

Multi-threaded Simulation: A Viable Approach to Completely Eliminate Communication and Synchronization Overhead?

Seiyang Yang, Hwachin Lee, Dusung Kim*, Maciej Ciesielski**

Dept. of Computer Eng., Pusan National University, Korea

* Dept. of ECE, University of Massachusetts, Amherst, USA

** LogicMill Technology, LLC., USA

Abstract – *Parallel simulation methods suffer from serious communication and synchronization overhead, which significantly reduces or negates the potential performance gain offered by parallel simulation. This paper introduces a new, possibly controversial, approach to distributed parallel simulation, based on temporal parallel simulation, aimed at completely eliminating the communication and synchronization overhead between processors. In contrast to traditional (spatial) distributed simulation, which partitions the design into multiple modules and simulates all modules in parallel, the proposed multi-threaded simulation partitions single simulation into multiple simulation runs in temporal domain.*

1 Introduction

With the increasing size and functional complexity of SoCs, the complexity of verification task is exploding accordingly. Since simulation remains the mostly widely used technique for functional and timing verification but its speed is severely lagging behind the performance requirement, there has been a great demand to increase simulation performance. Historically there have been three approaches to achieve it. First, based on HW-assisted simulation acceleration, relies on special hardware platform on which the synthesizable portion of design is emulated and executed.[1] The second method, distributed parallel simulation, partitions the design into separate design modules and performs concurrent simulation using multiple HDL simulators.[2] The third, most recent method, uses more abstracted design models for simulation. Cycle-based simulation or transaction-level simulation are examples of this approach.[3]

In this paper, we propose a new approach to distributed HDL simulation, termed *multi-threaded simulation*, which is based on the concept of temporal parallel simulation (patent pending). We believe that this new approach has great potential to achieve high verification productivity. In this paper we will outline the concept of temporal parallel simulation and show that it is compatible with the current and future design and verification flows. Then we will discuss issues that need to be resolved to make this method practical and show some preliminary experimentation results.

2 Temporal Parallel Simulation

The temporal parallel simulation presented in this

paper is a radical departure from the conventional parallel HDL simulation, which depends on spatial decomposition of the design to be simulated. In contrast, our temporal parallel simulation partitions the simulation in time, rather than in space. It consists of two major steps: the first, fast “high-grain” simulation run which stores essential information at selected checkpoints; and the second, low-level simulation, which is distributed to the individual processors. The first simulation must be run at the higher level of design abstraction (TLM, or RTL), and the second simulation must be run at the lower abstraction level (RTL or Gate Level). The designer must declare up front the size of the total simulation time (the number of cycles) and the number of processors. The entire simulation time is divided into slices, each to be executed on an independent simulator.

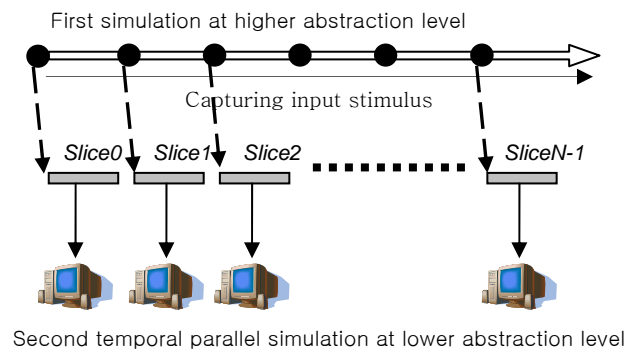


Fig. 1. Temporal parallel simulation

During the first simulation run, the design state is captured at the start time of each time slice. The design state consists of the state of all internal registers and memory print of the design. At the same time, the input stimulus of DUV (Design Under Verification) is continuously captured during the first simulation run. The captured input stimulus is then used as an input vector for each time slice during the second simulation, while the design state stored at each checkpoint serves as the initial design state for that simulation slice. As a result, the second simulation can be run independently for each slice.

The main idea of *temporal* parallel simulation is in line with the fact that all the current (and possibly future) design implementation flows are based on progressive refinement from high abstraction level to

low abstraction level. For example, the current de-facto standard design practice starts by developing a synthesizable model at a register transfer level (RTL). The RTL model is simulated to verify its functional behavior, and then it is automatically translated into a gate level (GL) model by logic synthesis tools. The gate level model is then implemented using physical (place and route) synthesis tools to obtain a GDSII layout format. During this last stage, the back-annotated delay information, e.g. SDF, is extracted. At this point, the gate-level, full timing simulation can be performed. Similarly, RTL model can be progressively refined from transaction level model (TLM).

Therefore, any simulation at lower abstraction level (this is, the second simulation run in our multi-threaded simulation scheme) can be performed in fully parallel fashion without incurring any communication or synchronization overhead, provided that a simulation at higher abstraction level (this is, our first simulation) allows to correctly partition the simulation at lower level abstraction in temporal fashion. This is possible if the model at the lower abstraction level is consistent with that of the higher abstraction level. We should note that the simulation time of our first simulation run should not be counted towards the total simulation time if the first simulation process is mandatory and is carried out at the higher abstraction level during the design refinement process. That is, if such a simulation has to be performed at that level, it can serve as the first simulation run for our multi-threaded simulation without additional time overhead.

3 Issues to Be Resolved

There are some important issues that need to be resolved when deploying the temporal parallel simulation in the design process. First of all, the lower abstraction model must be strictly consistent in terms of timing with its higher abstraction model. More specifically, RTL model should be consistent with its corresponding GL model and the TL model on a cycle-by-cycle basis. Any inconsistencies that may arise at cycle boundaries should be considered as design bugs that should be corrected. In other words, RTL behavior must not differ from the GL behavior and from the cycle-accurate TL behavior at cycle boundaries.

On the surface, this consistency requirement seems too restrictive. However, such a consistency requirement is naturally satisfied and enforced by the automated RTL to GL synthesis process. Also, if the consistency requirement is established among different abstraction levels, the RTL and GL verification closure can be achieved quickly because the temporal parallel simulation can greatly reduce the simulation time and automatic response checking environment can be easily constructed by utilizing the higher level abstraction model as a golden reference.

We believe that high simulation performance and automatic response checking capability provided by our temporal parallel simulation can result in significant increase in verification productivity.

4 Preliminary Experimentation

We applied our multi-threaded simulation to gate-level, full timing simulation. For this preliminary experimentation, a zero-delay gate level model was used instead of RTL as the corresponding higher-level abstraction model, even though it is not as fast as RTL. The second gate level timing simulation is split into 10 time slices with the gate-level design and SDF. Commercial Verilog simulators running on Linux machines were used for this experiment. We observed a 4x speed-up if the first simulation time is accounted for, and a 6x speed-up if only the second simulation is considered (neglecting the overhead of the first simulation run).

		Time (ratio)
Conventional gate-level timing simulation		
Multi-threaded Simulation (N=10)	1 st simul (zero-delay)	
	2 nd simul. (timing)	

5 Conclusions

This paper presents a radical solution to completely eliminate communication and synchronization overhead in distributed parallel simulation. It is accomplished by performing temporal partitioning of the simulation run, instead of spatial partitioning of the design into modules to be simulated individually. The simulation speed can be increased linearly, which was an unachievable dream in (spatial) parallel simulation, because of the unavoidable inter-simulation communication and synchronization overhead. To achieve this linear speedup, we may have to confine the implementation to a strictly progressive refinement process, in which lower abstraction model is strictly cycle-time consistent with its higher abstraction model. However, we believe that it is more than worthwhile because the temporal parallel simulation may make the fully automated and ultra fast dynamic verification possible by integrating the verification flow with implementation flow. The described method is a subject of a pending patent (2004).

References

- [1] Bauer, J. et al., "A Reconfigurable Logic Machine for Fast Event-driven Simulation," Proc. ACM/IEEE DAC, pp. 668-671, June 1998.
- [2] Fujimoto, R., "Parallel Discrete Event Simulation," Communications of the ACM, Vol. 33, Issue 10, pp. 30-53, Oct. 1990.
- [3] Ghenassia, F., *Transaction Level Modeling with SystemC*, Springer, Dordrecht, Netherlands, 2005.