

ECE 465, Spring 2009, Instructor: Prof. Shantanu Dutt

Midterm: Thurs, March 19, Time: 2-3:15pm

Exam Format: Closed Book and Notes, except for 1 2-sided sheet of notes, Total Points: 100

Important Note: You need to show all your work **clearly** in deriving the answers. Just writing down the final answers is not enough.

Suggestion: Begin by reading all questions and do those first that you think you know best.

1. Multi-function QM and PLA Delay

As we discussed in class, and as is shown in Fig. 1, there are 3 components to a PLA's delay:

- D_1 : Literal line delays: Each literal line X 's delay depends on the number m_X of PIs that have the literal X in their product terms. This is so because: (a) X will need to drive the gates of m_X transistors; (b) each gate contributes to the total capacitance C_X of the literal line X ; and (c) the delay on this line is proportional to RC_X , where R is the resistance of the gate (buffer or inverter) driving a literal line (R is the same for all literal lines).
- D_2 : Product/AND line delays: Each product/AND line Y 's delay depends on the number k_Y of functions in which it is included, since this is the number of transistor gates that product Y will drive in the OR plane.
- D_3 : OR line delays: OR lines do not drive any transistor gates. However, if there are p AND lines in the PLA, then each OR line has p transistor drains connected to it (irrespective of whether the transistors gate connection to an AND line is retained or blown), and each contributes a small amount of capacitance (much smaller than a gate capacitance) to an OR line. So minimizing the total number p of AND terms (which is the goal of PLA hardware cost minimization) also minimizes the D_3 component of a PLA's delay.

Here we will be concerned only with the D_2 component of a PLA's delay. Our objective, however, is first to have as few PIs as possible across all functions to be implemented by the PLA, and then reduce D_2 delays. Minimizing the total # of PIs (i.e., AND lines) is facilitated by having as much sharing of PIs across multiple functions as possible (as we have seen in class). However, *unnecessary* PI sharing does not reduce p (the # of AND lines), but increases the delay of one or more AND lines (i.e., increases the D_2 delays of one or more AND lines).

In class, we discussed that Rule 7 (stated below for your convenience) is not necessary to be applied to PLA hardware minimization, since after a PI is included in a function, there is *zero additional hardware cost* of including it in other functions for which it is an implicant. Please answer the following questions:

- (a) Give an example of *unnecessary* PI sharing using a relevant portion of a PIT (unnecessary PI sharing does not reduce the total number of AND lines). Explain clearly why your example shows unnecessary PI sharing. **7**
- (b) Explain, by showing a relevant portion of a PLA's transistor based schematic (of the type shown in Fig. 1), how unnecessary PI sharing can increase the delay of some AND line(s). **8**
- (c) Comment on whether it is useful or useless to apply Rule 7 during PLA optimization when we are interested in reducing p and also AND line delays (i.e., D_2 delays); note that the first priority is to reduce p , and once that is done, reduce the D_2 's. Explain your answer using a suitable example showing relevant portions of a PIT and a PLA. **20**

Rule 7 (changed to be applicable to PLAs): If a multi-function PI g is an EPI (or pseudo EPI) for some functions, but not for all for which it is an implicant, it is only chosen for the former function(s) and not right away for the latter. After this partial inclusion (for some functions), g 's cost is reduced to 0 (the 0 represents that there is no more hardware cost for including g in other functions for which it is an implicant), and it may be chosen later, for one or more of the remaining functions, based on this reduced cost (e.g., when it becomes a pseudo-EPI for one or more remaining functions).

2. You need to design a 16-bit adder of minimum possible delay using a divide-&-conquer (D&C) approach. To design the adder you have been provided with 32 full adders (FAs), a 4-bit 2:1 Mux and a 1-bit 2:1 Mux (an m -bit 2:1 Mux has two data inputs I_0, I_1 , and each input is an m -bit input, and the output is m -bits as well). It is not necessary to use *all* the given components, but you should use as many components as you can to minimize the adder's delay.

Note that a D&C breakup of an n -bit adder into two $(n/2)$ -bits adders implies a dependency of the higher-order $(n/2)$ -bit addition on the result of the lower-order $(n/2)$ -bit addition. The *stitch-up* function strategies that may be used to tackle this dependency can be one or a combination of “wait”, “speculative” (also called “design for all cases”) and “lookahead” strategies (as we discussed in class). With the above-mentioned hardware components at your disposal, use a combination of the “wait” and “speculative” strategies to obtain an adder with as small a delay as possible (note that for each break-up point or node of a problem/sub-problem in the tree, only one strategy can be used, but different break-up nodes can use different strategies). Assume that the delay of a FA is 10 ns, and the delay of a 2:1 Mux is 2 ns (irrespective of the number of bits for the inputs and output of the Mux).

Show or obtain the following (show your work clearly and methodically):

- (a) Your entire D&C tree, in which it is clear for each break-up node in the tree, whether the stitch-up is a “wait” or a “speculation”. **15**
 - (b) The corresponding adder circuit using the above components. **15**
 - (c) The delay of your adder (i.e., the delay of its maximum-delay or *critical* path), and the critical path in it. **10**
 - (d) For comparison, obtain the delay of a 16-bit ripple-carry adder (RCA). The delay of your adder should be less than the RCA's delay. **5**
3. Implement the function $f(A, B, C, D) = \sum_m(0, 5, 8, 10, 12, 14) + \sum_d(4, 7, 13, 15)$ using a 4:1 Mux and the minimum number of 2-input AND/OR gates. A is the MSB and D the LSB (the variables are given in left-to-right order from MSB to LSB); the corresponding K-map with the 1's and X's of the function are shown in Fig. 2. Assume that all variables and their complements are available. Show your work clearly. **20**