Logic Synthesis & Optimization

Spring 2019, Homework # 3
Due Date: Tuesday, March 5, 2019
Deadline for Tuesday March 5, 11:59pm, is strict!

This HW corresponds to concepts given in Chapters 4 and 5 in the textbook, and on Simple disjunctive decomposition and Ashenhurst-Curtis decomposition given in the slides on the class website. You should read those chapters completely to be able to solve these questions.

1) **10 points - (Prime implicant computation):** Consider the Boolean function $f = ab'c' + a'bc' + a'bc + abc'$.
   a) You are asked to use the recursive approach to computing prime implicants that we studied in class. Write the formula for computing the primes.
   b) As you will employ a *recursive approach* to Shannon's expansion-based prime computation, list the *terminal cases of this recursion*, i.e. when does recursion bottom-out?
   c) Using the order $a \prec b \prec c$ for performing the Shannon's expansion, generate all primes. Show ALL your steps clearly. If a cover is unate at any recursion level, be sure to exploit unateness for efficiency.

2) **20 points - (Table covering, branch and bound method):** Solve Problem 15 in Chapter 4 of the textbook. *In the textbook, rows are minterms and columns are primes.*

3) **10 points - Unate Recursive Paradigm:** Let $f(w, x, y, z)$ be a Boolean function whose implicant cover is given as $F = \{wxy, wyz', w'x'y', w'z, xy', w'y\}$. Is f a TAUTOLOGY? Describe your problem formulation and solve it using the unate recursive paradigm.

4) **25 points - (Logic manipulation using the Unate Recursive Paradigm):** Consider a completely specified Boolean function $f(w, x, y, z)$ whose cover is given as $F = \{w'xy'z, wxy'z, w'xyz, wxy\}$. Without using any prime generation or table covering approaches of two-level logic optimization, formulate and solve the following:
   a) Is xz an implicant of f?
   b) Is xz a prime implicant of f?

c) Is xz an essential prime implicant of f?

Of course, you have to use the elements of the unate recursive paradigm: So state your formulations clearly, and show your work.

5) **10 points - (Another small experiment with Espresso):** All of you have already downloaded and played with the Espresso program. In this simple experiment, you will see that the (heuristic) logic optimization problem is quite often related to making a two-level cover prime and irredundant. A short description of espresso's options (a short manual) can be found through the "tools" link on the class website. At the end of Chapter 5 of the text-book, you will find some examples that will give you a feel for Espresso, and the "kiss" format for describing single/multi-output PLAs. (Note: In the kiss format, .i stands for # of inputs; .o for # outputs; .ilb for input label (var names); .p for # of product terms; .e for end of file). I want you to run some experiments with espresso. Attach a print-out of your experiments, and a brief description of what you deduce from the experiment.

   a) Open a file, call it "cyclic.kiss".
   b) In this file, enter the cover (in espresso format) of a 3-variable function that exhibits the *worst-case "cyclic core"* of the two-level minimization problem. You should be able to create such a problem yourself. Of course, such a problem is also given in the textbook, but you will have to read the textbook to find it. *Make sure to write the cover using all the minterms of the function.*
   c) Run the command: 'espresso -Dreduce cyclic.kiss'. Verify that minterms cannot be "reduced".
   d) Run the command: 'espresso -Dprimes cyclic.kiss'. Verify whether espresso has computed all the primes of the function. You can save the output by redirecting it into another file: 'espresso -Dprimes cyclic.kiss $>$ cyclic_primes.kiss'.
   e) Can the cover of 'cyclic_primes.kiss' be expanded further? Verify your answer by running: 'espresso -Dexpand cyclic_primes.kiss'.
   f) Making this cover irredundant? Run: 'espresso -Dirred cyclic_primes.kiss'.
   g) The exact minimized cover can also be computed by running: 'espresso -Dexact cyclic.kiss'. What happens when you run: 'espresso cyclic.kiss', i.e. without the -Dexact option?

6) **15 points - (Simple Disjunctive Decomp):** Consider the Boolean Function f specified by the following minterms: $f(w,x,y,z) = \Sigma(1,3,5,7,8,11,13,15)$. Identify **all feasible simple disjunctive decompositions** in which the number of support variables in the **bound set** is equal to two. For each feasible decomposition, derive and show clearly the functions corresponding to the bound-set

and free-set variables. Feel free to use either the decomposition chart (K-map) method or use BDDs - your choice.

7) **10 points - (Ashenhurst-Curtis Decomposition):** Consider the Boolean function $f(a, b, c, d) = abd + a'bc' + a'b'c$. Represent the BDD of f with $a \prec b \prec c \prec d$ as the ordering of variables. Using a, b as the bound set, obtain a decomposition for f. Show all the decomposed sub-functions. Does your decomposition produce any *don't cares*? If so, do they lead to any logic simplification?