Jonas Fiala
Trinity Hall
jf613

Part II Project Proposal

Property-based Testing for Hardware

October 2019

Project Originator: *<What to put here?>*

Resources Required: See attached Project Resource Form

Project Supervisor: Matthew Naylor

Signature:

Director of Studies: Simon Moore and Hatice Gunes

Signatures:

Overseers: Pietro Lió and Robert Mullins

Signatures:

<Some comments are included, styled in this fashion, which won't be in the final Proposal!>

Introduction and Description of the Work

Test rigs are common when developing hardware and so many techniques have been developed for automatic testing and generation of test cases. Property-based testing is a useful and now commonplace idea in automatic testing.

There has been much research regarding automatic testing for software design.

QuickCheck implements random testing with a given or user specified distribution [1]

SmallCheck implements exhaustive testing up to some depth [3]

Both are existing testing libraries that encourage lightweight formal specification of software through the reward of automatic testing and small counter-examples. More recently this idea has also been applied to hardware.

BlueCheck implements random testing à la QuickCheck but for a HDL (Bluespec) [4]

Yet few automatic testing libraries have been created for open source HDLs, so this project will propose to bring bounded exhaustive testing to an exciting new Haskell library for hardware description; Blarney (https://github.com/mn416/blarney).

<What else to add? Somehow bring [2] into this?>

Resources Required

Haskell: require ghc version 8.6.1 for Blarney, but the default apt get ghc installs version 8.0.2 (this is what is on MCS) so must download manually.

CL account: can access required version of ghc if necessary, also to use Bluespec/BlueCheck if necessary.

Blarney: downloaded with git clone.

FPGA: for synthesizing testing rigs.

Machines in College: similar to MCS machines, only needed if personal hardware fails.

Personal laptop and computer: -

My laptop: 2.30 GHz CPU, 8 GB RAM, 118 GB Solid-state disk for OS and 447 GB Solid-state disk for data, Windows OS with Ubuntu VM

My computer: 3.50 GHz CPU, 16 GB RAM, 256 GB Solid-state disk for OS and 1 TB Solid-state disk for data, Windows OS with Ubuntu VM

Contingency plans: My contingency plans against data loss are that everything will be held under git on GitHub with daily checkpoints to my Google Drive and also weekly to USB Flash Drive kept only for that purpose. My contingency plans against hardware/software failure are that I can easily transition my work to the MCS/CL machines

Starting Point

I have basic prior knowledge, from taking the **Computer Design** course in **Part IB** and the **Digital Electronics** course in **Part IA** including the practicals for both. Therefore an important part of the project will be learning to use the required tools.

Substance and Structure of the Project

The aim of the project is to build a Haskell library for automatic testing of modules written in Blarney. I would primarily draw inspiration from SmallCheck [3] and apply it to a HDL, this would mean bounded exhaustive testing.

There are many further possibilities beyond the success criteria for this project:

Synthesis of a test rig to an FPGA and compare speeds against a simulated test rig [4]

Sequential logic modules could be supported, testing a bounded sequence of inputs [4]

Random testing similar to QuickCheck and BlueCheck [1][4]

Other verification methods (such as an SMT solver) could be created for Blarney and then compared to the bounded exhaustive testing method *<Cite?>*

References

[1] K. Claessen and J. Hughes. *QuickCheck: A Lightweight Tool for Random Testing of Haskell Programs*, in Proceedings of ICFP'2000, (2000)

[2] K. Claessen. *Embedded Languages for Describing and Verifying Hardware*. PhD thesis, Chalmers University of Technology and Göteborg University (2001)

[3] C. Runciman, M. Naylor, F. Lindblad. *SmallCheck and Lazy SmallCheck: automatic exhaustive testing for small values*, in Haskell Symposium '08. pp. 37–48. ACM (2008)

[4] M. Naylor and S. W. Moore. *A Generic Synthesisable Test Bench*, in MEMOCODE 2015, pp. 128–137 (2015)

Success Criteria

The following should be achieved:

- Implement a Haskell library for automatic testing of combinatorial circuits written in Blarney
- Devise a suite of example buggy circuits
- Measure the proportion of bugs reported and the size of counter-examples found

<Should this be more concrete? Can also include synthesis to FPGA as a success criteria since it will almost certainly be done?>

Timetable and Milestones

25th October to 21st November

*<4 weeks * 20% time (10hrs a week) = 40hrs>*

Research property-based testing for software and also hardware, making sure to fully understand the related work. Setup and test the SmallCheck/QuickCheck libraries for Haskell, to better understand how they work and to improve skills with Haskell. Learn to use Blarney, building some example circuits as a learning exercise.

Milestones: Implement small Blarney modules of combinatorial logic. Design structure of testing framework and type-based generators.

22nd November to 5th December

*<2 weeks * 50% time (25hrs a week) = 50hrs>*

Setup BlueCheck and Bluespec to test synthesizing test benches to an FPGA. Decide with the help of Matt on final design of the testing library, with further literature study if necessary. Create the enumerative generators.

Milestones: Lay out structure necessary to meet the success criteria and be ready to work on my own over the Christmas vacation.

6th to 26th December (Christmas vacation)

*<2 weeks (+ 1 weeks holiday) * 60% time (30hrs a week) = 60hrs>*

Complete success criteria, namely: Implement a Haskell library for automatic testing of combinatorial circuits written in Blarney. If any problems arise can use second half of Christmas vacation to finish the work, at the cost of fewer additional features. Otherwise if finished early start work on Dissertation.

Milestones: Code satisfies the first point of the success criteria. Introduction chapter of Dissertation mostly complete.

27th December to 16th January (Christmas vacation)

*<2 weeks (+ 1 weeks holiday) * 60% time (30hrs a week) = 60hrs>*

Start building additional features as able, with priority for synthesizing test rigs to an FPGA. Write Preparation chapter of Dissertation and start work on Implementation chapter.

Milestones: Synthesized test rig runs on FPGA. First 2 chapters of Dissertation mostly complete and structure of Implementation chapter laid out. Some of the proposed additional features complete.

17th January to 27th February

*<6 weeks * 6% time (3hrs a week) = 18hrs>*

Create progress report and presentation. Run the library on the example buggy circuits in preparation for Evaluation chapter. Start of lent term is very intense so use the long time at low intensity to make small improvements and find any bugs.

Milestones: Progress report and presentation done. Test data for Evaluation generated.

28th February to 12th March

*<2 weeks * 40% time (20hrs a week) = 40hrs>*

Complete all additional features and clean up code. Project should be stable at this point, with all changes being only minor after this point.

Milestones: Project is complete and stable to allow writing Dissertation.

13th March to 23rd April (Easter vacation)

*<2 weeks * 60% time (30hrs a week) = 60hrs>*

Finish up the Introduction and Preparation chapters and write Implementation chapter. Make only small crucial changes to code (e.g. bug fixes)

Milestones: First 3 chapters of Dissertation complete. Codebase finalized, ready for submission.

13th March to 23rd April (Easter vacation)

*<3 weeks (+ 1 week holiday) * 60% time (30hrs a week) = 90hrs>*

Write Evaluation and Conclusions chapters, running any additional tests to gather evaluation data if necessary. I can use this time to do additional work on the project if Timetable has been moved back, otherwise for revision for final exams.

Milestones: Dissertation finished, ready for submission if necessary.

24th to 30th April

*<1 week * 75% time = 37.5hrs>*

Review whole project, check the Dissertation, and spend a final few days on whatever is in greatest need of attention.

Milestones: Dissertation polished.

<Upper bound of 455.5hrs for project, with EV of about 350hrs>

1st to 7th May

*<1 week * ?? = up to 50hrs>*

Aim to submit Dissertation a week before the deadline.

Milestone: Submission of Dissertation.