Logic Synthesis & Optimization

Spring 2019, Homework # 2: A programming assignment with BDDs
Due Date: Monday, Feb 18, 2019

1) (**30 points**) In class we studied the concept of generalized cofactors. Given Boolean functions f, g, we have $f = gf_g + g'f_{g'}$, where $f_g, f_{g'}$ are the generalized cofactors. They are not unique, but they satisfy the following bounds: $f \cdot g \subseteq f_g \subseteq f + g'$. You are asked to derive these bounds. Note that proving the validity of these bounds is easy ($p \subseteq q \implies p' + q = 1$). Can you start from "first principles," e.g. Shannon's expansion, or use Venn diagrams, etc., and try to derive these bounds?
2) (**20 points**) Let $f = w'x'z' + wx'z + w'yz + wyz'$ and let $g = w \oplus z$. Can you identify generalized cofactors of f w.r.t. g, perform a decomposition and synthesize a circuit, as shown in the class notes on the webpage? Is your decomposition "cheaper" than a two-level logic implementation of f?
3) **Some Programming with the BDD package (50 points):** In this assignment, you are asked to write some simple programs using Prof. Fabio Somenzi's CUDD package. First of all, on the class website, I've provided a link to Fabio's webpage. From there you can download the CUDD package. Also, there is a link to online documentation that will be helpful. For the setup, do the following:
   - Log in to one of CADE machines, which supports x86-64 bit instruction set. Try compiling and running the CUDD package.
   - Download and untar the package in a specific directory. Go through the README file. You will also find directories such as, cudd, nanotrav, sis, st, util, etc.
   - Enter the ./cudd directory and just glance through the cudd.h and cuddInt.h files. These are header files that declare the data structures: DdNode, DdManager, DdChildren. Glance through these data-structures. You will extensively use pointers to DdManager and DdNode structures. You can also go through the #define macros declared.
   - Now just list all the *.c files in the cudd directory: 'ls -al *.c'. **NOTE:** the CUDD package implements BDDs, ADDs (called Algebraic Decision Diagrams) as well as ZDDs (called Zero-Suppressed BDDs). You should just ignore ADD and ZDD related files and routines; just concentrate on BDD related files. The file names are self explanatory. For example, cuddCof.c contains routines for performing cofactor computations. You don't need to study the routines - just get introduced to the package.
   - Now go through the nanotrav directory. Go through the README file. Also, glance through the "main.c" file. You may not understand much, but don't worry. We will overwrite this main.c

file with (y)our own somewhat simpler “main.c” file.

- Come back to the top directory. Go through the Makefile again. Just type ’make’ and the package will be compiled: libraries will be created and linked and an executable named ’nanotrav’ will be created in the ./nanotrav directory. Just run a test program as given in the ./nanotrav/README file. For example, type ’nanotrav -p 1 -cover C17.blif $>$ C17.out’ just to check that the program compiled properly and is indeed running fine.
- Now the ./nanotrav/main.c file is too complicated. I’ve written a simple program that: i) initializes a BDD manager; ii) creates variables; iii) performs some simple ITE computation; iv) prints out the resulting BDD, by declaring proper variables and calling proper sub-routines. This file, again called ’main.c’ is also uploaded on the class website - within the ’Homeworks’ section. Download this file, go through it properly. Replace ./nanotrav/main.c by this file ’main.c’ and compile and just run the program. Relate the output to the code in this file. **Note:** the CUDD documentation (online) or the .ps file in the cudd-2.4.1/cudd/doc/doc.ps will tell you all about these routines. In the .ps file, the index from page 43 onwards gives a list of all the BDD manipulation routines that you can use.
- **Now here are the assignments:** In this main.c file, you will write code that does the following:
  - Create ROBDD for $f = ab + ac + bc$. Use any variable order.
  - Write code that tests whether the function is positive unate or negative unate in variable b.
  - Repeat the above test for $f = a \oplus b \oplus c$, three input xor function.
  - Print the outputs properly. Try to use the Cudd_DumpDot() routine and try to use the ’dot’ package to print the BDD that you build or want to show. The ’dot’ package allows you to draw graphs. If you have never used this package before, you can get it from http://www.graphviz.org.
  - You can also use the BDD package to perform the *generalized co-factor* computations. [We will study these in class when we cover Boolean function decomposition.] The routines are given in file ’cuddGenCof.c’. Take a look at the *description* of the following subroutines: `i) bddRestrict, ii) bddConstrain, iii) bddSqueeze, iv) bddLICompaction`, etc. Using these routines, you are asked to do the following:
    - Given f, g in Question 2 above, use the BDD package’s various routines to compute f_g and $f_{g'}$.
    - It turns out that the *generalized co-factors are not unique* but agree with the bounds: $f \cdot g \subseteq f_g \subseteq f + g'$. Using the BDD printing routines, print out the functions, $f, g, f_g, f_{\overline{g}}$. Also, *using a suitable formulation*, test whether the bounds are satisfied for the above

functions.

– **Turn in** your code and the program output. Preserve your code. I may test it myself. Happy programming and good luck!