Logic CAD of VLSI Design - 0460880

Topic: SAT-based Gate-Level Formal Equivalence Checker

Introduction

Formal Equivalence Verification (FEV) provides a key technology that enables comparing the RTL to the synthesized gate-level implementation, or comparing multiple different gate-level implementations to each other. In recent years, BDDs, which were the cornerstone of formal models and FEV have been replaced by SAT solvers to overcome the capacity limitations of BDDs.

In this assignment

1. You will learn how to use and integrate your code with MiniSat which is an award-winning SAT solver (using the API)
2. You will implement a “Formal Equivalence Verifier (FEV)” which compares two gate-level Verilog circuits.

Detailed Tasks:

1. Implement a “Formal Equivalence Verifier” according to the following:
   a. Read two sets of Verilog files, and analyze them to obtain the primary inputs and outputs (PIs and POs) of the design.
   b. Match the PIs and POs between the two circuits (by name).
   c. Match the DFFs between the two circuits (by name after flattening)
   d. Formulate a CNF formula for a system comparing each pair of outputs and call MiniSat to prove the matching (think about how to handle DFF inputs and outputs).
   e. Use the MiniSat API to generate a SAT problem that encodes the circuit to compare the two Verilog files. The MiniSat solver should print UNSAT if the two circuits are equivalent and SAT with an assignment if they are not. The API (as already provided in the template code) should also write a DIMACS .cnf file.

   To understand how MiniSat API should be used, one can use the code example uploaded to Moodle and the example shown in class, as well as the online documentation of MiniSat. Note that the provided template code includes the creation of a DIMACS file (.cnf) after circuit minimization using the MiniSat API. You should expect that each run of the code will create a DIMACS file.

   Detailed Requirements:
   a. Correctness
   b. When mismatches are found, the code should clearly state the input vector that causes the mismatch (one example for such input, even if there are more).
   c. Keep your code efficient – important.
   d. Keep the code well documented and with good readability.

   [Dry] Provide documentation for your code and answer the next questions in a separate pdf report:
   a. What is the basic code structure/high-level algorithm?
   b. Provide an exact explanation of the comparison algorithm.
   c. How are constants being handled?

d. How are XOR gates being supported? Show the detailed Tseytin development if you use it.

Stage A – Build and write your own programs with HCM

1. **cd HCM**
2. **make** – not needed if you followed the workshop
3. **cd wet02**
4. Write your code in HW2ex1.cc file.
5. **make**
   NOTE: the tests for this exercise are automatic!
   If your submission will not compile on the virtual machine – it will be grated ZERO!

Stage B – Test your own programs with HCM

After finishing Stage A, make sure you run the following command from wet02 directory.

Now you are ready to generate the required output files for self-testing before submitting your work.

To generate TopLevel1355.cnf and TopLevel0409.cnf files run the following lines in the Virtual machine, in "wet02" directory.

- **./gl_verilog_fev -s TopLevel1355 stdcell.v c1355.v -i TopLevel1356 stdcell.v c1356.v**
- **./gl_verilog_fev -s TopLevel0409 stdcell.v c0409.v -i TopLevel0410 stdcell.v c0410.v**

You should test your code with **SAT and UNSAT comparison cases** based on the following benchmark circuits: **c1355.v** and **c0409.v** and get **UNSAT** and **SAT** respectively.

Stage C – Create your submission

After finishing Stage B, make sure your wet02 directory include only the .cc files and your final report.
Run the following command from wet02/ directory -

1. **cd ..**
2. **tar czf wet02_<id1>_<id2>.tar.gz wet02**
   (<id1> and <id2> are the id numbers of the students)
3. Upload wet02_<id1>_<id2>.tar.gz to Moodle
   (Example of the file name: wet02_123456789_147852369.tar.gz)

- Make sure that only the .cc your report files are in the folder !

Note: Bold lines are execution commands for the Linux environment (LUX).
Note: The following will be taken into consideration when grading your assignment:
a. The correct modeling of the behavior of the circuit.
b. Clear answers to all dry part questions.
c. The quality of the code and the documentation.

Topic	SAT- based Gate Level Formal Equivalence Checker
Submission date	
Exercise owner	Gilad Zilberman - zilberman@campus.technion.ac.il
Given Code files	**HW2_ex1.cc** **Makefile**
Given Input Verilog files	**c1355.v** **c1356.v** **c0409.v** **c0410.v**
Given Extra Verilog files	stdcell.v
Given Output files for self-testing	-
Files to submit	**HW2_ex1.cc** **HW2.pdf**
Submission format	**wet02_ < id1 > _ < id2 >.tar.gz**
NOTE: the tests for this exercise are automatic! wrong submission format will be grated ZERO!	

Questions about this WET exercise should be posted on Moodle. You required to follow the answers of the course staff and write your code according to the answers.

Good luck!