ECE 465, Fall 2009
Midterm Solution

1. Quine-McCluskey Optimization Technique

(a) State and give an example of the column covering rule in QM. 10
(b) Give the rationale for the column covering rule. 5
(c) Does the column covering rule affect the optimality of the final solution (i.e., does it preserve optimality or does it hamper optimality)? Clearly explain why or why not. 10

Ans. (a) A column $MT_i$ is said to cover another column $MT_j$, of $MT_i$ has at least X’s in all row positions that $MT_j$ has X’s in, and possibly some more. The figure on the right shows an example of column covering in which $MT_i$ covers $MT_j$.

The column covering rule states that the covering column ($MT_i$ in the above example) should be deleted.

(b) The rationale behind the column covering rule is that the covered column, $MT_j$, needs to be covered by some PI, and PI that is chosen to cover $MT_j$ will automatically cover the covering $MT_i$. Hence it is not necessary to explicitly consider $MT_i$ for covering, and it can thus be deleted from the PIT/RPIT to reduce the complexity of the problem (of determining a min-cost expression for the given function(s)).

(c) The column covering rule does not hamper optimality of the final solution since: (a) It does not change the set of PIs under consideration for the final solution; (b) It does not change the selection of PIs in a detrimental way. Part (a) is obvious. To understand (b), we consider the non-trivial case in which $MT_i$ and $MT_j$ do not have exactly the same X’s, and note that there are 2 ways a PI is chosen to be in a final expression: (i) It either becomes an EPI or pseudo-EPI (we will use them p-EPI henceforth to refer to both an EPI and a pseudo EPI); (ii) It is chosen in a cyclic PIT based on its cost and the # of MTs it covers. These two cases are considered next. Note that if in your answer, you had considered any of these scenarios in a formal manner as is presented next, you would have obtained full points, since any of these would show your grasp of this issue.

Case (i): The only PIs covering $MT_i$ that can be affected in their p-EPI status by $MT_i$’s deletion are those that cover $MT_i$ but not $MT_j$ ($PI_6$ in the above example). This is so because any PI, say, $PI_r$ covering both $MT_i$ and $MT_j$, can only become a p-EPI if $MT_j$ is a singleton column ($MT_j$ with only a single X in each row).
cannot be a singleton col as it covers MT\textsubscript{j}), and in that case PI\textsubscript{i} is chosen and covers both MT\textsubscript{i} and MT\textsubscript{j}, and we do not even have to proceed to applying column covering (in other words the choice of PI\textsubscript{i} to cover MT\textsubscript{i} and MT\textsubscript{j} is correct for optimality). Thus we only consider PIs such as PI\textsubscript{6} covering MT\textsubscript{i} but not MT\textsubscript{j}, whose p-EPI status can be affected after MT\textsubscript{j} is deleted. The only effect possible is that such a PI (let’s use PI\textsubscript{6} as an example without loss of generality) could have been a p-EPI had MT\textsubscript{i} not been deleted. But it is only possible for PI\textsubscript{6} to be a p-EPI via MT\textsubscript{i}, if either: (1) MT\textsubscript{i} is a singleton column when it covers MT\textsubscript{j} \Rightarrow MT\textsubscript{j} is a singleton col \Rightarrow PI\textsubscript{6} covers both MT\textsubscript{i} and MT\textsubscript{j}, and we reach a contradiction (we assumed that the PIs we are considering cover MT\textsubscript{i} but not MT\textsubscript{j}). Or (2) MT\textsubscript{i} becomes a singleton col at a later stage (if it is not deleted) with PI\textsubscript{6} as its only PI. This can only happen if all common PIs of MT\textsubscript{i} and MT\textsubscript{j} are deleted by covering or by becoming p-EPIs.

All such common PIs cannot be deleted by covering as this would leave MT\textsubscript{j} w/o any PIs, which is impossible (in other words, at least one of its PIs will remain uncovered by other PIs), and thus MT\textsubscript{i} cannot become a singleton col in this manner. If any common PI becomes a p-EPI, then it covers both MT\textsubscript{i} and MT\textsubscript{j}, and both are correctly deleted from the PIT, and we will never get to the stage where MT\textsubscript{i} becomes a singleton col. Thus any PI such as PI\textsubscript{6} can never be a p-EPI due to MT\textsubscript{i}, if MT\textsubscript{i} is not deleted.

**Case (ii):** A PI covering MT\textsubscript{i} but not MT\textsubscript{j} such as PI\textsubscript{6} can be chosen to break a cyclic (R)PIT if MT\textsubscript{i} is not deleted, but is not chosen if MT\textsubscript{i} is deleted (we assume that this choice is partly based on # of MTs, which changes for PI\textsubscript{6} due to MT\textsubscript{i}’s deletion and partly on cost, which is not affected by PI\textsubscript{6}’s deletion). This can happen since by deleting MT\textsubscript{i} we reduce the # of MTs covered by PI\textsubscript{6}. However, if MT\textsubscript{i} is not deleted, and PI\textsubscript{6} thus chosen to break a (R)PIT, then this decision is based on PI\textsubscript{6} redundantly covering MT\textsubscript{i}, since MT\textsubscript{i} will also be covered by a PI that is chosen to cover MT\textsubscript{j} (and this PI is not PI\textsubscript{6}, by our assumption). Thus the choice of PI\textsubscript{6} would be erroneous, since, by our assumption, if MT\textsubscript{i} is deleted, PI\textsubscript{6} is not chosen. Thus in this case the col covering rule actually helps the optimization process by eliminating an incorrect choice to break a cyclic (R)PIT.

*Thus in all scenarios we see that deleting MT\textsubscript{i} does not hamper expression optimization, and in one case may actually help it.*
2. **PLA-based design and PLA Delay**

Note: To do this problem tear out the two PLA schematics at the end of your question paper, write your name on them, program them (see below for instructions), and insert them together at this problem’s solution page in your answer book.

Consider the following non-SOP, non-POS expression of a function $f$:

$$f = (ab + bc' + cd)(ae + bd' + cd' + ce')$$

Using the two non-programmed PLA schematics given on the last 2 pages, do the following:

(a) Implement the function using the first schematic of a PLA without feedback. You need to show the PLA’s programming as described below. State the number of AND lines you needed to use; this is your hardware cost. Also, using the delay formulations given below, determine the exact delay $D$ of your PLA-based design.

(b) Implement the function using the second schematic of a PLA with feedback. You need to show the PLA’s programming as described below. State (the number of AND lines used) + (the number of feedback literal lines used) + (the number of OR lines used beyond the number of functions being implemented); this is your hardware cost. Also, using the delay formulations given below, determine the exact delay $D$ of your PLA-based design.

The programming of the PLAs should be shown as follows.

(i) Place $X$’s at the intersection of those literal lines (including feedback literal lines) $y$ and AND lines $p$ (this is in the **AND plane**), where $y$ belongs to the corresponding product term implemented on $p$. In the case of feedback PLAs, you also need to appropriately program the AND lines controlling the tri-state buffers controlling the feedback from some of the OR lines. Note that for each $X$ on it, a literal line drives a gate capacitance $C_g$. On the other hand, an $X$ on an AND line $p$ in the AND plane does not cause any extra capacitive load on $p$ (compared to not having the $X$).

(ii) Place $X$’s at the intersection of those AND lines $p$ and OR lines $z$ (this is in the **OR plane**), if $p$ belongs to the corresponding function implemented on $z$. Note that for each $X$ on it in the OR plane, an AND line drives a gate capacitance $C_g$. On the other hand, an $X$ on an OR line $z$ does not cause any extra capacitive load on $z$ (compared to not having the $X$).
Delay formulations:

(i) The delay of a literal line \( (D_1 \text{ delay}) = [\text{total # of AND lines used in the PLA}] + 10 \times [\# \text{ of X's on it}] \) units.

(ii) The delay of an AND line \( (D_2 \text{ delay}) = [\text{total # of literal lines used (including feedback literal lines, if any) in the PLA}] + 10 \times [\# \text{ of X's on it only in the OR plane}] \) units.

(iii) The delay of an OR line \( (D_3 \text{ delay}) = [\text{total # of AND lines used in the PLA}] \) units.

Ans. (a)

\[
\begin{align*}
    f &= (ab + bc + cd)(ae + bd + bd + ce) \\
    &= abe + abed + abc\bar{e} + ab\bar{c}e + bc\bar{d} + acde + \bar{bcd} + cd\bar{e}
\end{align*}
\]

Absorbtion

\[
\begin{align*}
    &= abe + abd + abc\bar{e} + bc\bar{d} + acde + \bar{bcd} + cd\bar{e}
\end{align*}
\]

AND lines used = 7
Extra literal lines used = 0
Extra OR lines = 0
Total cost = 7
Delay of all AND lines are the same (=10+10(1)) and OR lines are the same (=8). The maximum literal line delay determines the PLA delay, since the max literal line has at least a connection to some AND lines and this AND lines’ delay is the same as all other AND lines (=20).
Max delay literal lines=lone with most X’s=literal ‘a’ and ‘b’ lines. This delay = $8+10*4 = 48$. The PLA delay = max literal line delay (48) + any AND line delay (20) + the single OR line delay (8) = 76

Ans. (b)

\[ f = (ab + bc + cd) (ae + bd + \overline{bd} + ce) \]

No connections are needed to literal lines as both lines need to be at a constant logic ‘1’ value

AND lines used = 8
Additional literal lines used = 2
Additional OR lines used = 2
Total cost = 12
Delay= Again all AND lines used have the same delay of 12+10 (1) and all OR lines also have the same delay of 8.
Delay is again determined by the maximum literal line delay = $8+10(3)$= 38. (for the ‘b’ line that has 3 x’s).
Here total delay for $f_{1}f_{2} = 38 + 22 + 8 = 68$.
Delay for $f$ after $f_{1}f_{2}$ produced = $8 + 10 (1)$ (max literal line ($f_{1}$,$f_{2}$) delay)
+12 + 10 (1) (AND line delay)
+ 8 (OR line delay)
= 48

The total PLA delay = Delay of producing $f_{1}f_{2}$ + Delay of product $f$ after $f_{1}f_{2} = 68+48=116$
3. Divide & Conquer

(a) Show the D&C tree for designing an 8-bit NOR function using only 2-input NOR, NAND, AND and OR gates (you may not require all gate types). Note also that you are not to design an 8-bit OR and invert it, as this will use more gates than needed. Provide the rationale for your breakup and stitch-up schemes.

(b) Show the circuit for an 8-bit NOR using the above D&C approach.

(c) Assuming each 2-input gate has a delay of $t_d$, derive the delay of your 8-bit NOR circuit. What will be the delay of a $(2^k)$-bit NOR designed using your D&C approach?

Note that NOR is not an associative function.

Ans. a)

\[
\text{NOR} \left( x_7, x_6, \ldots, x_0 \right) = \overline{x_7 \cdot \overline{x_6 \cdot \ldots \cdot \overline{x_0}}} = \left( \overline{x_7 \cdot x_6 \cdot \ldots \cdot x_0} \right) \cdot \left( \overline{x_3 \cdot x_2 \cdot x_1 \cdot x_0} \right)
\]

\[
= \text{NOR} \left( x_7, x_6, x_3, x_4 \right) \cdot \text{NOR} \left( x_5, x_2, x_1, x_0 \right)
\]

\[
= \text{AND} \left( \text{NOR} \left( x_7, x_6, x_5, x_4 \right), \text{NOR} \left( x_3, x_2, x_1, x_0 \right) \right)
\]

Thus, in general, an $n$-bit NOR can be broken into two $n/2$-bit NOR’s stitched up by a 2-bit AND.
c) For \( n=2^k \)-bit NOR, then \( \log n = k \) levels of gates in the corresponding circuit. Delay = \( k t_d \).