系統晶片驗證 (SoC Verification)

105 學年下學期 電機系電子所選修課程 943 U0250

Homework #2 [Formal Verification Basics && BDD-Based Verification]

(Due: 9:00pm, Thursday, April 06, 2017)

[Guidance for turning in your homework]

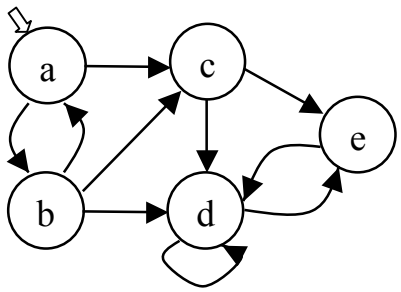
- Turn in your homework through Ceiba only. (i.e. NO hardcopy)
- For problems 1-5, you can choose to: (1) answer the problems in paper and sheet, and then photo or scan it as image file, or (2) use any text editor to answer the problems, and then convert it to PDF.
- For problems 6-7, name your programs or outputs as specified in the problem descriptions.
- Name your file(s) properly so that we can grade it(them). For example, prob1-5.jpg, prob1-5.pdf, prob3.a.jpg, etc.
- Create a directory named "yourStudentID_hw2" (e.g. b77503057_hw2) and put all the files in this directory. Compress it by:
  tar zcvf yourStudentID_hw2.tgz yourStudentID_hw2
- Submit this single compressed file to Ceiba.

1. Given the Kripke structure below, where 'a', 'b', 'c', 'd' and 'e' are atomic propositions and the initial state is denoted with an arrow. Please verify the correctness of the following temporal formulae by the explicit modeling checking technique. Explain why.

   (a) AGF (a → X d)

   (b) EG (b → AF d)

   (c) EFAG !c

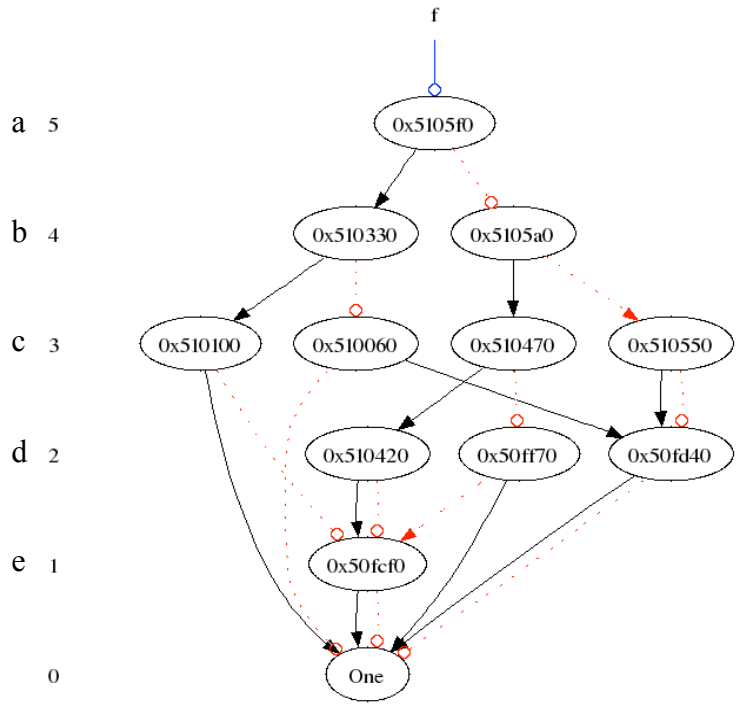


2. Given the following BDD *f* ---

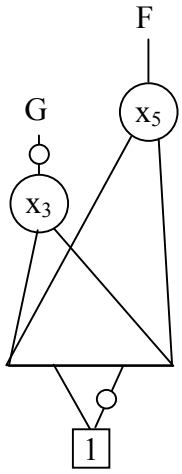
   (a) What are the values of *f* for (a, b, c, d, e) = (0, 0, 0, 0, 0), (1, 0, 1, 0, 1), and (0, 1, 0, 1, 0) ?

   (b) List all the cubes for "*f* = 0" in terms of input variables { a, b, c, d, e } (e.g. "10xxx"). Please list the cubes in ascending order (e.g. 00000, 00001, 0000x)

(c) Covert this BDD *f* to a BDD without complemented edges. Just draw the BDD nodes with input labels { a, b, c, d, e }, not the pointer addresses, and you can ignore the edges to constant '0', such as the BDDs in p18, lecture note #3.

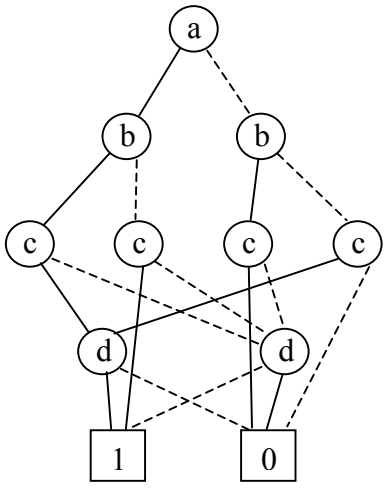


3. Let F, G be two BDD nodes, where G has the smaller (i.e. lower) top variable index than F. In addition, G has complemented edge while F has not (as shown below).



Let R = ITE(F, 0, G). Please use the rules in the lecture notes to standardize this ITE call for the entry of the computed cache.

4. Given the BDD below, perform dynamic variable reordering for ‘b’ and ‘c’ and draw the final BDD. What is the difference in the number of nodes?



5. Draw the *BMD for the formula $X^2 + Y^2$, where $X = x_1x_0$ and $Y = y_1y_0$ are two-bit variables with $x_1(y_1)$ and $x_0(y_0)$ being the MSB and LSB of X(Y), respectively. Use the variable order: { x_1, x_0, y_1, y_0 } where x_1 is the top variable.

- Please download a BDD package for the following problems:

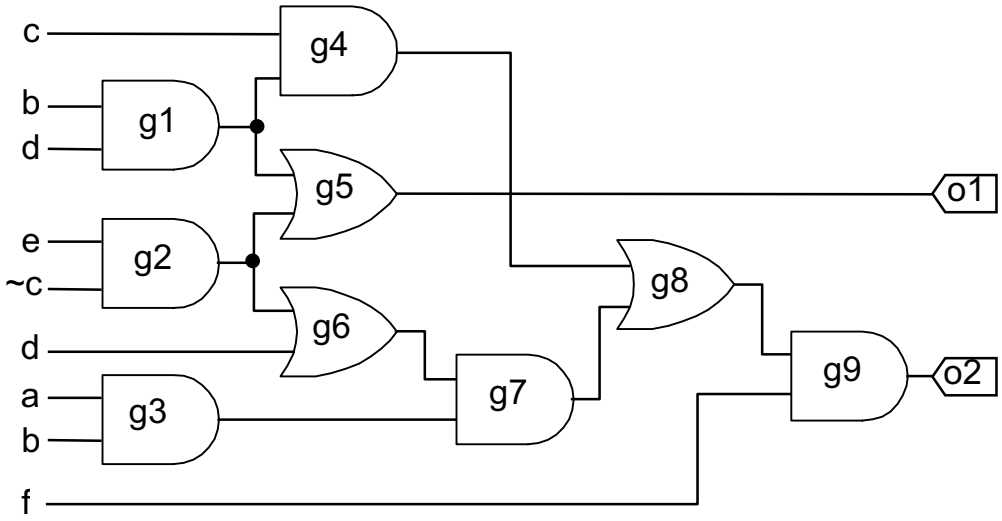
  URL: https://github.com/ric2k1/RicBDD

  To clone in your terminal, type: git clone https://github.com/ric2k1/RicBDD.git

  In case the download is blocked by firewall, try to download the zip file instead.

  (Optional) To draw the BDD to an image file, you need to download the "graphviz" from http://www.graphviz.org/Download.php. It will install the executable "dot" which is needed by our BDD drawing program. However, if the site is not alive anymore, you can use its web version: http://www.webgraphviz.com/, or find the source code from some other open source repository.

6. In this problem, we will perform the "redundancy" check and compute the Boolean difference for the circuit below:



(a) Rename "testBdd.cpp" to "p6a.cpp" to compute the BDDs for gates g1, g2,… to g9. Use "cout" to print our the BDDs of the above gates and name the output file "p6a.out" (i.e. ./testBdd > p6a.out). Turn in "p6a.cpp" and "p6a.out". (note: copy these two files to "yourStudentID_hw2" directory)

(b) Adding a wire from g5 to the input of g9. Repeat problem 6(a) but rename the files to "p6b.cpp" and "p6.out", accordingly. Check that the BDDs of the old and new g9's are functionally equivalent. Turn in "p6b.cpp" and "p6b.out".

(c) The Boolean difference of f with respect to g is denoted as:

- Diff($f|_g$) $\equiv df/dg$

  $\equiv f(g = 1) \oplus f(g = 0)$, where $\oplus$ is an XOR operator

Compute the BDD for Diff($g9|_{g5}$) on the new circuit (i.e the one with the added g5 → g9 wire). Similar to (a) and (b), please turn in files as "p6c.cpp" and "p6c.out".

(d) Since the newly added wire is redundant, explain why or why not the Diff() BDD in (c) is a constant '0'? Try to use BDD to verify your explanation. For example, if it is a constant '0', show that the Diff() BDD for other non-redundant wire is NOT a constant '0'. Otherwise, if it is NOT a constant '0', point out the missing conditions for the redundancy check and conjunct them with the Diff() function to make a '0' BDD node. Answer this sub-problem with any text editor but convert it to PDF before turning it in. Name the file "p6d.pdf".

7. "Restrict" operator and "Witness-BDD Generation Algorithm"

(a) Implement the "restrict()" operator as described in p30 of lecture note #4. You can choose to implement as a member function of class BddMgr (i.e. BddNode BddMgr::restrict(const BddNode& f, BddNode &g)), or of BddNode (i.e. BddNode BddNode::restrict(const BddNode &g) const). Turn in your revised files as "p7a.cpp" and/or "p7a.h".

(b) Given the pseudo code of "Witness-BDD Generation Algorithm", we would like to construct the BDD for the output assignment "o2 = 1" of the circuit in problem 6 by calling BuildBDD(o2, 1). Please use the "restrict()" function in (a) in your BDD construction. Please show the detail steps of how the BDDs are built, and compare the resultant BDD size (i.e. number of BDD nodes) with that of g9 in problem 6-(a). Answer this sub-problem with any text editor and convert it to PDF before turning it in. Name the file "p7b.pdf".

```
// Assume gate has only two fanins
BuildBDD(gate, value) {
   if (BDD for "gate=value" has been built)
      return hashedBDD;
   if (value is output controlling value) {
      for_each_fanin(fanin, faninV)
         BuildBDD(fanin, faninV);
      ConstructBDD(type, gate, faninBDDs);
   }
   else { // non-controlling
      return WitnessBDD(gate, value);
   }
}
WitnessBDD(gate, value) {
   for_each_fanin(i, fanin, faninV) {
      Let A = BuildBDD(fanin_i, witnessV);
      Let B = BuildBDD(otherFanin, careSpaceV);
      if (Restrict(A, B) != 0)
         return Restrict(A, B);
      // else try the next fanin
   }
}
```