A SystemC-based UVM verification infrastructure

Mike Bartley and Harshavardhan Narla

of Test and Verification Solutions Ltd

accellera SYSTEMS INITIATIVE™

2015 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

INTRODUCTION (or REQUIREMENTS)

SystemC-based design is now gaining traction with improvements in the tools for implementation of such designs. TVS recently undertook a project to verify a number SystemC design IP blocks. It was decided to implement a SystemC test bench that would be UVM-compliant with a TLM2.0 interface. This entailed the following developments which are detailed within the paper:

A C++ class library equivalent to the UVM class library.

A Functional Coverage Library (FCL).

Constraint based random verification was enabled through use of external randomization library called CRAVE.

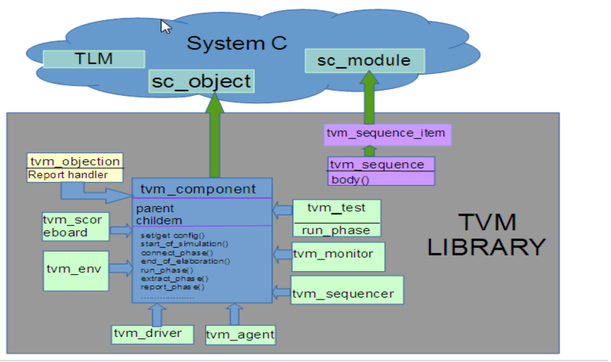

Fig 1 : Structure of TVM Library

OBJECTIVES

- To create a SystemC test bench which is UVM-compliant with a TLM2.0 interface.
- To improve the quality of the outgoing IP using constrained random verification.
- To create a Functional Coverage Library(FCL) which is compliant to SystemVerilog coverage.
- To have a well-defined verification strategy based on the IP requirements, feature extraction and a UVM test bench, supported by asureSIGNTM

RESULTS

- An infrastructure developed for the C++ class library named TVM (TVS Verification Methodology) equivalent to the UVM (Universal Verification Methodology) class library.
- The TVM library includes factory constructs, agents, monitors, scoreboard, drivers and sequencers enabling easy conversion from TVM->UVM or UVM->TVM.
- The TVM library mimicked UVM phases such as compile, build, run, check etc.
- Tests can be compiled using the free GCC based compilers. Figure 1 shows the structure of the TVM library.
- Functional coverage is enabled through a TVM Functional Coverage Library (FCL). It has support for multiple cover groups in a single instance; conditional coverage; exclusion or Inclusion of cross bins; transition Bins (one series of incrementing transition); auto bins (up to 16 bits); cross coverage (for up to 4 cover points).
- The functional coverage report is generated in csv or xml format using the "Tiny xml" library add-on. This format is compatible with TVS asureSign™ so that the coverage can be viewed against requirements.
- The library includes base classes for constraint driven randomization using CRAVE (Constrained RAndom Verification Environment).
- Code Coverage is also possible through gcov and lcov tools.

Initial Deployment Results

- The above methodology has been applied to block verification with the following initial results:
- Re-usable TVM agents were developed for the proprietary protocols based internal busses. These were re-used on subsequent IP blocks.
- High rates of both functional and coverage were achieved on all blocks.
  - 100% statement coverage on all blocks.
  - The structured feature extraction process was converted to a functional coverage model. TVS were then able to generate tests to hit 100% of the plan.
  - Throughout the coverage closure process progress was tracked via asureSIGN and reports generated.
- 35 bugs were discovered in RTL and another 7 in models.

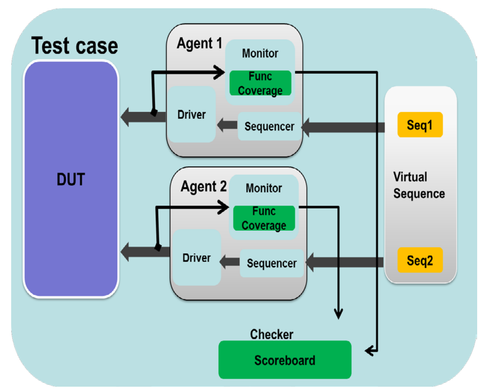

Fig 2 : Sample TVM Setup

CONCLUSIONS

The first Infrastructure developed was a C++ class library (named TVM) equivalent to the UVM class library. Constraint based random verification was then enabled through the use of an external randomization library called CRAVE. A Functional Coverage Library was also developed to enable Coverage Driven Verification (CDV). This library can be used either with TVM or individually on C++ based environments.

The application of the TVS verification methodology was successful. Constraint-based random verification was enacted and high coverage rates were achieved for both functional and code coverage (the latter enabled by freely available gcov and lcov tools). This enabled us to achieve high rates of bug detection.

Using the TVS-TVM infrastructure enabled our clients to benefit from a license free verification environment & lower costs. It was also able to demonstrate the improved IP quality through metrics such as functional, code coverage, and bug discovery rates.

The above libraries are all freely available for engineers wanting to write UVM-based SystemC test benches see contact information below..

REFERENCES

[1] Hoang M. Le and Rolf Drechsler, "CRAVE 2.0: The Next Generation Constrained Random Stimuli Generator for SystemC," DVCON Europe, 2014

[2] Universal Verification Methodology (UVM) 1.1 User's Guide, Accellera, May, 2011

[3] Universal Verification Methodology (UVM) 1.2 Class Reference, Accellera, Jun, 2014