Logic Synthesis & Verification, Fall 2017

National Taiwan University

Programming Assignment 1

Due on 2017/10/15 23:59 (by online submission to CEIBA)

1 [Using ABC]

(10%)

(a) Use BLIF manual (`http://www.eecs.berkeley.edu/~alanmi/publications/other/blif.pdf`) to create a BLIF file representing a four-bit adder.

(b) Perform the following steps to practice using ABC (`http://www.eecs.berkeley.edu/~alanmi/abc/`):

1. read the BLIF file into ABC (command "`read`")
2. check statistics (command "`print_stats`")
3. visualize the network structure (command "`show`")
4. convert to AIG (command "`strash`")
5. visualize the AIG (command "`show`")
6. convert to BDD (command "`collapse`")
7. visualize the BDD (command "`show_bdd`"; note that `show_bdd` only shows the first PO; command "`cone`" can be applied in combination to show other POs)

Items to turn in:

1. The BLIF file.
2. Screenshot of your ABC execution steps.
3. Results of "`show`" and "`show_bdd`".

Comment 1: For commands "`show`" and "`show_bdd`" to work, please download the binary of software "dot" from GraphViz webpage (`http://www.graphviz.org`) and put it in the same directory as the ABC binary or anywhere else in the path.

Comment 2: Make sure GSview and Ghostscript are installed on your computer. (`http://pages.cs.wisc.edu/~ghost/gsview/`) A proper path of `gsview32.exe` needs to be specified in "`\src\base\abc\abcShow.c`." For Windows 7, the path is likely to be `C:\Program Files (x86)\Ghostgum\gsview\gsview32.exe`.

2 [ABC Boolean Function Representations]

(10%) In ABC there are different ways to represent Boolean functions.

(a) Please compare the following differences.
   1. logic network in AIG (by command "`aig`") vs. structurally hashed AIG (by command "`srtash`")
   2. logic network in BDD (by command "`bdd`") vs. collapsed BDD (by command "`collapse`")

(b) Given a structurally hashed AIG, please find a sequence of ABC commands to covert it to a logic network in SOP.

3 [Programming ABC]

(80%) Write a procedure in ABC to find hidden majority (MAJ) gates in a given AIG. Integrate your MAJ identification procedure into ABC, so that running command "`MAJ_find`" (defined by function named "`Abc_CommandMajFind`") would invoke your code to report the found MAJ gates. To fix the notation, let $f = \text{MAJ}(a, b, c)$, where f is the output, and a, b, c are the inputs. So the command output may list, e.g., $25 = \text{MAJ}(5, -14, 22)$, for AIG node (ID) 25 being the output with inputs 5, negated 14, and 22. Also it should summarize the total number of the found MAJ gates at the end.

Note that `Abc_NtkStrash` can be used to convert a logic network to an AIG. Refer to the code of command `strash` in `abc.c` for its usage. The instructions about how to add your code to ABC can be found in `PA1_README.txt`. Please do NOT modify any ABC code to make your program stand alone.

Programming help:

Example of code to iterate over the objects

```
void Abc_NtkCleanCopy( Abc_Ntk_t * pNtk )
{
  Abc_Obj_t * pObj;
  int i;
  Abc_NtkForEachObj( pNtk, pObj, i )
    pObj->pCopy = NULL;
}
```

Items to turn in:

1. Your code (not the whole ABC package) and a readme file for special compilation instruction, if there is any.