Name:______________________________________

Perm #:______________________________________

Lab Section TA:________________________________

ECE 152A-Winter 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 22 pages in this exam. When you are instructed to start, first check that you have all the pages.

7. There is a total of 100 points on this exam.

PROBLEMS:

Problem # 1 [30 points] LOGIC DESIGN

The circuit shown in Figure 1 is called a "2-input (unsigned) adder". It has two 2-bit data inputs "*a*" and "*b*" and one 1-bit control input "*Add*". It treats its data inputs as **unsigned** 2-bit numbers. Further, the circuit has one 3-bit output "*y*" and one 1-bit output "*Valid*". If *Add* is asserted, this circuit adds its two 2-bit data inputs to produce a 3-bit output *y*, and sets *Valid* to 1, to indicate that the *y* vector has a valid output value. If *Add* is not asserted, then the circuit sets *Valid* to 0, to indicate that the output *y* is not a valid output value (and the vector *y* can be set to any desired value in this case).

(You must follow the following convention in designing this circuit: We denote the most significant bit (MSB) of *a* as *a[1]* and the LSB of *a* as *a[0]*. Similarly, for input *b*. We denote the MSB of *y* as *y[2]*, and the LSB of *y* as *y[0]*.)

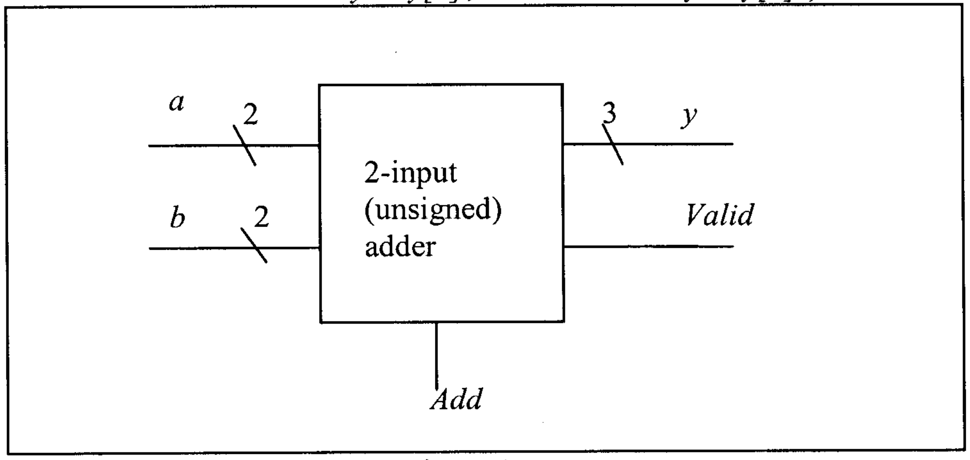


Figure 1

Each part of this problem may introduce some additional assumptions. The additional assumptions of each part are to be used only for that part. Do NOT carry the additional assumptions of one part into the other parts of the problem.

(a) **(12 points)** Find a minimum sum-of-products (SOP) and a minimum product-of-sums (POS) expression for each of the outputs *y[1]*, and *Valid*. (Note that you do NOT need to write such expressions for *y[0]* and *y[2]* in this part.)

(HINT: In order to save time, it might be better to write down the Karnaugh map for each output variable directly rather than writing down the truth table first.)

You may use any technique you wish (or a combination of techniques); however, **you must write your final answers on the following lines:**

Minimum SOP:

y[1] = ____

Valid = ____

Minimum POS:

y[1] = ____

Valid = ____

(You may do any extra work in the following space:)

(b) **(5 points)** For only the output variable *y[1]*, write down an algebraic expression for each prime implicant and say which of these expressions are essential prime implicants.

(c) **(10 points)** By drawing the schematic, implement the 2-bit unsigned adder using **only** XOR and NAND gates. Use design intuition to aim for the smallest number of gates. (Complex designs will be penalized, and simple designs with small hardware will win points.)

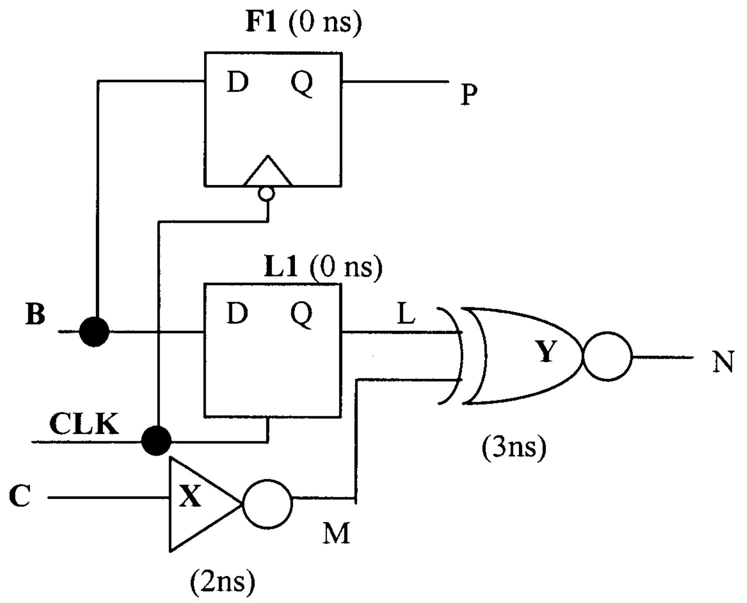
(d) **(3 points)** Is it possible to implement the 2-bit unsigned adder using **only** NAND gates? Why or why not?

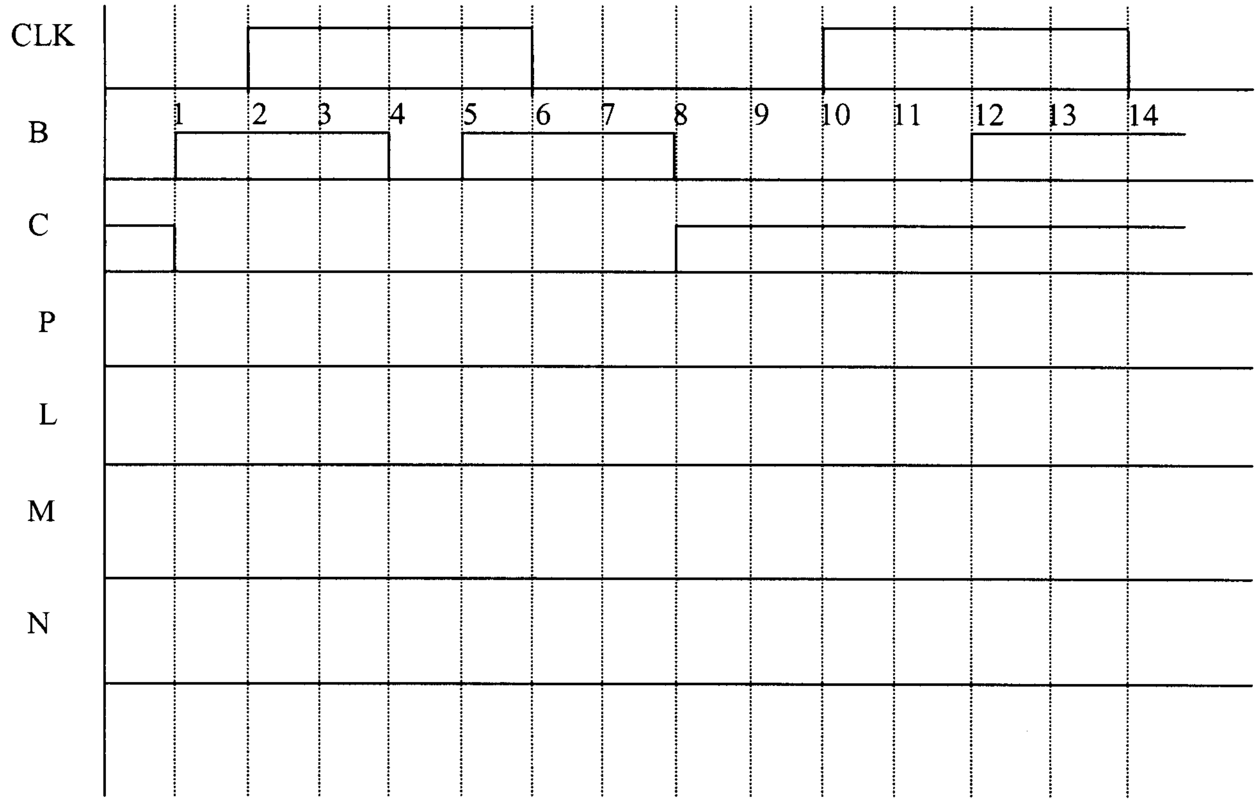
Problem # 2 [16 points] VERILOG

(a) **(8 points)** In **Verilog**, implement a 4:1 mux using only 2:1 muxes as building blocks. Assume that the 2:1 mux module is available. First, write down the prototype declaration for a 2:1 mux module, including the input and output declarations for the terminals, (but do not show the rest of the implementation). Then, write down the implementation of a 4:1 mux that uses 2:1 muxes.

(b) **(8 points)** Write a complete Verilog module for a "2:4 decoder with Enable", **using procedural assignments in a 'case' or a 'casex' statement**. This decoder has an Enable input. If Enable is not asserted, the decoder outputs the zero vector. If Enable is asserted, the decoder decodes as usual.

Problem # 3 [16 points]:
The circuit for this problem is given below. **See the next page for the problem statement.**





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 C are data input signals. L, M, N, and P are nodes in the circuit.

The delay of the inverter X is 2 ns, and the delay of the XNOR gate Y is 3 ns.

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

(Large black dots indicate wire connections. No large black dot means there is no connection between crossing wires.)

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

Assume that the data inputs B and C have been stable for a very long time at B = 0 and C = 1, before time t = 0.

Complete the timing diagram **that appears beneath the circuit on the previous page**. The waveforms for the inputs B, C and CLK are given. Fill in the waveforms for P, L, M and N.

Problem # 4 [38 points]:
We want to implement a **synchronous** counter that is described as follows: The only inputs to the counter are a 1-bit control input called "Load", a 1-bit control input called "Reset", a 2-bit data input vector A, and a clock input. The counter outputs the current count.

When Load is asserted, the counter loads synchronously the input vector A, regardless of the value of Reset. (That is, the count is set to the 2-bit data input vector A.) If Load == 0, and Reset == 1, the counter **sets to its output to the value of 0** in a synchronous fashion. Otherwise, the counter counts in the sequence 0, 2, 1, 0, 2, 1, 0, 2, 1, ...

We will implement the counter as a **Moore machine**.

(a) **(2 point)** Draw the interface schematic for this counter.

(b) **(10 points)** Draw the state diagram for this counter. (All transition arrows must be shown and marked completely.)

(c) **(10 points)** Write down the state table that corresponds to your state diagram from Part (b).

(d) **(10 points)** Write down the next-state and output equations in minimum SOP form by using K-map minimization.

(e) **(6 points)** Implement this counter using only D flip-flops. (Draw the schematic.)