Potential Use of RTL Co-Simulation

K.Priyadarsini, S.Karthik

***Abstract*— This paper's objective is to highlight the potential use of C/RTL Co-Simulation and application of it in satisfying the time constraints involved in verification of industrial, complex, monolithic. Co-Simulation performed in Xilinx software has provided a platform for further analysis of designs. Steps before performing co-simulation has provided statistical data about how much resources the system design requires which can be further analyzed, part-by-part, using profiling method. The profiling allows demarcation of a distinctive line between resource-intensive processes and time-intensive processes. A further study into this matter would purge the need of resource-intensive and time-consuming devices. This could be a small step towards attaining a level of production where the outcome is a better device and error-less production of these devices.**

***Keywords*— monolithic; constraints; complex; stylin.**

I. INTRODUCTION

Co-simulation has become one of the most important issues in hardware/software co-design of very complex systems, especially for Multiprocessor System on Chip (MpSoC). One of the most important ways of simulating complex hardware/software systems is the use of high-level languages with the Co-simulation abilities of EDA tools.

As design gets more complicated, EDA-industry specialists have cautioned, the way toward checking these plans will turn out to be increasingly costly and tedious. Sometimes, the verification procedure might be considerably more intricate than the plan that is being checked. In the past few years, high-performance embedded systems for control, signal-processing, and communication applications have delivered on the first part of the warning (i.e., complexity). Such system complexity has been driven by the trend to miniaturize designs and integrate multiple functional units into a single device.

Designing a system involves simulating the designed system. Most of the systems that are commercialized are complex and require extensive time requirements to simulate. This is usually solved by using specialized workstations with high processing powers. Since a processor must analyze the given command that points to an address which contains a set of instructions to be executed to perform the parent command, it makes the simulation a cumbersome process when a larger network is involved. But the time consumed is the overhead that has to be compensated in some way or other.

To counter this increasing complexity, designers are adopting a "design-to-test" methodology. They are benefiting from the following: the bit-true simulation of Model-Based Design executable models, which identify and fix flaws during design; automatic code generation to quickly prototype designs; and the use of the original executable models to directly validate final implementations through co-simulation.

II. C /RTL CO-SIMULATION

A. What is it?

A Co-simulation allows simulating an entire system whose parts have been designed in different languages, say SystemC and HDL. Simulating both the codes in their respective environment must give a coherent and correct simulated output. In co-simulation the distinctive subsystems, which in itself forms a coupled problem domain, are modeled and mimicked in a distributed way. System modeling is done on the subsystem level without having the coupled problem domain at the top of the priority list. Moreover, the coupled simulation is completed by running the subsystems in a black-box manner. Amidst the simulation, the subsystems will trade information.

B. C-based Co-Simulation/RTL-based Co-Simulation?

One important point to remember is that a C based co-simulation in HDL requires a test bench which verifies the C program with a set of values. When creating a test bench in C or another language to test and validate a device under test(DUT) written in VHDL or Verilog, the designer must rely on co-simulating the two environments. The disadvantage of maintaining test benches in HDL is that the verification is very slow due to the inbred nature of HDL simulators.

Comparatively, in a RTL-based co-simulation, although an RTL verification cannot be avoided, the benefits of improving the availability of design resources, after a successful RTL verification, can be plenty including availability of more design resources, fewer errors, faster time to market and lower overall costs.

C. What to use and why?

Many EDA vendors, such as ARM, Synopsys, and CoWare, provide ESL cosimulation tools. These system-level cosimulation tools enable designers to simulate the software part on the microprocessor model along with the hardware part designed in RTL or SystemC. In these cosimulation tools, TLM is widely used for simulation acceleration by a factor of 1,000 to 10,000. Also, their debugging and profiling capabilities provide a very powerful cosimulation environment for system-level architecture exploration and pre-silicon-embedded software development.

D. Configuration problems

It is not easy to configure a system using SystemC or SpecC because there are too many components to describe in a system.

Revised Manuscript Received on October 05, 2019.
* Correspondence Author
S.Karthik, Department of ECE, SRMIST, Vadapalani, Chennai India. Email:skarthiksrmvdp@gmail.com
K.Priyadarsini, Department of CSE, VISTAS, Pallavaram, Chennai India. Email:priyadarsinikk@gmail.com

For easier system implementation using SystemC or SpecC, a special approach called Electronic System Level (ESL) design has been employed. ESL describes a SoC design in a sufficiently abstract and high-level fashion to quickly explore the design space and provide virtual prototypes for system implementation in hardware and software.

Co-Simulation plays an important role in checking the ESL design correctly. The system defined in both SystemC and HDL is verified for their code integrity, failing which the design is termed faulty or consisting of errors.

III. PERFORMING C/RTL CO-SIMULATION

Before we begin to perform a co-simulation we must perform a few other operations.

First is to acquire a C code file, add the header files, add the test benches related to the C code and add constraints. The C test bench, along with constraint file added in the beginning are vital to Co-simulation as it allows easier evaluation of the designed system. Instead of going through all the process, we can write a .tcl script to automate multiple processes.

After adding C code, constraints and header files, we must do a C simulation. The C simulation checks for any code error or integrity issues with the code and might warn, if any, about it. Given below shows the result of a C Simulation.

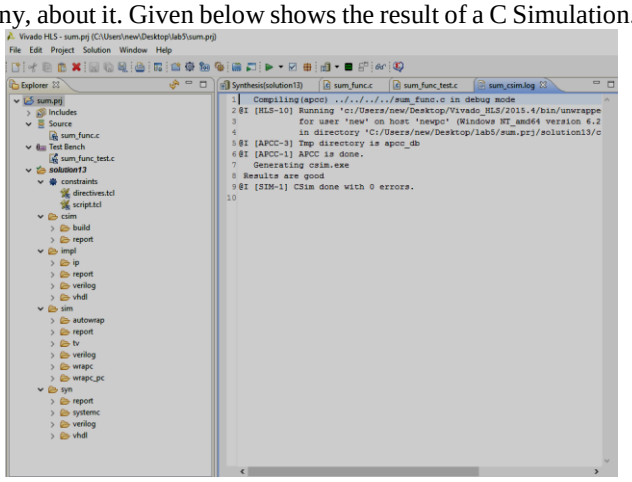

Fig.1. C Simulation Result

Note: The above diagram shows the result of the C simulation which is successful in status.

After a successful C simulation, a C synthesis is performed. The C synthesis gives a detailed report of resources involved and used to execute the program. Given in Figure 2, we can see the resource utilization for implementing an FIR filter. Parallely, the performance of the code or system can also be checked in Figure 3.

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	-	-
FIFO	0	-	65	292
Instance	-	15	746	915
Memory	-	-	-	-
Multiplexer	-	-	-	-
Register	-	-	6	-
Total	0	15	817	1207
Available	280	220	106400	53200
Utilization (%)	0	6	~0	2

Fig.2. Resource Utilization

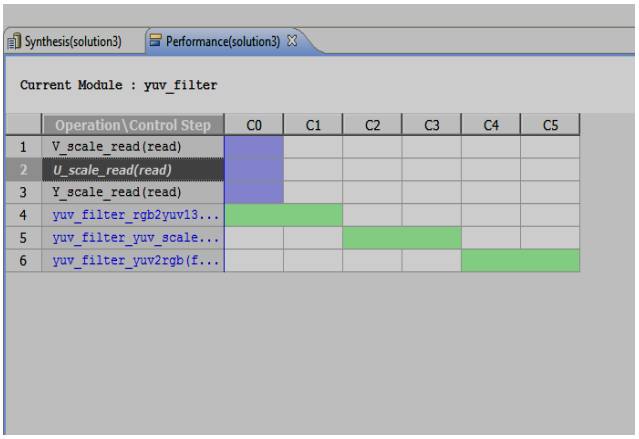

Fig.3. Design Performance

Figure 2 shows the resources used by the code of YUV filter. There are 53,200 LUTs or Look-Up-Tables available in the Targeted device. There are 106400 Flip Flops available to use. Likewise, many other resources of the targeted device are listed and their usage profiled. This report is acquired after C synthesis and updated after C/RTL Co simulation with Performance report.

Figure 3 shows the pipelining of the processes involved in the system design code. C0, C1, C2, etc are system cycles in which the instructions work. Pipelining the processes in a dense manner could reduce the time taken to simulate the design but would increase the resource usage.

The time-resource conundrum must be taken into consideration while designing a system in HDL. Any time crunch could lead to slower process handling while a resource crunch could give multiple issues.

The C/RTL Co simulation performed after a successful C synthesis, simulated the entire design that is spread in C code and HDL code as one system. The result gives performance reports and shows how the code lagged in certain areas and how the code excelled in certain other areas.

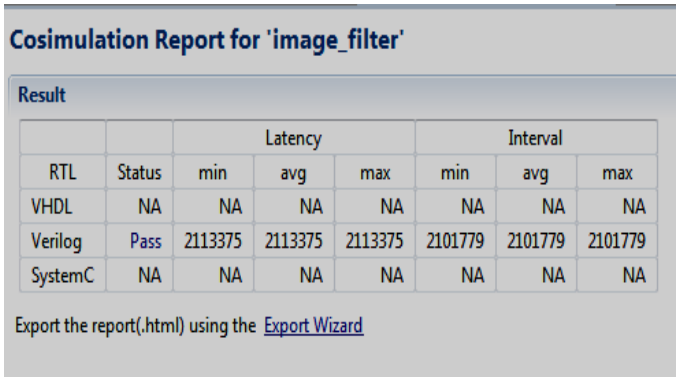

Fig.4. Co-simulation Result

Note:

1. During C/RTL Co simulation, static elaboration of modules is done and the software performs a simulation data flow. The modules created by running the TCL file or manually in the software are compiled.
2. The TCL file is an executable or rather a script within the Vivado tool, that is used to perform functions that would otherwise take multiple, painstaking and long steps. A TCL file can be written using a simple notepad application and saved with a .tcl extension.
3. The system can be viewed as per each discrete time frame as well. Given below is the diagram that shows the inputs

(as per test bench) and outputs over a time period.

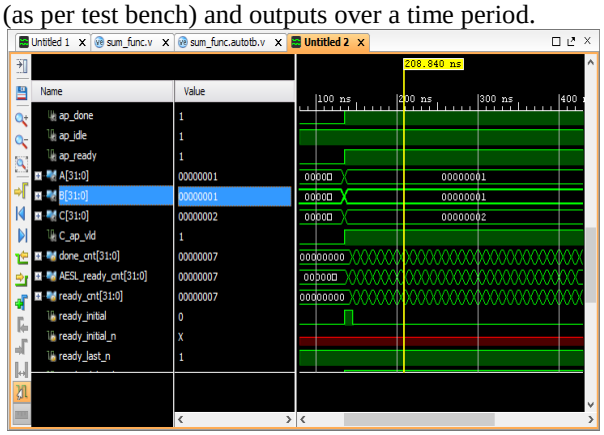

Fig.5. System Graph

The Co-Simulation report acquired can be used for various analysis and system optimization. It can also be used to find faults within the system by comparing the desired output with received output of the system for a given test bench.

IV. SYSTEM SIMULATION TIME

System complexity introduces delay in simulating the system for verification. To hasten the process of simulation, a heterogeneous architecture based processing system can be designed. Using a heterogeneous architecture, a program can be assigned to the processor and/or the FPGA respectively.

Using an FPGA, a conglomerate of certain logic functions such as AND, OR, etc can be realized with least effort when compared to a processor. The functions that take more time can be assigned to the FPGA while the functions that take least time in a processor can be assigned to the processor. This analysis is done by using a method called Profiling.

Profiling by definition is a software-intrusive tool that is used to analyze parameters of a program which is to be optimized. Program analysis tools such as Xilinx SDK are extremely important for understanding program behavior on various architectures of boards and their prototypes. Using SDK application, a person can profile a program running on embedded hardware.

Profiling provides two kinds of information that a person can use to optimize the program:

- A histogram with which you can identify the functions in the program that take up the most execution time
- A call graph that shows what functions called which other functions, and how many times

The future possibility of such an arrangement can allow speed-simulation, faster processing times and lesser turnover times in the testing and prototyping of devices in electronics sector.

V. RESULTS AND CONCLUSION

The first potential use of Co-simulation could be for improving the speed of simulation in Portable Electronics sector.

Many companies that design electronic devices rely on simulation to check if the system has flaws. These simulations take huge time and thereby reduce the turn-around time for the company. By using Co-Simulation, the company can research further into reducing Simulation times by the usage of Heterogenous architectures or Multi-Threading. Hence, reduction of time leads to decrease in Time-To-Market, thereby more profit indirectly.

The second potential use is in Embedded Electronics sector and Industrial Automation sector which rely heavily on electronics. Co-simulation would allow system engineers to target areas of industry, which was previously overlooked due to complexity, environmental, practical usage and size issues.

This also includes the medical equipment manufacturers and designers that can use Co-simulation to design more complex electronics systems with a simpler algorithm. Co-Simulation with Profiling could be the tools that would enable faster time-to-market and profit the electronics industry. It would also reduce the probability of errors that could lead to product failure and product call-back.

REFERENCES

1. Nihal Kulkarni, “Electronic Circuit Design: From Concept to Implemetation,”CRC Press, Florida. Pp 430-439, 2008
2. https://en.wikipedia.org/wiki/Co-simulation
3. Sicklinger, S.; Belsky, V.; Engelmann, B.; Elmqvist, H.; Olsson, H.; Wüchner, R.; Bletzinger, K.-U. (11 May 2014). "Interface Jacobian-based Co-Simulation". International Journal for Numerical Methods in Engineering. 98 (6): 418–444. doi:10.1002/nme.4637
4. Electronic Design Automation For Integrated Circuits Handbook, by Lavagno, Martin, and Scheffer, ISBN 0-8493-3096-3, 2006
5. Combinatorial Algorithms for Integrated Circuit Layout, by Thomas Lengauer, ISBN 3-519-02110-2, Teubner Verlag, 1997.
6. The Electronic Design Automation Handbook, by Dirk Jansen et al., Kluwer Academic Publishers, ISBN 1-4020-7502-2, 200
7. http://www.staticfreesoft.com/documentsTextbook.html: Computer Aids for VLSI Design by Steven M. Rubin
8. Michel Robert; Bruno Rouzeyre; Christian Piguet; Marie-Lise Flottes(5 December 2001). “SoC Design Methodologies: IFIP TC10/WG 10.5 Eleventh International Conference on Very Large Scale Integration of Systems-on-Chips(VLSI-SOC’01)” Montpellier, France.

AUTHORS PROFILE

S.Karthik has completed his PhD in the area of VLSI.He has published more than 20 international journals.His research area includes High Performance computing, Reconfigurable architecture,VLSI Testing,DFT,Low power VLSI techniques. He has attended many conference and workshops throughout the world.He is well versed in HDLs like Verilog,VHDL.He has worked in many development board like PI,PYCOM,ZED.He was actively involved in projects such as low power pattern generation for BIST architecture, fault-tolerant architectures, dynamic partial reconfiguration in FPGAs, and low power adders. He believes in life-long learning. To update his skill set he has attended numerous training programs/workshops such as the Cadence training program, workshop in FPGA Based System Design using ALTERA EDA Tools and in Advanced Engineering Optimization and Modeling using MATLAB & SCILAB, just to name a few.

K.Priyadarsini has completed her M.Tech from VIT University. She was an Intern at CTS. She has worked in many IT companies has developer. She started her career at SRM University. She did her PhD from VISTAS. Her area of research includes High performance cloud computing, security aspects at Multicloud, data science. Data mining,Big data analytics. She has completed many online MOOC courses like Python for Data Science, Cloud computing . She is interested in Object Oriented Programming and carrying out workin building software models . She has published more than 10 international journals and she has presented many papers in conferences.