ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 3, Issue 4, July 2014

Simulation Performance Optimization of Virtual Prototypes

Sammidi Mounika, B S Renuka

Abstract— Virtual prototyping is becoming increasingly important to embedded software developers, engineers, managers and marketing teams to enable concurrent hardware and software development. The software teams need an extremely fast execution virtual prototype to complete software development as early as possible in the product development life cycle before the silicon is available. The fast execution of virtual prototypes depends on their simulation performance. This paper introduces a method to optimize the simulation performance of virtual prototypes. This method uses the technique of parallelizing the simulation. The rate at which the simulation speed gets increased is also discussed in the paper.

***Index Terms*—Concurrent hardware and software development, Optimization, Parallelizing, Simulation performance, Virtual prototyping**

I. INTRODUCTION

In Today's embedded systems, the number of cores is increasing rapidly to add additional performance [1]. The software coding complexity is increasing with the increase in number of cores. And the software necessary for the multi-processor systems is becoming highly interactive The increasing coding complexity and development costs are making the software development an essential and large part of a chip design. Accordingly, System on Chip (SoC) design teams needs to spend more time writing software than building hardware. With multi-core chip sets and with the volume of software to be developed, the software development needs to start as early as when the chip is specified. Virtual prototypes (VPs) are one of the tools that software engineers are increasingly turning to for some of the solutions that they need to test and debug the software and in some cases virtual prototypes are the only solutions that can provide an answer. Even the real hardware cannot address some of them. Virtual prototypes can be made available just a few weeks into the project schedule, which allows the software team to begin porting operating systems and developing device drivers without having to wait for the hardware team to write a single line of Register-Transfer-Level (RTL) code [2].

Virtual prototypes should be fast enough for the software development to get completed much before the silicon availability. As a result, optimization of simulation performance of virtual prototypes has become an increasingly important area to be explored. The fast execution platform helps in reducing the time for testing and debugging the complex software.

II. PERFORMANCE OF VIRTUAL PROTOTYPES

One of the parameters that affect the performance of a virtual prototype is its simulation speed. The simulation speed of virtual platform depends on many factors like Host machine and cache, ISS speed, abstraction level, temporal accuracy, coding style, compiler, design complexity, cache/MMU models etc. It is likely that factors such as cache/Memory Management Unit (MMU), Instruction Set Simulator (ISS) speed and compiler are predetermined for a Host machine and are unmodified. The design complexity and temporal accuracy depend on VP model development and are likely to be unavoidable [1]. Hence, a new methodology is to be implemented to improve the performance of virtual platform.

III. TRADITIONAL METHODS

The simulation speed of virtual prototypes has been increased by implementing following methods.

A. Improve speed of the simulator

This method increases simulation speed by providing a faster execution engine. The speed of execution of simulator is fixed for a specific computing platform.

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 3, Issue 4, July 2014

B. Run less simulation

One of the most interesting concepts in SystemC TLM-2.0 is the concept of Direct Memory Interface (DMI) [3]. The great feature of Seamless was to simulate less by using backdoor memory accesses to skip simulation of bus transactions.

DMI used with SystemC simulation falls into the category of *run faster by simulating less*. It uses direct access to memory data (via pointer dereferences) and avoids the overhead of function calls to retrieve data from memory and peripheral models. In the 1990's, co-verification tools used back door memory accesses to avoid Verilog and VHDL bus transactions. SystemC TLM-2.0 does not use detailed bus protocols at the signal level; it uses C++ function calls between models. On the surface, using function calls sounds pretty fast compared to using a signal-based bus model with clock, bus request, grant, address phase, data phase, etc.

With DMI, the simulation runs at a speed in which the simulated time is about equal to the wall clock time. The reason for such a high speed is that using TLM-2.0 function calls forces the CPU to break out of its blazing fast execution for all instructions that access memory. This cripples the entire effort processor model creators put into making the instruction translation so fast. It also demonstrates that even function calls take time when billions of them are required to run 2.5 billion instructions. Simulating invisible activity is a waste of time.

Of course, simulating less also has drawbacks. One difficulty of DMI is that it is so abstract that there is no visibility into what is happening. In fact, DMI is pretty much invisible. Nothing is visible when the simulation is running. This leads to the second challenge that simulations using DMI can be hard to debug. If the setup is not correct, strange things can happen. . If the transactions which set up the DMI address ranges are not done correctly, the result can be very ugly. The end result is memory corruption that is hard to identify.

To help with debugging, the ability to monitor DMI activity and print the DMI memory map is very useful. A new method is going to be discussed in this paper which avoids the disadvantages of traditional methods for speed optimization. This method uses the techniques of parallelizing the simulation.

IV. PARALLEL SIMULATION

The concept of parallel simulation is proposed for partitioning the virtual platform into two executables. The two executables are run in parallel and communication is established between these two executables through Inter Process Communication (IPC) mechanism [4]. IPC facilitates the division of labor between the two executables making them run separately. Parallel simulation allows two simulations to run individually and uses IPC mechanism to enable data sharing when synchronization is needed between two simulations. TCP/IP socket API can be used for incorporating IPC between two simulations [5].

Virtual platforms are developed in systemC language which supports single threaded mechanism where the flow of execution is sequential [6]. Parallel simulation is suitable for a multi-processor system with single simulation kernel where the processors stay in idle state waiting for their turn of execution to occur (till other processors finish their task). The simulation speed advantage can be gained for a multi-core system by making the cores run individually [7], [8].

Parallel simulation is not suitable in cases where there is more number of synchronization points during the course of simulation. In such cases, the usage of parallel simulation incurs overhead (as data transfer takes more time with parallel simulation than unparalleled simulation) reducing the speed of simulation. The overhead is due to the fact that data has to pass through additional components such as connector blocks/wrappers incorporated for enabling communication between simulations running in parallel.

A. Applicability of Parallel Simulation

Parallel Simulation works at its best when

1) The subsystems are loosely coupled

2) Inter-subsystem interface has following characteristics

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 3, Issue 4, July 2014

i. Asynchronous behavior

ii. Low traffic

Speedup depends on parallelism between subsystems. If characteristics are not met, applying parallel simulation method for speed optimization may not pay off. The platform could slow down and not work; communication latency may break synchronous interfaces

B. IPC through TCP/IP Socket API

TCP/IP socket API is used to connect one VP to another VP/application running on same/different operating system (Windows/LINUX). It has its own interface classes to make the communication possible between two VPs. It is implemented in C++. It uses client-server mechanism. One VP acts as client and the other as server. Parallel simulation supports TCP/IP, named pipe and shared memory depending on the type of data being transferred.

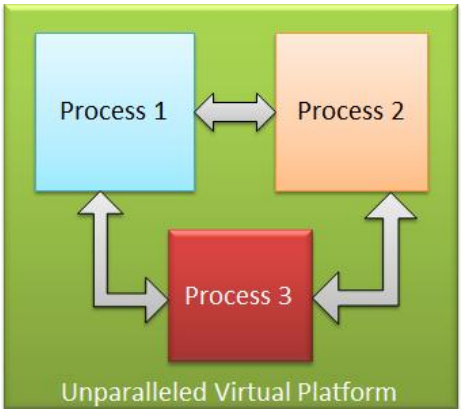

Fig.1 Unparalleled Virtual platform

The Fig.1 shows monolithic systemC simulation of virtual platform running as a single simulation thread. The processes are run one after the other. The processes are directly connected to each other. In the context of parallel simulation, the platform is divided into sub systems/processes and subsystems are run in parallel as shown in Fig.2. The communication between these processes is established using TCP/IP socket API. The connector block/wrapper acts as user API. The connector block of one subsystem is connected to the connector block of other subsystem through TCP/IP scoket API.

Connector blocks consist of input/output ports, TLM sockets and a clock. They implement the methods that are declared in the TCP/IP socket API to send/receive data/messages to/from other connector blocks. TCP/IP socket API converts the transactions such as read/write into messages that can be sent or received though IPC mechanism over a TCP/IP socket or a named pipe.

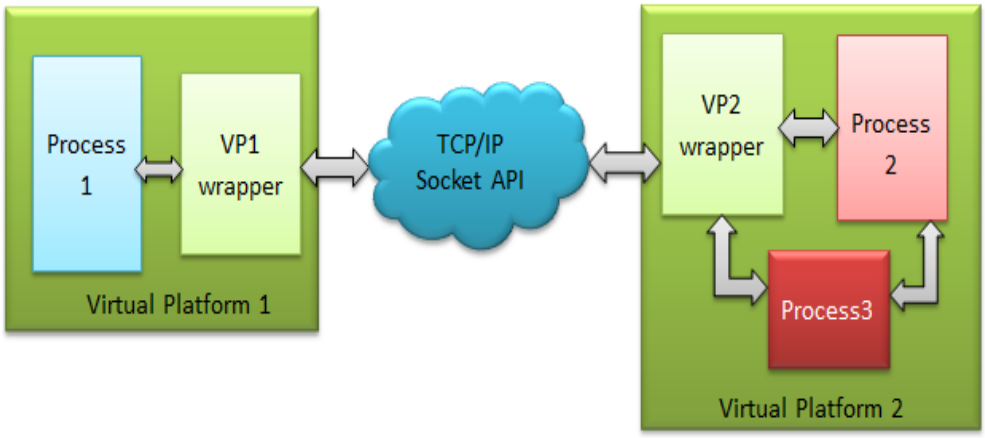

Fig.2 Parallel Simulation

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 3, Issue 4, July 2014

C. Benefits of parallel simulation

1) Leverages multi core host machines for speed
2) Existing models can be used without modifications

V. RESULTS

The method of parallel simulation has been implemented for a virtual prototype of an embedded system. Fig.3 shows relative speed up with the help of parallel simulation. The data exchange between two simulations happens till time point 'A' and from time point 'B' to time point 'C'. There is no exchange of data from 'A' to 'B'. In the graph, the thick line represents the simulation without parallelization and the thin line represents parallel simulation. The slope of the graph for a given period indicates the speed of simulation during that period. Lesser the slope more the speed of simulation i.e. the real time or wall time should be less for a given period of simulation.

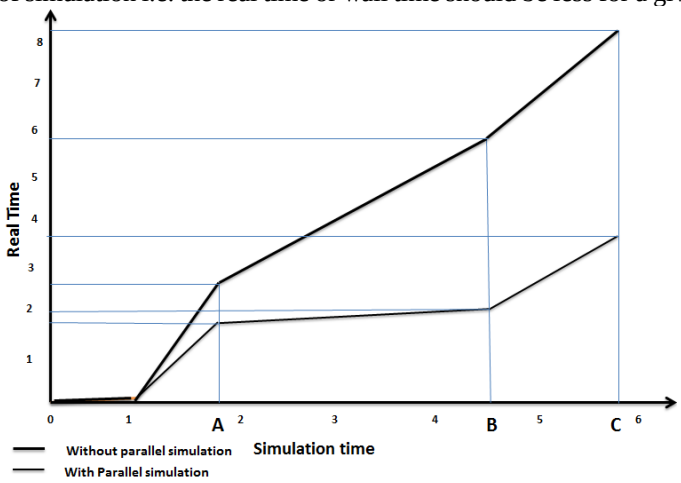

Fig.3 Relative speed up with parallel simulation

From the graph, it is observed that the slope is less between time points 'A' and 'B' indicating that the simulation speed is high during this period of simulation. The overall speed achieved through parallel simulation is 2.5 of that of unparalleled simulation.

VI. CONCLUSION

The proposed method has been experimented in view of optimizing the performance of virtual prototypes. From Results, we can conclude that the proposed method will contribute 2.5 times of speed improvement. The proposed method enables to simulate complex virtual platforms in a faster and more effective way and hence the software/firmware development will be faster. This methodology enables leading semiconductor and electronics companies to deliver more competitive and higher quality products up to 12 months faster.

REFERENCES

[1] Bryan Schauer "Multicore Processors – A Necessity" released September 2008 [online]. Available: http://www.csa.com/discoveryguides/multicore/review.pdf

[2] Arjen Damstra "Virtual prototyping through co-simulation in hardware/software and mechatronics co-design" released on April 2008.

[3] OSCI TLM-2.0 Language Reference Manual, Software version: TLM 2.0.1, Document version: JA32.

[4] Inter Process Communication [online]. Available: http://msdn.microsoft.com/en-us/library/windows/desktop/aa365574(v=vs.85).aspx

[5] Rajinder Yadav "Client / Server Programming with TCP/IP Sockets" released on Sept 9, 2007 Revision: Mar 11, 2008.

[6] IEEE Standard SystemC® Language Reference Manual, IEEE Computer Society Sponsored by the Design Automation Standards Committee.

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 3, Issue 4, July 2014

[7] Jose J.Blanco-Pillado, Ken D.Olum, and Benjamin Shlaer "A new parallel simulation technique" Submitted on 17 Nov 2010 [online]. Available: http://arxiv.org/abs/1011.4046

[8] Jason R. Ghidella1, Amory Wakefield2, Silvina Grad-Freilich3, Jon Friedman4 and Vinod Cherian5 "The Use of Computing Clusters and Automatic Code Generation to Speed up Simulation Tasks" The Math Works, Inc. Natick, MA, 01760 [online].Available: http://www.mathworks.com/tagteam/44587_Paper_AIAA07_Accel_Simulations.pdf

AUTHOR BIOGRAPHY

Sammidi Mounika
Industrial Electronics, M.Tech,
Sri Jayachamarajendra College of Engineering, Mysore, Karnataka, India.

B S Renuka
Associate Professor, Department of Electronics and Communications Engineering,
Sri Jayachamarajendra College of Engineering, Mysore, Karnataka, India