International Journal for Research in Engineering Application & Management (IJREAM)
ISSN : 2454-9150 Vol-05, Issue-03, June 2019

Hardware and Software Co-simulation Environment with Integrated Graphical User Interface

[1]Ojaswi Zope, [2]Charudatta Kulkarni

[1]Student, [2]Associate Professor, Dr. Vishwanath Karad MIT World Peace University, Pune, India,

[1]ojaswizope22@gmail.com, [2]cvk1971@gmail.com

Abstract : - **Co-simulation of hardware and software together provides a methodology for ASIC verification. ASIC based devices will have a part of software running on hardware to manipulate it. So, co-simulation plays important role in good design tradeoffs. Therefore, this methodology can be used to verify hardware and software functionality precisely after prototype of hardware is on tap. In this paper, a flow is described to add a graphical user interface in co-simulation environment. A graphical user interface is a display in which user can see the different parameters of device under test, depending upon how it is integrated and can either manipulate it. Graphical user interface provides easy view to the registers of hardware while software is running on hardware.**

Keywords — Co-simulation, DUT, GUI, Interface, models, Verification, VPI.

I. INTRODUCTION

The Complexity of designing hardware and software is increasing due to recent demands in ASIC [1]. Debugging the product after implementation becomes difficult. Therefore, these types of complexity demands some new ways of simulation and early stages of debugging before actual product implementation [3]. Co-simulation plays crucial role in development and debugging ASIC. Co-simulation environment includes a hardware running on an emulator and a simulator for software simulation.

Nowadays, it is generally accepted that hardware verification languages should be assembled with new software techniques [5]. The inference taken out is that, the verification of system is of utmost complexity and time-exhausting exercise for the overall design process. Approach for verification in which another language is used as a verification method requires co-simulation. This graphical user interface should contain parameters necessary to monitor at the runtime or to manipulate it. It provides an efficient and an accelerated way for verification. In this paper, a flow is described as to how to add graphical user interface in the running environment and about how it gets compiled in environment.

II. PREVIOUS WORK

The proposed techniques, for hardware and software co-simulation has been tradeoff between number of factors and emerging requirement for the development of ASIC. Many environments are designed for improving the speed of co-simulation. In Paper by Rowson [3], he referred to the techniques available for co-simulation with an importance towards the strengths and weaknesses of each hardware and software. He mentioned about different techniques available with and without using models.

The techniques with software models are nano-second accurate timing model, cycle accurate and instruction set accurate. The fastest software model emulates the instruction set accurately, which verifies all the register values correctly and at correct time stamp. Though the emulation speed of instruction level model is fast, debugging using these models is difficult.

The techniques requiring no software models as mentioned by Rowson are synchronized handshake technique, virtual hardware technique, bus functional technique, hardware modeler technique and emulation. The bus functional models technique creates a test-bench in which it verifies weather the interfaces are correct. Hardware modelers use an absolute model and most specific models from the functionality perspective. Emulation provides the nearest possible hardware prototype to be available and gives the clarity of internals similar to the actual hardware.

In paper by Vojin [2], they presented methodology for compiled co-simulation. This technique is the fastest and precise instruction set simulation. Simulator attends to give support of standard debugger (C level debugger which is gdb) in environment. Using standard debugger in for debugging of hardware and software gives the biggest advantage. The simulator translates every instruction exactly to one or more host instruction.

III. HARDWARE AND SOFTWARE COMPONENTS

The components of hardware and software are developed simultaneously. System includes hardware emulator and

 DOI : 10.35291/2454-9150.2019.0247

software simulator, only synthesizable part runs on emulator and non- synthesizable part runs on software simulator. Device under test behaves as an exact hardware. It would contain exact register values setup from software running on it. While elaboration of the design, hardware and software components are separated proper connections between device under test and test-bench are to be made.

GUI (Graphical User Interface) gets compiled and runs in software simulator. GUI is always specific and to be designed according to the verification model to be used as required for device under test. It contains parameters which are to be observed from the device under test. GUI is designed by using GTK toolkit which is open source platform.

During co-simulation, at run-time a GUI is invoked. GUI can be responsible for monitoring output values from hardware or setting values in hardware. Interface between GUI and hardware is through socket channel. Socket is a way of connecting two nodes on a channel to communicate with each other. One socket (node) listens on a particular port at an IP, while other socket reaches out to the other to form a connection. Synthesizable part which will go in hardware is pure VHDL/Verilog code [6]. Non-synthesizable part which includes test-bench, GUI executable, Verilog Procedural Interface (VPI) functions and socket nodes will go in software simulator. Verilog procedural interface is the method in which one can add functionality to the HDL using C-language. These functions are linked to the simulator so that they can be identified during the compilation and elaboration. VPI function definition and GUI executable forms a dynamic library (.so) file.

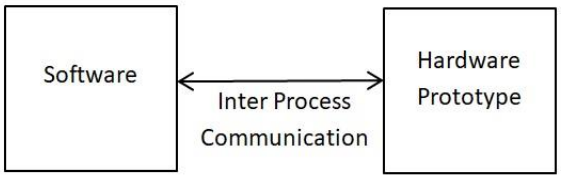

Fig. 1 Communication between hardware and software

IV. METHODOLOGY

Approach is based on co-simulating hardware and software with GUI. Where, GUI plays a part for monitoring and setting values of hardware. VPI functions which are linked to the simulator and are invoked as C functions to which inter-process communication utilities are linked. Fig. 2 shows the components of co-simulation environment. Hardware may not be on host system, it runs on emulator. Simulation runs on host system. As soon as software is dumped on hardware, it tends to change the hardware register parameters and registers manipulates the required functionality. As per changes in registers of hardware, required VPI functions are invoked and those values are manipulated on GUI.

There are two socket nodes at two different processes communicating with each other. One socket node is at GUI side forming one process and other socket node is at simulation side (Simulator Process) forming second process to form a socket channel that is inter-process communication. Information between hardware and GUI is exchanged via socket channel. The read and recv functions are used to send and receive data through socket channel.

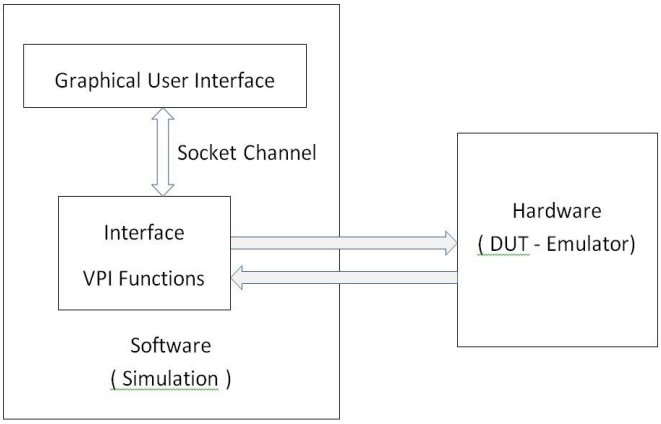

Fig. 2. Components of Co-simulation Environment

A. Input to the hardware from GUI.

Any input logic value entered in input box is a key press event for GUI. Data can be entered through the GUI at any human speed, data forms a queue. Then, data is send through the socket channel to the VPI function. According to the models, VPI function writes values in hardware. VPI function using its utility access routines can write logic values on ports of hardware.

B. Output to the hardware from GUI.

Any output logic values coming up on hardware port is monitored. As soon as the change is occurred, VPI function retrieves that logic value using access routine and sends through the socket channel on GUI. While sending data on GUI it is necessary to encode data in a format so that it gets identified by the GUI on where to put. Monitoring hardware values at simulation time would decrease verification time.

V. CONCLUSION

This Methodology with included GUI will help to decrease the verification time and various design trade-offs. GUI allows one to monitor port values or register values at run time on GUI and also to set value on ports at run time. Communication between two processes through socket channel happens at no-time. Co-simulation will turn out to be a fast process altogether when device under test parameter values are monitored using GUI.

 DOI : 10.35291/2454-9150.2019.0247

REFERENCES

[1] K. K. Y. S. T. A. W. S. K. C. S. H. Yongjoo Kim, "An Integrated Hardware-Software Cosimulation Environment for Heterogeneous Systems Prototyping," in *Proceedings of ASP-DAC'95/CHDL'95/VLSI'95 with EDA Technofair*, Chiba, Japan, 1995.

[2] H. M. V. Zivojnovic, "Compiled HW/SW co-simulation," in *33rd Design Automation Conference Proceedings*, Las Vegas, NV, USA, 2002.

[3] J. Rowson, "Hardware/Software Co-Simulation," San Diego, CA, USA, 2006.

[4] M. S. G DeMicheli, Hardware/software Co-design, 2013.

[5] R. K. S. S. G. T. David Becker, " An Engineering Environment for Hardware/Software Co-Simulation," in *design automation conference*, 1992.

[6] E. F. L. L. Bassam Tabbara, "Fast Hardware-Software Co-simulation Using VHDL Models," in *Design, Automation and Test in Europe Conference and Exhibition Research gate*, 1999.

[7] M. B. R. C. C. C. A. J. M. L. D. T. O. Y. A. Ghosh, "A Hardware-Software Co-simulator for Embedded System Design and Debugging," in *Design Automation Conference*, 1995.

 DOI : 10.35291/2454-9150.2019.0247