International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 04 Issue 06
May 2017

Optimizing The Virtual Prototype To Speed Up Simulations

Chandini S P
Department of electronics and communication
SJCE, Mysore
Email: chandinisp@gmail.com

Dr. S Gayathri
Department of electronics and communication
SJCE, Mysore
Email: sgmurthy_65@yahoo.com

***Abstract*—we propose a generalized approach for optimizing speed of virtual prototype. The fast execution of virtual platform depends on its simulation performance. The main objective is to optimize simulation performance of virtual platform by improving its simulation speed in Virtualizer. We explain measuring simulation performance of virtual platform, identifying factors affecting the simulation performance and techniques used for performance optimization. The methodology implemented here can be followed for performance optimization of any complex virtual platform.**

Keywords—virtual prototype; simulation; modem;

I. INTRODUCTION

There is a significant raise in the number of cores and the processing power. The number of cores in a computing device is increasing rapidly and thus increasing the total device complexity. Accordingly, developing software for these complex multi-core systems is a great challenge than building hardware. The amount of interface between the cores is growing as well, meaning that there are many more opportunities for problems to exist in temporal interactions, resource conflux and performance issues. Time to market is decreasing with expected increase in complexity As a result, the software development and verification became an essential part of the product development life cycle. In order to stop software being on the critical path of product development it is necessary to find some way of enabling software development to be conducted in parallel with the hardware development.

A virtual prototype is a software- simulation- based, architectural- level model of the installed framework. Since this model has same limits as the equipment model, real- world abilities and effects can be shown. This ensures the equipment models will work when built and limits the hardware- software consolidation effort in the late periods of the equipment extending validation scope. The framework model can fuse processors, transports, equipment fringe portions and even models of modem subsystems that are a bit of the general framework. The virtual model transforms into the splendid reference for the hardware and programming headway effort. The processor models in the virtual model are solidly associated with code. Right when programming updates are consolidated, similar sets that will continue running in the previous model can be executed inside the virtual model. This grants system designers to survey hardware structures by running sensible programming loads. It moreover lets programming specialists to research their code using building models, much sooner than the low down RTL design is done, thus enabling parallel development of hardware equipment and programming.

II. ANALYSIS OF DIFFERENT METHODS TO REDUCE DEVELOPMENT TIME

A. Use a previous generation product

Sometimes a previous generation product can be used if it is similar enough and the capability patched in using a make shift prototype. This can be expensive and still relies on the hardware availability for new capabilities.

B. Create an early prototype using FPGA

In some cases emulation, rapid prototyping using FPGAs can make the hardware available beforehand. They are also bit true and cycle accurate. But this is still after hardware design has been completed and can be expensive.

C. Virtual prototype

A virtual prototype is a software- simulation- based, architectural- level model of the embedded system**.** This method provides fast prototyping without much compromise in accuracy.

III. PLATFORM CREATOR TOOL ANALYSIS

Virtualizer provides tool for the creation, distribution and use of virtual prototypes. Virtualizer makes virtual prototyping easy and of immediate value to software developers. Virtualizer tool supports lot of new features including profiling where run-time performance can be measured during IP development itself. Virtualizer is a set of tools: VP Explorer and VP Analyzer.

International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 04 Issue 06
May 2017

A. *VP Explorer*

It is an Analysis and debug tool for the platform level. One can launch VP Explorer standalone and open a project based simulation or can use platform creator to build the simulation and start VP Explorer.

VP Explorer is primarily intended for:

- Running and debugging SystemC- based simulations in depth.
- Controlling simulation execution.
- Tracing and analyzing simulation output during or after simulation.

B. *VP Analyser*

It helps in debugging hardware and software designs. It gives access to hardware registers used in the peripherals and the CPUs.

Main Features of VP Analyzer:

- Control a simulation and its processor cores
- Investigate the SystemC hardware hierarchy of the simulation
- Create customized views on the simulation
- Investigate and modify values of hardware registers and memory
- Look at the status of processor cores through hardware registers and the Disassembly view
- Use breakpoints and watch points

IV. PERFORMANCE ANALYSIS AND IMPROVEMENTS

ISCTLM offers a complete situation for the era of SystemC/C++ based TLM equipment models. It can be viewed as a reflection level above C++/SystemC that offers extra dialect builds and exemplified usefulness to encourage TL show advancement and use. ISCTLM executes the base convention initiator and target attachments and furthermore offers essential TLM part usefulness for registers, nearby recollections, transport API, address deciphering, and troubleshoot capacities. These libraries are the reason for the improvement of VP. We presented a few improvement upgrades in our library to support the speed of VP stages.

A. *DMI implementation*

The Direct Memory Interface, or DMI, gives a methods by which an initiator can get immediate access to a region of memory possessed by an objective, from that point getting to that memory utilizing an immediate pointer as opposed to through the transport interface. The DMI offers a huge potential increment in recreation speed for memory access among initiator and target in light of the fact that once settled it can sidestep the ordinary way of different b_transport or nb_transport calls from initiator through interconnect segments to target.

1) DMI for memory

DMI streamlines the asset get to, offering a rapid up potential for the model creator since DMI gives direct access to the physical memory range of an asset instantiated in a target. There are two direct memory interfaces, one for approaches the forward way from initiator to target, and a moment for approaches the regressive way from target to initiator. The forward way is utilized to ask for a specific method of DMI get to a given address, and returns a reference to a DMI descriptor of sort tlm_dmi, which contains the limits of the DMI area. The retrogressive way is utilized by the target to invalidate DMI pointers already settled utilizing the forward way.

The forward and in reverse ways may go through zero, one or many interconnect segments to the forward and in reverse ways for the comparing transport calls through similar attachments DMI gives forward way (initiator – target) and in addition in reverse way (target – initiator) interface strategies. The forward way is executed utilizing get_direct_mem_ptr() technique and the regressive way is actualized utilizing invalidate_direct_mem_ptr().

2) DMI for registers

DMI for registers empowers indirect access gets to between the registers and the center. DMI solicitations to any asset will be acknowledged just when DMI permitted is valid. As a matter of course, just Memory objects have DMI enabled set to genuine. Registers and Address Spaces must be empowered unequivocally by setDmiAllowed(true).

B. *Enabling Temporal decoupling for target sockets*

In a model, the hold up () calls are constantly costly in light of the fact that they give unequivocal synchronization focuses in SC_THREADS and include costly setting switch in the SystemC part. The hold up () calls ought to be stayed away from wherever conceivable. Worldly decoupling grants a critical recreation speed change by decreasing the quantity of setting switches and occasions. Fleeting decoupling component in SystemC procedures to keep running in front of reenactment time for a measure of time known as the time quantum, and is related with the inexactly planned coding style. When utilizing fleeting decoupling, the defers

commented on to the b_transport techniques are to be deciphered as neighborhood time balances characterized with respect to the present recreation time as returned by sc_time_stamp(), otherwise called the quantum limit. The worldwide quantum is the default time interim between progressive quantum limits. Of course, transient decoupling is not empowered for target attachments. It must be asked for unequivocally by calling mSocket.setTemporalDecoupling (genuine).

For most extreme reenactment speed, all initiators ought to utilize transient decoupling, and the quantity of other runnable SystemC procedures ought to be zero or limited. In a perfect situation, the main runnable SystemC procedures will have a place with transiently decoupled initiators, and each procedure will keep running ahead to the finish of its time quantum before respecting the SystemC part. A transiently decoupled initiator is not obliged to utilize a period quantum if correspondence with different procedures is unequivocally synchronized. Worldly decoupling keeps running with regards to the standard SystemC reenactment bit, so occasions can be planned, forms suspended and continued, and approximately coordinated models can be blended with other coding styles.

V. CONCLUSION

Virtual prototyping is the most ideal approach for achieving time to market. It empowers simultaneous hardware and programming improvement which meets advancement plans. The procedure of parallel simulation can be connected just when there is less information activity between the partitioned platforms. In complex frameworks, this may not be the situation, there exists a considerable measure of information sharing among the parts. The usage of parallel simulation can be stretched out to stages which share an immense measure of information.

References

[1] Damstra, A. (2008). Virtual prototyping through co-simulation in hardware/software and mechatronics co-design. In A. Damstra.

[2] Donovan, D. C. (n.d.). Systemc: From The Ground Up. Eklectic Ally, Inc. Kluwer Academic Publishers.

[3] Inter-Process Communication Mechanism. (2016, Feb 14). Retrieved from MSDN: http://msdn.microsoft.com

[4] Jason R. Ghidella1, A. W.-F. (n.d.). The Use of Computing Clusters and Automatic Code Generation to Speed-up Simulation Tasks. Retrieved from MathWorks: http://www.mathworks.com/tagteam/44587_Paper_AIAA07_Accel_Simulations.pdf

[5] Jose J.Blanco-Pillado, K. D. (2010, Nov 17). A new parallel simulation technique. Retrieved from http://arxiv.org/abs/1011.4046