CSEE E6861y **Homework 3** Handout 19
Prof. Steven Nowick February 18, 2016

This homework is due at the *beginning of class* on Thursday, March 3.

NOTE: A correct answer without adequate explanation or derivation will have points deducted. To get full credit, (a) write legibly/type, and (b) show all work (label relevant items, show deriviations, include explanations).

1. (10 points) **Essentials #1: given the set of all primes.** You are given a single-output binary-valued function f, given by the following cover F:

$v\ w\ x\ y\ z$	f
1 1 0 1 -	1
0 1 - - 0	1
1 - 0 - 0	1
1 1 - 1 0	1
1 - 0 1 0	1

   Assume this cover is the ON-set of the function, and that the DC-set is empty. You are to determine if the cube, $vwyz'$, listed as the second-to-last row in this PLA file, is *essential.*

   Follow the "containment" approach outlined in Sections 4 and 5, and the later complete example, in Handout #12. Once you formulate the problem as a tautology-check problem, you do *not* need to work through the recursion tree. Simply present the formulation, and give the final answer by inspection with a clear justification.

2. (10 points) **Essentials #1: given the set of all primes.** In Handout #12 (sections 3-5, and the written example), it was shown that if one is given the set of *all primes* P, one can determine the set of essential primes E by the following the formula:

$$E = \{c \in P : c \not\leq (P - \{c\})\}$$

   Effectively, this formula treats the problem of whether a prime c is essential, as a "containment" problem using the cofactor and tautology operations.

   One limitation of the above formula is that **it ignores don't-cares:** it only gives a correct answer if the DC-set is empty. If the DC-set is not empty, the above formula will classify a prime c as "essential" if it is the only one to cover certain DC-set minterms. However, by our definitions, this is not essential: an "essential" must cover at least one ON-set 'distinguished minterm' covered by no other prime.

   In this problem, you are to **fix the above formula,** to accurately define essential primes when the DC-set is *not empty*.

   What to Do: Given a function f, its set of primes P, and a DC-set F^{DC}, (i) *rewrite* the above formula to correctly produce the set of essentials, now properly taking into account the don't cares; (ii) explain in 3-4 sentences what the new algorithm is for finding the essential primes, that is, translate your new formula from (i) into a step-by-step procedure for finding the essentials.

3. (10 points) **Fast Complementation.** You are given the following cover F for a single-output binary-valued function f:

$a\ b\ c\ d$	f
0 - - 0	1
1 0 1 -	1
1 1 1 0	1

Assume that cover F contains both the ON-set and DC-set of the given function f. For this problem you are to compute a cover F' for the complement of the function, which is the OFF-set of the function, following the procedure described in Handout #15, and as illustrated in the example on Handout #12. Follow these handouts closely, using: (i) Basic Rules for Termination (B1-B4) (whenever possible); (ii) Recursive Complementation Theorem to guide the recursion; and (iii) Rule U1 to simplify recursion. However, you do not need to use the "cube merger" optimization.

For choice of splitting variable, follow the guidelines in Handout #11 (Tautology Handout). *Show your work for each step:* clearly draw the resulting recursion tree, indicating splitting all variables, and showing the results derived and returned at each step.

4. (15 points) **Fast Prime Generation.** You are given a single-output binary-valued function f, given by the following cover F:

$v\ w\ x\ y\ z$	f
1 1 0 1 -	1
0 1 - - 0	1
1 - 0 1 0	1
1 1 - 1 0	1
1 - 0 - 0	1

Assume the function has only ON-set (described by the PLA file above) and OFF-set, and that the DC-set is empty.

For this problem you are to compute the set of all prime implicants of the function. Follow the approach in Handouts #17 and #18. Use: (i) Basic Rules for Termination (whenever possible), (ii) Unate Rule for Termination (whenever possible), (iii) Prime Consensus Theorem to guide the recursion, (iv) SCC (single-cube containment) operation to eliminate non-primes. To select a splitting variable, follow the guidelines on Handout #11 (Tautology Handout).

Show all your work: clearly draw the resulting recursion tree, indicating splitting all variables, and labelling the results derived and returned at each step. Once completed, *also write out algebraically the list of final primes*.

5. (15 points) **Essentials #2: not given the set of all primes.** You are given the following cover of a 4-input single-output function f:

$$f(a,b,c,d) = a'b + ab' + bc'd' + a'c'd + acd$$

This cover consists of 5 prime implicants. Note that *it does not contain all prime implicants of f!* Assume that f only has an ON-set and OFF-set; the DC-set is empty. You are to use the technique presented in class and assigned reading to evaluate if two of the primes are essential.

What to Do: (a) Draw the K-map of the function with the given cover F; (b) set up the problem formulation for evaluating if *two of the primes (ab' and $a'c'd$) are essential* (see below for details).

In particular, for part (b), you should use the results presented in class to transform this problem into a *tautology checking problem*. Simply *show all the steps* to take the initial problem and recast it as a tautology checking problem on a particular new cover, which you should derive and write out in PLA representation. Once you have formulated the problem as a tautology check, **you can stop:** *do not solve this problem using the tautology check algorithm!* Simply *indicate the answer (if either of the above 2 primes (ab' and $a'c'd$) is essential by inspection.* Also, once you are done, you can check the cover drawn in the corresponding K-map to confirm your results. *Show all work.*

6. (15 points) **Espresso: EXPAND/IRRED/REDUCE Steps.** In this problem, you are to manually perform the key 3 steps of espresso, iterating on a given cover until an optimal solution is reached.

   Note: This problem should not be a lot of work! However, it does require care in performing each step accurately. One of the goals is that you observe the iterative transformations (i.e. morphing) of one implementation into another as the algorithm is applied, converging stepwise from a suboptimal to an optimal solution. Refer to the class assigned reading and slides for this problem.

   The starting point is an incompletely-specified 3-output function, given by the following minterm expansions:

   $f_1(x, y, z) = \Sigma m(5, 7) + \Sigma d(3, 4, 6)$

   $f_2(x, y, z) = \Sigma m(0, 1, 5) + \Sigma d(3, 7)$

   $f_3(x, y, z) = \Sigma m(0, 1, 3, 7 + \Sigma d(4)$

   Assume your starting point is a cover F for the above multi-output function, that has been derived through an earlier partial run of espresso (i.e. assume several steps have been run, and F is the current solution, but is not yet optimal):

$x\ y\ z$	$f_1\ f_2\ f_3$
1 - -	1 0 0
- 0 1	0 1 0
0 0 0	0 1 0
0 0 0	0 0 1
0 - 1	0 0 1
1 1 1	0 0 1

   Using the above F as a starting point of this problem, you are to iterate the three key steps of espresso in order — *expand, irredundant, reduce* — to reach a minimum-cost solution. At each step, as in class, you will be drawing 3 different "views" of the cover — in a hypercube, as a digital circuit, and as a PLA file representation —, where in each you will highlight the changes produced by the step.

   Note: Follow the directions below carefully; they give you guidelines on assumptions for performing each step, such as on cube ordering, etc. Also, refer not only to assigned reading, but also to class lecture slides (available online) and your lecture notes, for details of steps.

   (a) **Given Cover F.** Given the above 3-output function, and corresponding given cover F, *neatly draw* three different views: (i) *hypercube representation,* clearly showing the 3-output function, annotated with ON-set minterms (black dots) and DC-set minterms (X's) (you can omit the OFF-set minterms to avoid clutter), as well as the implicants of the given multi-output cover F clearly marked; (ii) *multi-output 2-level digital circuit,* showing the AND-OR circuit for this 3-output function based on the given cover F, properly indicating all sharing of AND gates; and (iii) *multi-output PLA file for F,* which is given to you above, so simply copied here.

   (b) **EXPAND Step (#1).** You are now to perform the expand step on the given cover F of part(a). To avoid a lot of extra calculation, I am simplifying some of the steps, but others will still be done rigorously as presented in class. *Be sure to follow the directions carefully.*

   Cube Ordering. In class, we covered a systematic algorithm to weight the cubes, and determine their final order for expansion. Here, *ignore* that algorithm. Instead, expand the cubes *in the order they appear in the given PLA file above,* from top to bottom.

   Expansion Direction. In class, we covered a systematic approach to determine the direction to expand each cube. For a given cube, there are 3 steps to follow in order: (i) expansion to fully-contain the most other cubes; (ii) further expansion (1 variable at a time) to partially-overlap the most possible other cubes; and (iii) further expansion into a maximal prime. For this problem, for each cube in order, *follow these 3 steps in order, but do them by inspection!*. That is, unless you want, you do not need to write out all the details of each step. Just do the right thing, but by examination.

What to Show: Once you have completed the above expand step, neatly show 3 representations of the resulting new cover (as you did in part(a) above, following the same directions): (i) hypercube representation, (ii) 2-level representation, and (iii) PLA file representation. *Clearly mark the differences from the corresponding representations in part(a).*

(c) **IRRED Step (#1).** You are now to perform the irredundant step on the resulting cover of part(b). You do not need to draw out the detailed prime implicant table for this step – you can solve the problem by inspection. However, you must produce the optimal irredundant solution.

What to Show: Once you have completed the irredundant step, neatly show 3 representations of the resulting new cover (as you did in part(b) above, following the same directions): (i) hypercube representation, (ii) 2-level representation, and (iii) PLA file representation. *Clearly mark the differences from the corresponding representations in part(b).*

(d) **REDUCE Step (#1).** You are now to perform the reduce step on the resulting cover of part(c). For "cube ordering", use the order of the cubes in the PLA file of part(c), from top to bottom.

What to Show: Once you have completed the reduce step, neatly show 3 representations of the resulting new cover (as you did in part(c) above, following the same directions): (i) hypercube representation, (ii) 2-level representation, and (iii) PLA file representation. *Clearly mark the differences from the corresponding representations in part(c).*

(e) **EXPAND (#2).** Repeat part(b), starting with the resulting cover from part(d), and show the results in the three representations.

(f) **IRRED (#2).** Repeat part(c), starting with the resulting cover from part(e), and show the results in the three representations.

(g) **"MAKE-SPARSE".** You will now apply a simplified version of the "make-sparse" step to reduce the cost of your final multi-output cover, if possible.

What to Do: In particular, examine each AND-gate (i.e. implicant) which is shared by two or more of the outputs (i.e. is connected to two or more OR gates). If any of the connections (i.e. wire fanouts) of such an AND-gate to an OR gate can be removed, and still result in a valid cover, then remove such a connection. Follow this simple procedure on all such AND-gates. Follow this 'reduce-output-part' step by any possible 'expand-input-part' step, as elaborated on Lecture #5 slides.

What to Show: Once you have completed this simplified make-sparse step, neatly show 3 representations of the resulting new cover (as you did in part(f) above, following the same directions): (i) hypercube representation, (ii) 2-level representation, and (iii) PLA file representation. *Clearly mark the differences from the corresponding representations in part(f).*

7. (25 points) **Unate Recursive Paradigm: Boolean AND.**

In this problem, you are to use your experience with unate recursive algorithms to design a new algorithm: to compute the *Boolean-AND* of two Boolean functions, f and g.

Overview: In tautology-checking and complementation, a unate recursive paradigm is used. Each algorithm begins with a cover F of a function f. A divide-and-conquer approach is then used, where the cover is split recursively using splitting variables in some order. The recursion terminates when certain termination conditions are satisfied. These termination conditions are different for each of the above algorithms. The results are then combined together into a solution. In this homework problem, you will also return a single cover, but your starting point will now be *two covers*, F and G, not one cover.

Review: Boolean AND. Given two (single-output) Boolean functions f and g, each having the same input variables (i.e. same size K-maps), the *Boolean AND,* $f \cdot g$, is defined as 1 at each minterm where functions f and g are both 1. For all other minterms, the resulting Boolean AND is 0.

Assumption: Assume that both f and g are fully-specified functions, i.e. have no don't cares (DC-set is empty).

What To Do: As usual, you will not be given functions, but rather *covers* F and G (for functions f and g, respectively). Your algorithm will take F and G, and eventually return a single *cover* H for the ON-set of the Boolean AND $f \cdot g$. Sketch clearly and precisely a unate recursive algorithm to compute this Boolean AND of two functions, f and g.

In particular, your answer should be concrete and clear, addressing the following issues.

(i) Give a short overview (1-2 paragraphs) of your proposed approach.

(ii) Briefly, how does algorithm handle two covers, instead of one? (how is splitting performed, how are results assembled?)

(iii) What termination rules do you propose? (these can be simple, but must still allow early termination). For each termination condition, what result is returned? You should try to find a varied and powerful set of termination conditions, including simple basic ones, as well as more sophisticated ones.

Note: your conditions should be easy to apply, and should improve the search rather than making it more complicated or slower. Follow the guidelines of the existing algorithms, and see if you can also come up with new conditions, too.

(iv) What form of Shannon decomposition will you use? Write out the new Shannon decomposition equation for returning the resulting cover H (i.e. Boolean-AND result), given the two initial covers F and G.

(v) Can you exploit unateness properties in (a) forming termination rules, or in (b) directing and simplifying the search? (By analogy, for tautology check, note that the unateness condition resulted in Rule U1 for (a), and for Rule U2 for (b); see if you can come up with analogous conditions for (a) and (b) for Boolean AND.)

(vi) What criteria do you propose to select a good splitting variable, given that the algorithm handles two covers instead of one?

Note: If you provide a solid basic solution (which is correct and not trivial), which is well-documented, you will obtain 15 points. The remaining points will be awarded for more creative and effective solutions.