系統晶片驗證 (SoC Verification)

110 學年下學期 電機系電子所選修課程 943 U0250

Homework #1 [Getting Familiar with V3 Framework]

(Due: 9:00pm, Thursday, March 10, 2022)

[Objectives]

1. Getting familiar with the verification framework "V3" (developed by Design Verification Lab in NTU).

2. Writing monitors (assertions) and performing simulation-based verification on a provided RTL design.

[Disclaimers]

1. We only support Linux platform. Please compile and test the homework on Linux platform only. Although it is possible to compile it in other platforms, we test and grade your homework on a Linux environment. It will be of fewer problems if you implement and test your program on the same platform.

2. Currently V3 can be compiled with "-std=c++98" only. Although it is OK to compile it with c++11 and above, there is an unsolvable compile error in "include/map.h" and we cannot guarantee if there will be no side effect.

[Problems]

1. [Installing V3]

   In your Linux terminal, decompress "hw1.tgz" by "tar zxvf hw1.tgz". Rename "hw1" directory to "yourStudentID_hw1" first. Follow the instructions in "V3_readme" to install V3. You should be able to clone the V3 source code from bitbucket, install boolector and minisat solvers, install quteRTL front-end, and then compile V3. You may see some warning message at the end of compilation, such as:

   ```
   (.text+0x3f69): warning: Using 'getpwent' in statically linked
   applications requires at runtime the shared libraries from the
   glibc version used for linking
   ```

   Just ignore them if you don't see any error message.

   Upon the success of compilation, just "cd V3" and type "./v3", and you should see V3 running (with "v3> " command prompt). Type "help" to see if it works properly.

2. [Read and Write Designs Tutorial]

   Go through the "*Read and Write Designs*" tutorial in Chapter 1 of "tutorials/V3_tutorial.pdf" and repeat it for the "*vending.v*" design under the "*vending-simple*" directory. In essence, do the following:

   (a) Read the spec of the design "SPEC-vedning machine.pdf" to understand architecture and design of the vending machine.

   (b) Read in the original RTL design and write it out into "rtl", "btor", and "aig" formats. Name them as "*vending-rtl.v*", "*vending.btor*", and "*vending.aig*" and put them in the same (i.e. "*vending-simple*") directory.

   (c) Report the network statistics by the command "`print ntk -verbose`" for the three output designs in (a). Put the output message into a file "*hw1.2c.pdf*" under the "*reports*" directory.

   (d) Plot the "rtl" and "aig" networks by the command "`plot ntk -level k -png`". Choose the proper "`k`" value so that the entire network can be plotted. Name the output files "*plot.vending-rtl.png*" and "*plot.vending-aig.png*" and put them in the "*reports*" directory.

3. [Design Simulation Tutorial]

   Go through the "*Design Simulation*" tutorial in Chapter 2 of "tutorials/V3_tutorial.pdf". Please note that the vending machine design inside this tutorial is the original (more complicated) one. In this problem, you are asked to perform simulation for the simplified one (i.e. "*vending.v*" under the "*vending-simple*" directory).

   Please note there is at least one bug in the design and the monitor "*p*" in line 112 is enough to reveal it. This monitor signifies the bug that when there is no transaction or the transaction fails (e.g. not enough changes), the output changes are NOT equal to the input changes. The input pattern file "*input.pattern*" is for your reference. You should try to generate more test patterns to reveal the bug and figure out the cause(s). Please refer to Appendix B of V3's tutorial for the format of the simulation pattern, and please pay attention to the order of the input variables in the pattern file.

   Create a subdirectory "*debug*" under the "*vending-simple*" directory, and put your input pattern file(s) and output log(s) in it. You should also include a file "*hw1.3.pdf*" to describe how you find the bug and fix it.

4. Save the rectified design as "*vending-fixed.v*" in the "*vending-simple*" directory. In order to verify the design, you should write more "monitors" to guard the potential bugs and generate more patterns to check if any of the monitors can be triggered. If another bug is found, you need to debug. Note that sometimes it is

not a bug in the design, but a bug in the monitors you write. Repeat this process (i.e. writing more monitors and creating more patterns) until you think the verification quality is good enough.

Since the manual pattern creation process is very tedious, you are encouraged to write some "driver" program to automatically generate legal patterns for simulation.

Put your input pattern file(s) and output log(s) in the subdirectory "*verify*" under "*vending-simple*". You should also write a report "*hw1.4.pdf*" in the same directory to describe: (1) your monitors and how sound you think they can guard the design, and (2) the test patterns and their verification results.

[Submission of homework]

Please name the files and place them in directories properly or as suggested by the problem description above. Rename the root directory "*hw1*" as "*yourStudentID_hw1*" and compress it (MUST be under Linux/OSX workstation) by the command:

"`tar zcvf yourStudentID_hw1.tgz yourStudentID_hw1`".

Submit the .tgz file only. Failure to abide by the above rules may result in deduction of your HW grade.