HDL IP Cores System as an Online Testbench Provider

Vladimir Zdraveski, Andrej Dimitrovski and Dimitar Trajanov

Abstract - A huge part of the HDL development process is spent on testing and simulation. Supporting the idea of a testbench design automation, we present a module of the HDL IP Cores system, integrated with a client-side eclipse plug-in, as an automatic testbench search engine embedded inside the designer's native programming environment. The concept is extended with the use of a simulator for compatibility verification and existing results ranking improvement.

Keywords - automation, HDL, search, testbench, verification.

I. Introduction

The frequent use of the programmable chips in production systems requires reliability and stability, so the whole process of HDL design must be kept in an enterprise level, guaranteeing customer's satisfaction and security. Companies, aware of this fact, are very interested in the improvement of testing and verification methods of their products [1][2]. Some of them have a separate testing departments, fully dedicated to the IP cores verification [3][4]. and also have advanced testing process workflows, that greatly decrease the error probability [5].

The era of open-source development allows a significant rise in the development of applications. Outside the commercial world, there is a bunch of open source HDL including test benches. There are several web portals, groups and online communities [6]-[12]. These repositories have too many projects in which it is very difficult to find the required component and also there is no central hub to connect all of them together, so the designer have to visit them one by one. Yet many people worldwide work on a similar projects and design very related IP cores and testbenches, but unfortunately are not aware of each other, do the same thing and waste time.

There are also numerous of open source testbench files on the Internet and enterprise development allows and stimulates the code reuse, that is of a particular importance time saving. But, it is very difficult to get the right one, due to the large growth of the set of existing testbench components available online, which should be searched including the procedure of checking the parts, that the testbench contains, before using it in your project.

More over large companies have their own huge repositories of testbenches, used in the past (fully verified), and face the same problem of lack of automation tools inside their borders, that will support the code reuse and speed up the development process. So, the testing and debugging process is yet quite time consuming [13] and commercially inefficient.

The HDL design, without doubts, depends on the testing and simulation, that sometimes take a lot of time [14]. Very often, the time required to write a testbench is comparable to the time required to prepare the IP core itself, that is a quite undesirable information for the companies' management teams.

In that direction we present a concept for testbench retrieval and seamless integration into the designers' native workspace. Using our existing HDL IP Cores system [15] and integrating existing client-side tools and simulators, we managed to go a step forward and provide the users an automated functionality of a testbench search and download directly inside their HDL programming environment.

II. Related Work

A. HDL IP Cores System - Overview

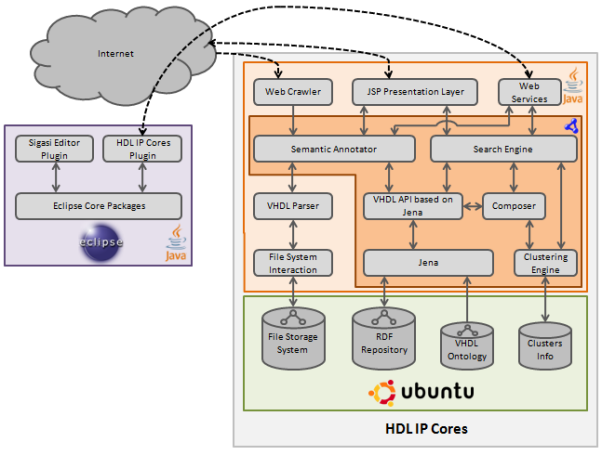

Fig. 1 - HDL IP Cores architecture

The system, Fig. 1, consists of a server side application and a client side eclipse plug-in. The server side application contains a web crawler, that downloads IP cores from the Internet and then each IP core passes through a process of annotation. The metadata together with the IP core source information are stored in RDF Repository. HDL IP Cores system does a deeper semantic annotation and provides

Vladimir Zdraveski, Andrej Dimitrovski and Dimitar Trajanov are with the Department of Information Systems and Network Technologies, Faculty of Computer Science and Engineering, "Ss. Cyril and Methodius" University, Ruger Boskovik 16, 1000 Skopje, Macedonia, E-mail: {vladimir.zdraveski, dimitar.trajanov, andrej.dimitrovski}@finki.ukim.mk

components ranking by similarity and compatibility with a given component.

The second part of our system is the client side Eclipse plug-in, that can access the core system through a web service interface. The main idea is to provide the users with the functionality of searching and downloading IP Cores available and annotated in the server-side repository. We use the "Sigasi" editor plug-in [16] for the client side HDL functionalities, but any other plug-in, that provides HDL editor, may be used. Annotation is based on the knowledge placed in our custom ontology [17].

B. Functionalities of the existing simulators

Verissimo System Verilog Testbench Linter [18] is a coding guideline and verification methodology compliance checker, that enables engineers to perform an additional audit of their testbenches. With this tool, designers can check whether their code is free of language pitfalls and semantic or style issues, and compliant with the appropriate methodologies. Verissimo can be customized to check specific group or corporate coding guidelines to ensure consistency and best practices in code development. For example, the possibility of implementing the same functionality in multiple ways may impact the simulation performance or lead to unexpected behavior.

VCS' Native Testbench (NTB) [19] technology provides built-in natively-compiled support for full-featured System Verilog and Open Vera testbenches, including object-oriented, constrained-random stimulus and functional coverage capabilities. VCS further expands its capabilities with Echo constraint expression convergence technology. Echo automatically generates stimuli to efficiently cover the testbench constraint space, significantly reducing the manual effort needed to verify a large number of functional scenarios. Echo is a perfect fit for all teams using System Verilog testbenches with random constraints.

Cadence's Enterprise Simulator [20] supports all IEEE-standard languages, the Open Verification Methodology (OVM), Accellera's Universal Verification Methodology (UVM), and the ***e*** Reuse Methodology (***e***RM), making it quick and easy to integrate with your established verification flows. Functionality of Enterprise Simulator provides a high-throughput channel between the testbench and the device under test (DUT). This enables automated metric-driven verification of embedded software exactly as if it was another part of the DUT. Today, Enterprise Simulator fuels testbench automation, reuse, and analysis to verify designs from the system level, through RTL, to the gate level. It supports the metric-driven approach implemented by Incisive Enterprise Manager. Its native-compiled architecture speeds the simultaneous simulation of transaction-level, behavioural, low-power, RTL, and gate-level models—critical to the verification of a modern multi-language, multi-abstraction and mixed-signal SoC.

Xilinx ISE Simulator (ISim) [21] provides a complete, full-featured HDL simulator integrated within ISE. HDL simulation now can be an even more fundamental step within your design flow with the tight integration with the ISim within your design environment. Xilinx tools automatically generate lines of VHDL code in the testbench file to get you started with circuit signals definition and define the inputs and outputs. The simulator has few other tools in order to run, pause and stop the simulation.

Despite the mentioned simulator implementations and their basic logic [22], there are also useful ideas for conceptual improvement [23][24]. All previously mentioned simulators have different features that implement the simulation (verification) of the testbench components. But they come to the scene after the designer would manually instantiate a testbench component in the simulator and then use the available tools. None of the previously mentioned simulators offers an automatic online search for testbench components and easy code reuse.

IV. Test bench provider module

Our approach is a context aware testbench search tool, that use ontology-based knowledgebase. Our system uses semantic annotated data to find the right testbench component and integrate into the development environment of the user, in our case Eclipse. HDL IP Cores system will be used to facilitate the generation of the result list of testbenches. But also, a server side verification of the compatibility between the testbench and component is required. The system should perform a verification i.e. automatic simulation of the testbench, producing ranked result list of suitable testbenches. The simulation also could be made on the client side, using a simulator, that is embedded in our plug-in or engaging with another third-party plug-in simulator, already installed inside the client environment.

A. Test-bench retrieval process

In our system, the search for a testbench is made by the use of a search engine, based on OWL domain ontology and RDF knowledgebase [15]. Since the HDL files have a predefined structure by themselves, the annotation is done automatically, using custom ontology as a domain data schema. Although the process requires no further input by the end users, it is a step forward in the HDL code search engines improvement.

Our tool enables the designer to search directly from his workspace, i.e. to run the "Find Testbench"-tool, Fig. 2, with a right-click on the component file. User's component is then sent to the HDL IP Cores server, annotated and used as an input to the compatible testbenches list generator.

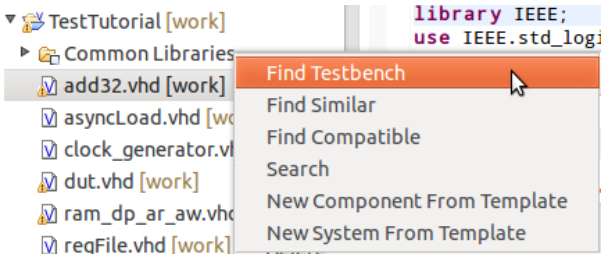

Fig. 2 - Client side functionality

The search is made by matching the semantic concepts specified in the user's request and the semantic annotations of the components available in the RDF repository of the system, Fig. 3. Also a very important part of the result list is the port map between the user's required component and resulting testbench's component (the component which is inside the testbench in the repository, Fig. 3), that enables the client plug-in to instantiate the user's component inside the testbench and run the simulation automatically .

A new window will display the testbench components that correspond to the selected component. This is especially important for the designer, because he will obtain the testbench compatible to his component within his project, without Internet browsing and downloading hundreds of files and archives.

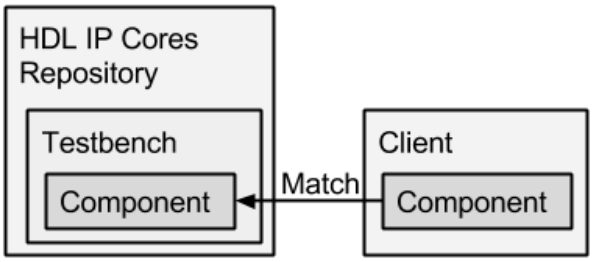

Fig. 3 - The matching process

The system provides a preview of the test bench component and after the user's choice, the testbench will be instantly downloaded to the project and lead the user just a step to the simulation execution.

We have to note that automatic file download is done in run time, directly from the original file URL. There is no HDL code in the HDL IP Cores repository, supporting the intellectual property and licensing paradigm and possible licensing condition changes in future. Direct download is available for the license free IP cores only and when the IP core is published under different license, the system will redirect the user to the IP core provider's web site.

B. Testbench compatibility verification

The testbench generates input values for the design under testing and checks the answers, so design must be simulated carefully to find errors. The patterns, VHDL simulation stimuli, are described in a specific formalism, that can be captured using a dedicated language generation pattern. Once a VHDL behavioral description is written and a set of test vectors have been determined, a functional simulation is started.

To simulate a design the testbench must be compatible with the architecture or otherwise warnings or errors will appear in the simulator's output. We propose to instantiate the testbench, run the simulator automatically and use the simulator's output in the testbench results ranking process. Moreover, the result list may contain warning and error flags on each result item, notifying the client that if he chooses that testbench he will have to review and correct warnings and/or errors in order to go on with the simulation process. In our system there are two possible verification solutions, a client side and a server side.

The first scenario is the simulation on the client-side application (Eclipse plug-in) that includes simulation of the actual testbench component. There are several different simulation tools. One is DVT Eclipse, which is a plug-in for Eclipse [25], providing a satisfying environment to simulate a testbench. DVT integrates seamlessly with all major hardware simulators to enable simplified simulation analysis. The designer may execute the simulation of the testbench component right in his project. Using the capabilities of DVT and its integration into our system will enable automatic compatibility verification on the list of testbenches received from the HDL IP Cores server.

This will allow verification of the semantically annotated testbenches for the selected component, and will show to designer the test benches which are the most compatible with the component architecture. The simulator would be integrated on the client side as a part of the Eclipse Core Packages. It will accelerate the process of finding the most compatible test benches and will save time compared with the manual simulation, Fig. 4, because the testbench will already be inside the designer's project and will encompass syntax and semantic checks with errors highlighted as the designer types, will do an initial port mapping.

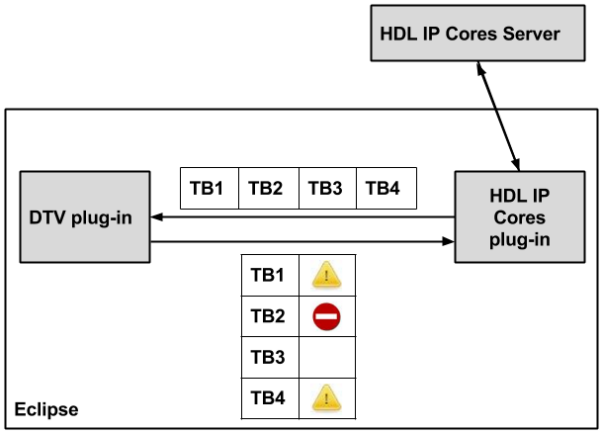

Fig. 4 - Client side simulator

Since the compatibility check is made on the client side, we may optionally provide integration with few other simulators available for Eclipse and the user will be able to choose among them and use his preferred one together with our system.

The second possible solution architecture, Fig. 5, is to set the simulator [26] on the server side. Its role again will be to determine the compatibility between components and test benches that obtain ranked results. The integration of the simulator on the server side would generate overhead data and additional server's CPU usage due to the verification (simulation) simulation. The delay will depend on the size and type of the component and the testbench, but the client will get the final list and HDL designer will not have to install additional client-side plug-in, but the HDL IP Cores plug-in only.

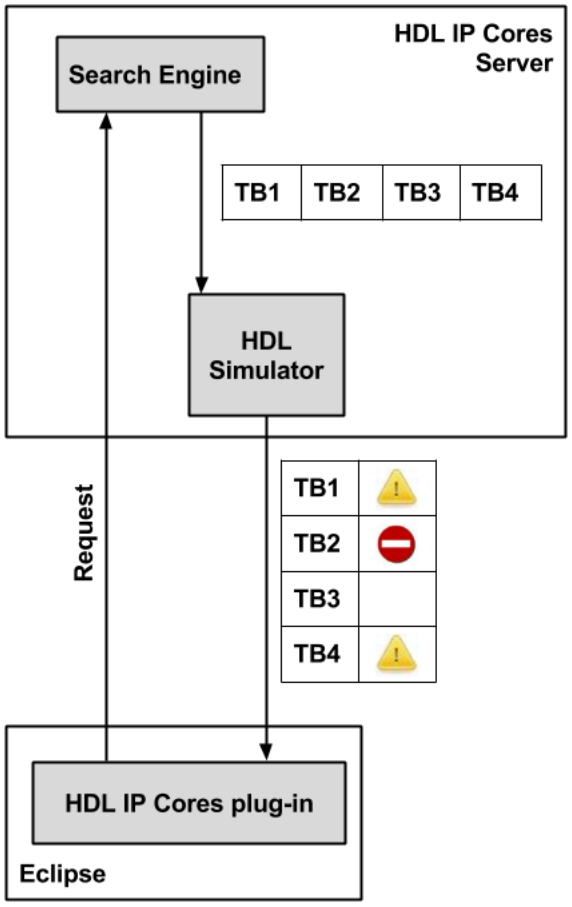

Fig. 5 - Server side simulator

IV. Conclusion and Future Work

In order to speed up the testbench generation process we described our existing system's testbench searching feature and two possible improvement concepts using a third-party HDL simulator. The system architecture and the client side functionality were described in details, providing a global picture of the whole concept.

The next steps would be to implement both scenarios and do a performance tests and evaluation survey in order to get user feedback comments.

The main benefit of our proposed concept is that testbenches will be annotated according to a central ontology (inside a company or worldwide) and users will be able to find and download a specific testbench faster, very easy and directly via their native designing workspace, without a need to open a browser and visit tens of web pages.

In a commercial environment it is possible to deploy a local instance of the HDL IP Cores system and integrate it with company's native code storage engine, inheriting users credentials and access permissions and rules. This way we will provide the HDL-designers with the described functionalities, keeping their code repositories inside the company, that is of essential importance in an enterprise environment.

Acknowledgement

The work in this paper was partially financed by the Faculty of Computer Science and Engineering, at the "Ss. Cyril and Methodius" University in Skopje, as a part of the research project "Semantic Sky 2.0: Enterprise Knowledge Management".

References

[1] Arabi, K., "Special session 6C: New topic mixed-signal test impact to SoC commercialization," *VLSI Test Symposium (VTS), 2010 28th* , vol., no., pp.212,212, 19-22 April 2010.

[2] Chung-Yang Huang; Yu-Fan Yin; Chih-Jen Hsu; Chung-Yang Huang; Ting-Mao Chang, "SoC HW/SW verification and validation," *Design Automation Conference (ASP-DAC), 2011 16th Asia and South Pacific* , vol., no., pp.297,300, 25-28 Jan. 2011.

[3] He Zhang; Chunyu Wu; Wenjing Zhang; Jiwei Wang, "Design on SOC module-level functional verification platform," *Mechanic Automation and Control Engineering (MACE), 2011 Second International Conference on*, vol., no., pp.4012,4015, 15-17 July 2011.

[4] Qingdong Meng; Zhaolin Li; Fang Wang, "Functional verification of external memory interface IP core based on restricted random testbench," *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on* , vol.7, no., pp.V7-253,V7-257, 16-18 April 2010.

[5] Lulu Feng; Zibin Dai; Wei Li; Jianlei Cheng, "Design and application of reusable SoC verification platform," *ASIC (ASICON), 2011 IEEE 9th*

International Conference on , vol., no., pp.957,960, 25-28 Oct. 2011.
[6] Open Cores – web portal, http://opencores.org/
[7] Java optimized processor - group , http://tech.groups.yahoo.com/group/java-processor/
[8] IP supermarket – web portal, http://www.ipsupermarket.com/index.php
[9] Infineon – web portal, http://www.ipsupermarket.com/index.php
[10] Lattice – web portal, http://www.latticesemi.com/
[11] Chip Estimate – web portal, http://www.chipestimate.com/
[12] Design & Reuse – web portal, http://www.design-reuse.com/
[13] Cheng, X.; Ruan, A.W.; Liao, Y.B.; Li, P.; Huang, H.C., "A run-time RTL debugging methodology for FPGA-based co-simulation," *Communications, Circuits and Systems (ICCCAS), 2010 International Conference on*, vol., no., pp.891,895, 28-30 July 2010.
[14] Corno, F.; Sanchez, E.; Reorda, M.S.; Squillero, G., "Automatic test program generation: a case study," *Design & Test of Computers, IEEE* , vol.21, no.2, pp.102,109, Mar-Apr 2004.
[15] V. Zdraveski, M. Jovanovik, R. Stojanov and D. Trajanov. "HDL IP Cores Search Engine based on Semantic Web Technologies", *ICT Innovations 2010, Communications in Computer and Information Science*, Volume 83, 2011, pp 306-315, Ohrid, Macedonia, September 2010.
[16] Sigasi - intelligent HDL editor plug-in for eclipse http://www.sigasi.com/
[17] V. Zdraveski, D. Trajanov. "VHDL IP Cores Ontology", *Conference for Informatics and Information Technology (CIIT)*, Bitola, April 2013.
[18] Verissimo System Verilog Testbench Linter http://www.dvteclipse.com/Verissimo_SystemVerilog_Testbench_Linter.html
[19] VCS' Native Testbench (NTB) http://www.synopsys.com/Tools/Verification/FunctionalVerification/Documents/vcs-ds.pdf
[20] Cadence's Enterprise Simulator http://www.cadence.com/products/fv/enterprise_simulator/pages/default.aspx
[21] Xilinx ISE Simulator (ISim) http://mazsola.iit.uni-miskolc.hu/~kulcsfm/DigRendII_elemei/VHDL_anyagok/ISE_Simulator_ISim_VHDL_TestBenchTutorial.pdf
[22] Brown, A.D.; Nichols, K. G.; Zwolinski, M., "Issues in the design of a logic simulator: an improved caching technique for event-queue management," *Circuits, Devices and Systems, IEE Proceedings* - , vol.142, no.5, pp.293,298, Oct 1995.
[23] Maksimovic, D.M.; Litovski, V.B., "Timing simulation with VHDL simulators," *Microelectronics, 2002. MIEL 2002. 23rd International Conference on* , vol.2, no., pp.655,658, 2002.
[24] Maksimovic, D.M.; Litovski, V.B., "Tuning logic simulators for timing analysis," *Electronics Letters* , vol.35, no.10, pp.800,802, 13 May 1999.
[25] DVT Eclipse - client side integration http://www.dvteclipse.com/?gclid=CM3Ex_G4_7oCFc1V3godKA4Abw
[26] Cadence simulator - server side integration http://www.cadence.com/ip/vip/pages/default.aspx