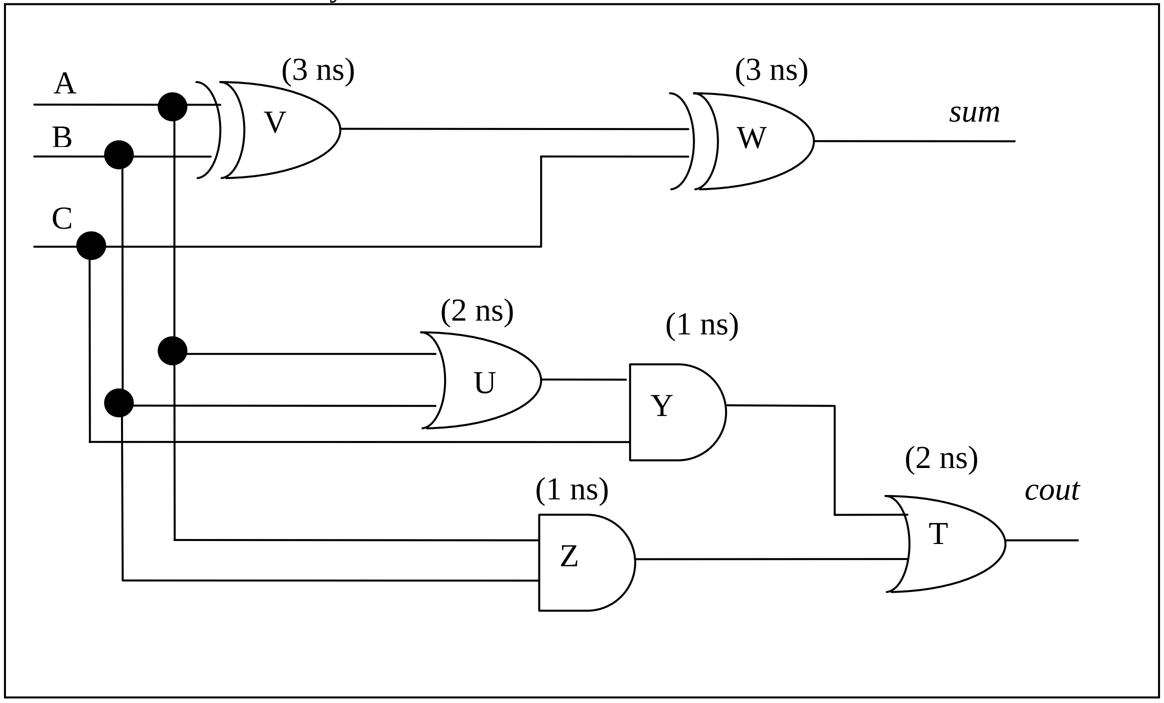
Problem # 6 FULL ADDER [20 points]

The following is an implementation of the full adder. It uses the Boolean equations:

$$sum = A \oplus B \oplus C$$

$$cout = (A + B)C + AB$$

where *C* denotes the Carry-In:



In this problem, assume that this full adder is used as a combinational logic block by itself, and is not cascaded with any other blocks.

T, U, V, W, Y and Z are the names of the logic gates shown.

Assume that the external wires have negligible (i.e. zero) delay, and the gates have the delays (in nanoseconds) shown above each gate in the figure.

For the above implementation of the full adder, fill out the following:

(a) **(1 point)** Gate count: ____

(b) **(2 points)** Number of logic levels: __________________________________

(c) **(3 points)** What is the maximum fan-in?______________________________

(d) **(2 points)** What is the maximum fan-out seen **by any input or any gate**?

____

(e) **(2 points)** Maximum delay (in nanoseconds):________________________________

(f) **(2 points)** Minimum delay (in nanoseconds):_______________________________

(g) **(3 points)** Critical path (Write down the input variable that begins the path, the sequence of gates on the path, and the output variable at the end of the path):

____

(h) **(5 points)** Give one critical transition. (There may be many, but write down only one of these.)

Previous values of all inputs	New values of all inputs

DIFFICULTY LEVEL 1

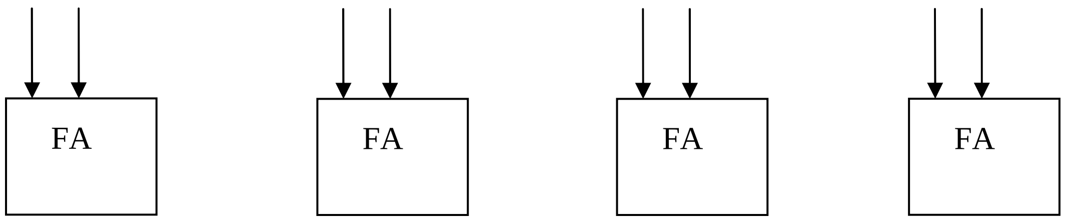
Problem # 7 AREA / DELAY ANALYSIS [25 points]

In this problem, we implement a 4-bit ripple carry adder using 4 full adders in cascade.

The inputs are A[3:0], and B[3:0], and the output is sum[3:0].

Assume that we use the full adder implementation given in Problem # 6. (ANSWERS BASED ON OTHER FULL ADDER IMPLEMENTATIONS WILL RECEIVE ZERO CREDIT.)

(a) **(5 points)** In the implementation schematic below, show how to connect the 4 full adders. YOU MUST LABEL ANY REMAINING VARIABLES CLEARLY ON THE FIGURE.



(b) **(4 points)** What is the total gate count of this implementation? (Show your work.)

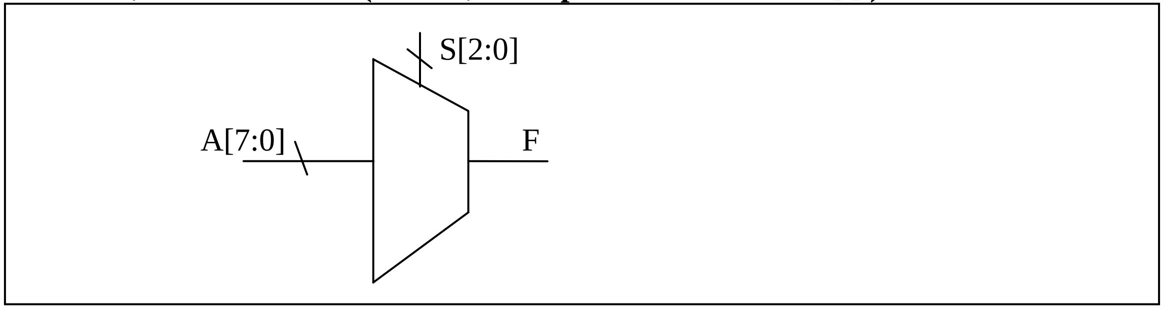
(c) **(13 points)** Assume that all of the input variables are available at time t = 0 nanoseconds (indicated by @0 nanoseconds). On your diagram in (a), clearly show when each of the outputs become available. Use the @ notation.
THE DELAYS ARE IN NANOSECONDS. THE GATE DELAYS IN NANOSECONDS ARE GIVEN IN PROBLEM # 6. (ASSUMING A UNIT DELAY FOR EACH GATE WILL RESULT IN ZERO CREDIT.)

(d) **(3 points)** What is the worst-case delay **(in nanoseconds)** of this ripple carry implementation? (Assume that this ripple carry adder will be used by itself, and will not be cascaded with other ripple carry adders.)

Problem # 8 8:1 MUX [15 points]

The interface schematic of the 8:1 mux appears below, where A[7:0] is the data input; S[2:0] is the Select input, and F is the output.

A[0] is the input to address 0; A[1] is the input to address 1; ...; A[7] is the input to address 7, of the 8:1 mux. (That is, the input to address i is A[i].)

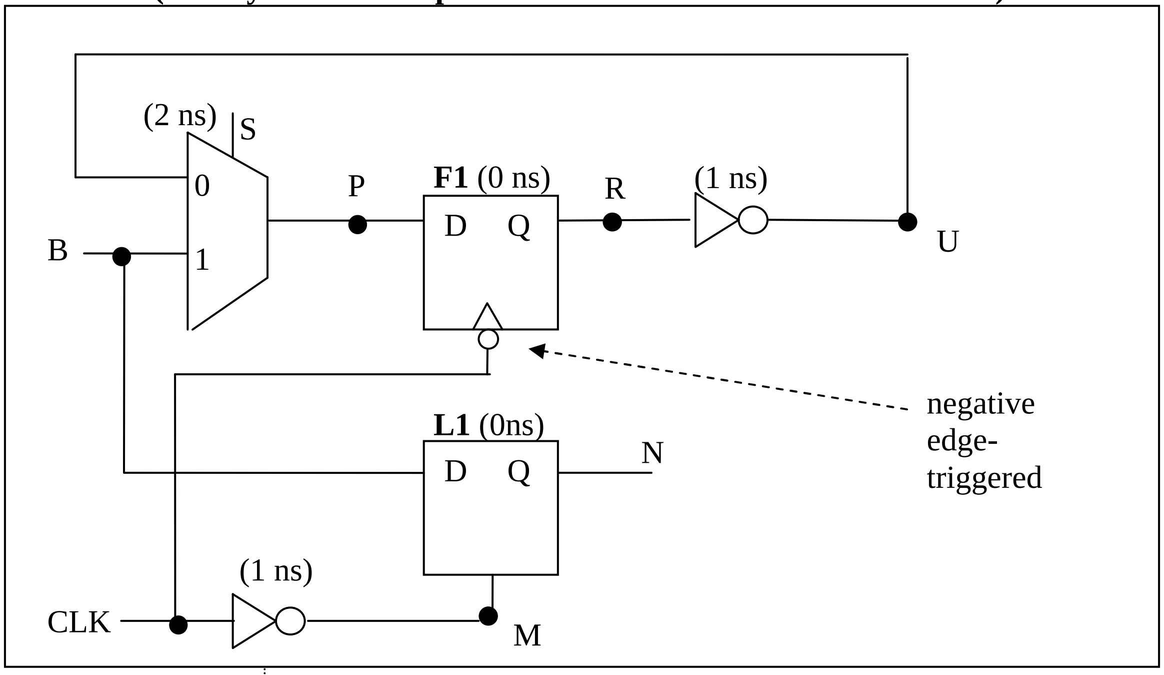


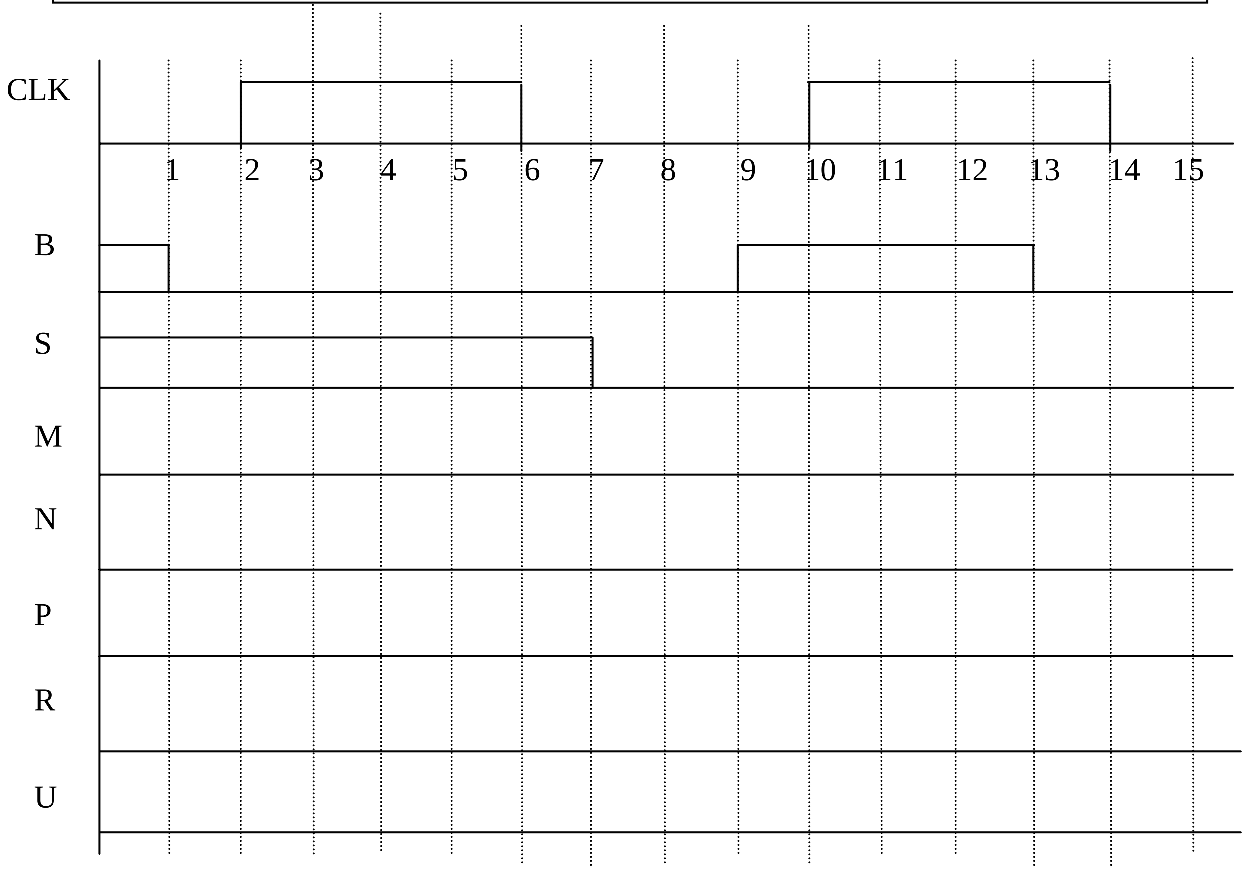
By drawing the implementation schematic, implement an 8:1 mux using **ONLY** 4:1 muxes. Use the minimum number of 4:1 muxes possible. **You MUST use the input and output variables given in the interface schematic above.**

(This page is for the continuation of your answer to Problem # 8.)

Problem # 9 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 **negative** edge-triggered D flip-flop.

CLK is the clock input signal.

B and S 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
2:1 mux:	2 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 B and S have been stable for a very long time before time t = 0, at their initial values shown: B = 1, S = 1.

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 B, S, 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.)

Problem # 10 VERILOG [15 points]

By completing the module below, implement the **half adder** in Verilog using **ONLY procedural assignments**. **(NO credit will be given for implementations that use ANY continuous assignments.)**

```
module halfAdder(A, B, cout, sum);

        input A, B;
        output cout, sum;
```

(On this page, you may continue your answer to Problem # 10.)

DIFFICULTY LEVEL 2

(Advice: Look through Problems 11, 12 and 13. Start with the one(s) that you think you can answer.)

Problem # 11　　　　SHIFT REGISTER DESIGN　　　　[12 points]

By drawing the schematic, implement a 3-bit shift register using only 3 D flip-flops as memory elements (and you may use additional **combinational** logic elements). The shift register MUST correctly implement BOTH of the following functions. (NO credit will be given for implementations that can perform only one of these functions.)

1) When a control input "SR" is asserted, the shift register shifts the bits to the right (in each clock cycle).

2) When a control input "SL" is asserted, the shift register shifts the bits to the left (in each clock cycle).

The same serial input, called "K", is used as a data feed for both the SR and SL modes.

If neither SR nor SL is asserted, the shift register holds its current value.

There is no parallel load capability.

(Assume that the external circuitry assures that SR and SL are never asserted at the same time.)

(This page is for the continuation of your solution for Problem # 11.)

Problem # 12 FUNCTIONAL COMPLETENESS [12 points]

A new logic gate G has been proposed that has the following truth table. (A and B are inputs and F is the output.)

A	B	F
0	0	0
0	1	1
1	0	0
1	1	0

Prove or disprove: G is a universal gate.

You MUST circle your assertion here: TRUE FALSE

Proof: (Only correct proofs will get credit. No credit for just guessing TRUE or FALSE or for giving a partially correct proof.)

(This page is for the continuation of your solution for Problem # 12.)

Problem # 13 FLIP-FLOP IMPLEMENTATION [12 points]

By drawing the schematic, implement a **T flip-flop** with **synchronous reset** using ONLY NAND gates. (Implementations that do not work or use ANY other gates will get zero credit.)

PUT YOUR FINAL ANSWER IN A BOX.

(This page is for the continuation of your solution for Problem # 13.)