AUTOMATED REGRESSION ENVIRONMENT FOR IMPROVING DESIGN VERIFICATION PRODUCTIVITY

By
Keshav Rath
(Roll No. 2013104)

SUPERVISOR(S):

EXTERNAL:

Mrs. RAMA KOWSALYA NAMBURI
Senior Project Manager
Canon ISDC, Bangalore

INTERNAL:

Prof. P. N. KONDEKAR
Professor
PDPM IIIT DM Jabalpur

Electronics and Communication Engineering (B. Tech 2013)

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, DESIGN AND MANUFACTURING JABALPUR

Report

(27th July 2016 – 9th August 2016)

1. INTRODUCTION

The main purpose of this project is to enhance the design and verification productivity using PERL for automation. It will enhance the speed for the analysis of the reports which further help in the design and verification productivity.

PERL is the most well-known language for the practical extraction of the files and reporting language. It has borrowed features from other programming languages C, AWK, sed (stream editor), sh (shell scripting), and BASIC (Beginners' All-purpose Symbolic Instruction Code) as well. It provides a powerful text processing facilities, facilitating easy manipulation of text files.

For this project prior knowledge of PERL as well as VHDL has to be acquired. Some basic knowledge about verification methodology (such as OVM, UVM) is also required.

PERL

PERL stands for Practical Extraction and Report Language. Perl is a stable, cross platform programming language. It is also called a scripting language that has been developed by Larry Wall in 1987. This language was initially developed as a general purpose Unix Scripting Language but later it became a different independent Language. This scripting language has a lot of features such as supporting databases from third party such as Oracle, Sybase, MySQL. It can support Unicode, it's Y2K compliant, can be both procedural as well object-oriented. This language is used for application such as graphics programming, system administration, network programming, finance, bioinformatics, and other applications.

For this language all the basics were learnt, which include the basic syntax and semantics to using external CPAN modules for better productivity.

CPAN

Perl has mechanisms to use external libraries of code, making one file contain common routines used by several programs. Perl calls these modules. The

Comprehensive Perl Archive Network (CPAN) is a repository of over 150,000 software modules and accompanying documentation for 33,000 distributions, written in the Perl programming language by over 12,000 contributors. The CPAN's main purpose is to help programmers locate modules and programs not included in the Perl standard distribution. Its structure is decentralized. Files on the CPAN are referred to as distributions. A distribution may consist of one or more modules, documentation files, or programs packaged in a common archiving format, such as a gzip tar archive or a Zip file.

For the understanding of this and to practice these modules available, following modules I have downloaded and practiced upon them.

CPAN Class: Spreadsheet

- It helps to access an excel file.
- It helps to read or write an excel file through the PERL code.
- Spreadsheet::WriteExcel – To write a string, number to an excel file (Excel 95-2003).
- Spreadsheet::XLSX – To read the content of the excel file (Excel 2007).
- Spreadsheet::ParseExcel – Also to read the content of the excel file (Excel 95-2003).
- Spreadsheet::WriteExcel::Chart – To draw a pie chart using the data in the excel sheet in the sheet itself.

Spreadsheet is a broad class in CPAN and it has the many modules in it. Some of them I have described above. All the above modules are not independent; they need some of the other modules for their implementation. Some of those dependant modules are:

- Crypt::RC4 – To use the RC4 encryption algorithms
- Digest::MD5 – Used for RSA data encryption.
- OLE::Storage_Lite – For Document Interfacing
- ExtUtils::MakeMaker – For creating Makefile module.
- Data::Dumper – For advanced data structures, suitable both for printing and evaluation.
- IO::Scalar – For I/O of strings, basics of OO(object oriented) interface

3. TASK

Previously a PERL script was created to run single test or regression tests according to the users' desires. These tests are Verilog and SystemVerilog codes which have to be either tested one at a time or in a bulk (parallel or sequentially). For both kinds of tests there are options which can be provided according to the users' requirement for the given situation. These options can be mandatory or optional. Some of the mandatory options can be the name of the test(s), the list of testbenches, and the directories to be included before running any testcase and the tool and the simulation mode to be used for running these testcases. The options which are options can be listed as enabling GUI with which we can see the waveform generation and schematic diagrams of blocks. Coverage reports can also be generated. A summary of some important details of each test has to be written in an excel file.

4. RESULTS AND DISCUSSION

Initially PERL was studied and some further research into CPAN modules was done in order to get the initial task completed. The initial task was completed as per requirement and tested with the limited options available. The tests either passed or failed according to the nature of the test. Various options can be given in order to make the test run under various circumstances.

5. CONCLUSION

A hands on experience on PERL language and its external libraries storage such as CPAN was developed. A script was written in order to run test/tests in a particular manner so as to test every possible way.