Function Level Differencing Final Report

2016 Fall

11/20/2016

# Team Members Information

|  |  |  |
| --- | --- | --- |
| Team Members | Attended Classes Remotely? | Attended Meetings Remotely? |
| Harsh Fatepuria | No | No |
| Rakshith Subramanya | No | No |
| Kiran Kumar Budnar Venkatesh | No | No |

# Project Objective

The project objective was to add differencing feature on function level to the UCC C++ version.

# Project Summary

The main objective of this project was to introduce function level granularity to differencing of UCC. To introduce this functionality, we have made use of the differencing code that is already present in UCC. Adding a flag “**-funcDiff**” in the command line or selecting the new “**Function Level Differencing**” on GUI triggers the function level differencing. Currently function level differencing is dependent on differencing. Meaning, it can only be triggered when one has enabled differencing (-d). Currently, function level differencing is supported only by 4 programming languages namely – Python, Fortran, Verilog and IDL.

## Test Summary

<Testers should summarize the tests they ran. As in, what functions were tested, or the types of tests ran. >

# Projects’ Decisions

The approach we used to accomplish the objective is we use the same matched pairs list that the differencing tool creates, then for each entry in this list we call our own function parser that parses the files in each baseline and creates one temporary file for each functions present in that file. After this the differencing tool is used on this new temporary baselines. After each iteration the temporary baselines are deleted.

# Known Bugs/Issues

1. Supports only 4 programming languages. More function parsers for different languages are to be added.

# Development Notes

|  |  |  |  |
| --- | --- | --- | --- |
| **General Description of New Feature or Code Modification** | | | |
| 1. Introduce function level granularity to differencing of UCC. 2. Adding a flag “**-funcDiff**” in the command line triggers the function level differencing. 3. Currently function level differencing is dependent on differencing. Meaning, it can only be triggered when one has enabled differencing (-d). 4. Currently, function level differencing is supported only by 4 programming languages namely – Python, Fortran, Verilog and IDL. | | | |
| **New Files** | **Description** | **New Functions** | **Description** |
| FunctionParser.cpp | This file contains the function parsing class methods | callParser | Call function parser as per the ClassType of file |
| numberOfSpacesAtBeginning | Calculate number of white spaces in the start of a string |
| pythonParser | Parse the methods in a given python source code file and store them in different files |
| fortranParser | Parse the methods in a given Fortran source code file and store them in different files |
| verilogParser | Parse the methods in a given Verilog source code file and store them in different files |
| idlParser | Parse the methods in a given IDL source code file and store them in different files |
|  |  | In CUtill.cpp, RmPath | For a given path, this method deletes all directory and its content. |
| FunctionParser.h | Header file for FunctionParser.cpp |  |  |
| **Modified Files** | **Description** | **Modified Functions** | **Description** |
| GMainWindow.cpp |  | on\_btnStart\_clicked | Added check if function differencing enabled in GUI |
| GMainWindow.cpp |  | on\_btnStart\_clicked | Added check if function differencing enabled in GUI |
| GMainWindow.cpp |  | on\_chkDifferencing\_clicked | Enabling function level differencing checkbox if differencing checkbox selected |
| GMainWindow.ui | Added code for enabling a checkbox for function level differencing |  |  |
| gucc.pro | Added src paths for new files (FunctionParser.cpp and FunctionParser.h) |  |  |
| **Deleted Files** | **Description** | **Deleted Functions** | **Description** |
|  |  |  |  |

# Set Up and Improve Instructions

**Adding more function parsers for other programming languages**

1. Add a parser method to the file FunctionParser.cpp for each programming language you want UCC to support.
2. Add a case in callParser() method in FunctionParser.cpp for each programming language and call the appropriate parser form this case.

# Final Project Plan

|  |  |
| --- | --- |
| **Week** | **Activities** |
| 9/5/16 - 9/18/16 (2 Weeks) | Set up, build UCC source code |
| 9/19/16 - 9/25/16 | Initial project plan |
| 9/26/16 - 10/9/16 (2 Weeks) | Understanding UCC C++ codebase |
| 10/10/16 - 10/16/16 | Started working on Python Parser |
| 10/17/16 - 10/23/16 | Added GUI support and command line flag for function level differencing |
| 10/24/16 - 10/30/16 | Initial version of function level differencing for Python language. Simultaneously testing of codes are also carried out |
| 10/31/16 - 11/6/16 | Created separate output files for function level differencing. Fixed minor bugs that were found previously.  Simultaneously testing of codes are also carried out |
| 11/7/16 - 11/13/16 | Added few more language parsers(Fortran and Verilog) and enabled threading support for function level differencing.  Simultaneously testing of codes are also carried out |
| 11/14/16 - 11/20/16 | Added support for IDL language and few more bug fixes |
| 11/21/16 - 11/27/16 | Documenting final report |

# Team’s Strengths and Weaknesses

* Distribution of the tasks.
* Team meetings were regular and productive.
* Contributing to team’s objectives and working effectively

# Final Deliverables Checklist

* Completed Final Report
* Copy of original/baseline source code (before modifications) – be sure it is the code you used to make modifications to
* Final source code with modifications
* Final version of Test Cases
* Final version of Defect Log
* Final version of all other documents
* Other. Explain: