A New Distributed Event-driven Gate-level HDL Simulation by Accurate Prediction

Dusung Kim[1] Maciej Ciesielski[1] Seiyang Yang[2]
[1]Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, MA, USA 01003
{dukim, ciesiel}@ecs.umass.edu
[2]Department of Computer Engineering
Pusan National University, Busan, Korea, 609-735
syyang@pusan.ac.kr

Abstract— **This paper describes a new and efficient solution to a distributed event-driven gate-level HDL simulation. It is based on a novel concept of spatial parallelism using accurate prediction of input and output signals of individual local modules in local simulations, derived from a model at a higher abstraction level (RTL). Using the predicted rather than actual signal values makes it possible to eliminate or greatly reduce the communication and synchronization overhead in a distributed event-driven simulation.**

I. INTRODUCTION

In order to handle the ever-increasing design complexity of Systems on Chip (SoC), engineers have explored a host of verification methods: event-driven HDL simulation, hardware-assisted simulation acceleration, emulation, prototyping, and formal verification. Among those, event-driven HDL simulation remains the most widely used technique for functional and timing verification and will remain such for a foreseeable future. Simulation owes this popularity to its many advantages: ease of use, readily available simulation software, inexpensive computing platforms, controllability, 100% signal visibility (essential for efficient debugging), etc. However, event-driven HDL simulation suffers from very low runtime performance (even less than single cycle per second) dictated by its inherently sequential nature.

Distributed parallel simulation [1][2] has been proposed to address the low performance of HDL simulation. So far, however, most of the previous methods have not been successful for several reasons: 1) partitioning of design under verification (DUV) into local modules is a known very difficult and design-dependent problem; and 2) an inherent dependence between the modules in DUV imposes heavy synchronization and inter-module communication overhead. These overheads are particularly serious in gate-level (GL) timing simulation.

In this paper, we propose a new method that removes the major deficiencies of current distributed event-driven GL simulation by using highly accurate inter-module signal prediction. Our approach drastically minimizes the communication and synchronization overhead in distributed simulation, and is immune to sub-optimal partitioning. In theory, if the prediction is 100% accurate, the communication and synchronization overhead is completely eliminated, in which case even a linear speed-up can be achieved. In this method, each local simulation is first executed with the predicted inputs, rather than with the actual inputs from other modules in other local simulations. As long as the predictions are correct, each local simulation is executed totally independently from other local simulations, without incurring any communication and synchronization overhead. Only when the prediction fails, the communication and synchronization among local simulations is necessary; this happens when the simulation is resumed after having the design/simulation states properly restored. A detailed description of this new method is given in Section III.

II. PREVIOUS WORK

A rich body of literature exists in the area of parallel simulation, known as Parallel Discrete Event Simulation (PDES) concept [8]. Chamberlain [6] discusses several issues related to this concept, such as partitioning, synchronization, and granularity.

There are basically two types of synchronization methods (protocols) in PDES: conservative (lockstep-base) and optimistic (rollback-base). It is known that the optimistic method is typically faster, although its performance is strongly design-dependent. The performance depends on the number and granularity of the inserted checkpoints (at compile time), the number of actual rollbacks (at run time), and many other factors, such as the type of design, the way it is partitioned, and how often the communication and synchronization between the modules actually occurs.

Gafni [9] used state saving concept and rollback mechanism by restoring the saved state. Time Warp [7] (the “optimistic” approach) was able to reduce message passing overhead by using shared memory, although this method was not initially intended for event-driven HDL simulation. Fujimoto [8] and Nicol [12] improved the “conservative” method by introducing the concept of lookahead.

Bauer [4] proposed several improvements over the standard Time Warp to increase the speedup, including incremental state saving, bounding window, and synchronization granularity; he also applied the concept to gate-level simulation. It achieved speedup between 2 and 4 over the sequential LDSIM simulator on 12 processors. Bagrodia et. al. [3] developed a parallel gate-level circuit simulator in the MAISIE simulation language and implemented it on both distributed memory and shared memory parallel architectures, achieving speedup of 2-3 on 8 processors. Lungeanu [11] proposed a “dynamic” approach, which combines conservative and optimistic approaches by switching between the two protocols depending on the amount of rollback. They demonstrated speedup of up to 11 on 16 processors on a circuit with 14k gates.

Li et. al. [10] claim to have developed the first Verilog distributed simulator even though it failed to get the desired performance improvement. Zhu et. al. [14] was able to achieve a considerable speedup improvement with a large gate-level decoder design. However, such a design is a special case that provides almost ideal partitioning, which is generally not possible to achieve.

Most of the results in this area have been demonstrated only on small to moderate size, single-clock designs that can be partitioned without incurring significant inter-module communication and synchronization. Only a few commercial products have been developed, including SimCluster [1] and MP-Sim [2], the latter one requiring a proprietary simulator. However, they have not attracted the expected attention of designers, due to their limited performance and scalability. Most recently, the parallel gate-level simulation methods using GPU (Graphic Processing Unit) have been proposed [5]. They are confined to gate-level simulation mode with zero delay only, and their performance is strongly dependent upon the type of design being simulated.

III. A New Distributed Parallel Event-driven Gate-level HDL Simulation

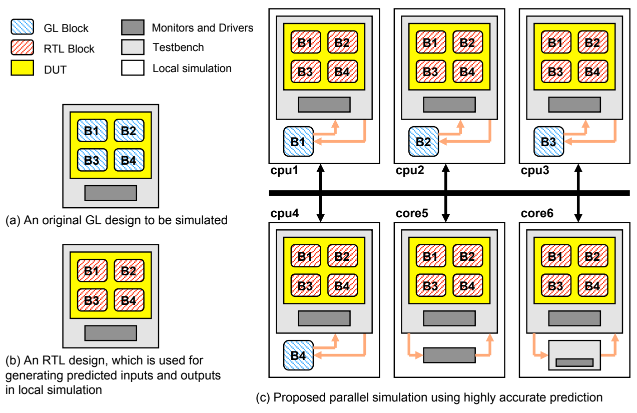
The main goal of the proposed parallel event-driven GL HDL simulation using highly accurate inter-module signal prediction is to make the simulation immune to suboptimal design partitioning and eliminate, as much as possible, computational overhead associated with data synchronization and inter-module communication. The simulation process consists of the following steps:

- Each local simulation is executed with the predicted input, without incurring any communication or synchronization cost for exchanging data with other local simulations. At the same time, each local simulation performs checkpointing for possible future rollback. (We will refer to this step as the <u>prediction phase</u>.)
- In the prediction phase, each local simulation compares the computed actual output with the predicted output. If the comparison fails, the failure notice and the current simulation time stamp are globally broadcast to all local simulations.
- Once the failure notice and the simulation time stamp are received, each local simulation rolls back to the nearest checkpoint. From this point on, each local simulation is executed with the actual inputs, incurring the communication and synchronization overhead for exchanging data with other local simulations. (We will call this the <u>actual phase</u>.)
- In the actual phase, each local simulation compares the actual input with the predicted input, and counts the number of matches. If the number of matches exceeds some predetermined threshold value in all local simulations, the actual phase is switched back to the prediction phase and the simulation continues in the prediction phase.

In this approach, the communication and synchronization overhead among local simulations occurs only during the actual phase, and there is no such communication and synchronization overhead during the prediction phase. Therefore, the success of this method depends entirely on the prediction accuracy. In theory, if the prediction is 100% correct, the communication and synchronization overhead is totally eliminated. To increase checkpointing efficiency, we could use snapshots of design state. Practically, obtaining a design state even for a large-scale design can be done quickly [15]. From a resource point of view, as all the globally passed checkpoints during the simulation can be removed, we don't expect to see significant resource consumption. Moreover, the checkpoint granularity can be adjusted dynamically depending on rollback frequency.

Our tactic to obtain such accurately predicted inputs and outputs quickly is to use the model at a higher abstraction level. For GL timing simulation considered in this work RTL model can be used as an accurate and fast predictor. In other words, in each local simulation a partitioned GL module is executed together with the entire RTL design model, i.e. the original DUV at RTL and with the testbench, as a predictor. The purpose of executing the entire RTL in each local simulation is to generate the predicted inputs and outputs accurately and quickly. Those predicted inputs and outputs are then used in the prediction phase.

Conceptual view of the proposed parallel simulation using highly accurate prediction is shown in Fig. 1. Fig. 1(a) shows the original GL design to be simulated. Fig. 1(b) shows the corresponding RTL model, which is used for generating predicted inputs and outputs for each local simulation, as an accurate and fast predictor. It is known that an RTL simulation is at least 100 times faster than GL model (see the preliminary experimentation in Section IV). As shown in Fig. 1(c), the original GL design is partitioned into six modules, each to run


GL Block
RTL Block
DUT
Monitors and Drivers
Testbench
Local simulation
B1
B2
B3
B4
cpu1
cpu2
cpu3
cpu4
core5
core6
(a) An original GL design to be simulated
(b) An RTL design, which is used for generating predicted inputs and outputs in local simulation
(c) Proposed parallel simulation using highly accurate prediction
Figure 1. Basic concept of the proposed parallel event-driven HDL simulation using highly accurate prediction.

as local simulation on a local HDL simulator. Each local simulation is depicted to use an RTL model as the accurate and fast predictor at the top, and a partitioned GL block at the bottom left of each figure. Each local simulation, executed on a separate CPU or a processor core, is performed with each partitioned GL module and with the (fast) RTL model for the entire design. The RTL model for the entire design in each local simulation generates the required highly accurate predicted inputs and outputs, needed for this method.

Usually, GL netlist is the outcome of logic synthesis that starts with RTL model of the design. Therefore, we can assume that GL and RTL designs are 100% cycle-by-cycle equivalent for designs with a single clock, and almost 100% cycle-by-cycle equivalent for designs with multiple asynchronous clocks (the reason for occasional discrepancy will be explained shortly).

However, for our approach to work, the predicted inputs and outputs must have accurate delays assigned to the combinational logic so that every local simulation is executed in the prediction phase as often as possible. This means that in order to minimize prediction failures, we must annotate the exact delay values to the predicted inputs and outputs obtained from the RTL model. To accomplish this, we will explore a simple but efficient partitioning strategy, explained below.

A better partitioning scheme satisfying the above requirement is to partition the GL design into multiple modules at the flipflop boundary (Fig. 2). This way, the size of each partition can be increased or decreased easily, depending on the number of CPU or processor cores available. Also, the predicted inputs and outputs obtained from RTL DUV can be made accurate by annotating simple yet accurate delay values. This is because there are no multiple combinational paths between the partitions that would make the prediction of exact delay values difficult; there is just a single connecting path from a flipflop in one partition to other partition(s). Those annotated simple delay values are actually clock-to-Q delays of the corresponding flipflops, which are easily obtained from the timing characteristics of the flipflops. This helps to significantly increase the prediction accuracy.

The detailed architecture for local simulation in our parallel event-driven GL HDL simulation is shown in Fig. 3. From the computer architecture point of view, our strategy is to significantly off-load the heavy I/O processing, necessary for communication and synchronization activities in conventional distributed parallel simulation by only slightly increasing the internal computation at each processor.

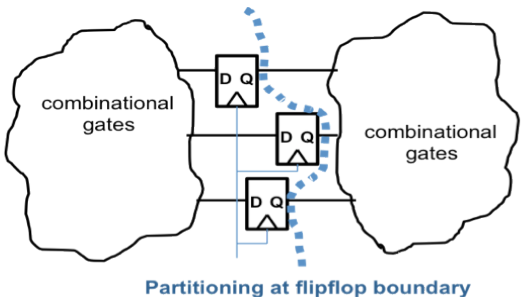


Figure 2. Partitioning scheme for accurate prediction.

Our approach provides an important additional benefit: RTL model naturally plays a role of the reference model. Therefore, our method can automatically determine whether the design to be simulated is consistent with the model at the higher level of abstraction. Possible simulation mismatches between the RTL and GL models (caused by functional difference and/or timing violation) can be automatically detected and reported to the designer or verification engineer for a possible investigation and debugging.

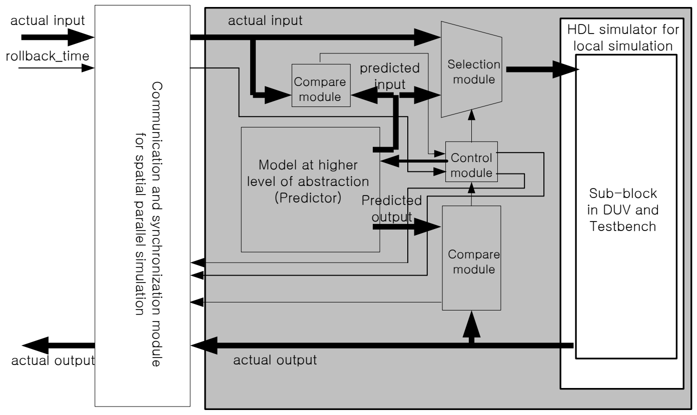

actual input
rollback_time
actual output
Communication and synchronization module for spatial parallel simulation
actual input
Compare module
predicted input
Selection module
HDL simulator for local simulation
Model at higher level of abstraction (Predictor)
Control module
Predicted output
Compare module
Sub-block in DUV and Testbench
actual output

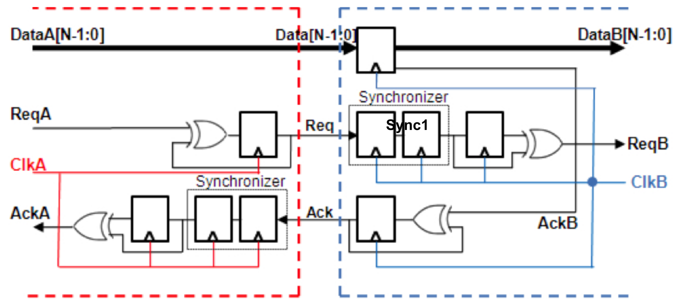
Figure 3. Detailed architecture for local simulation.

IV. Preliminary Experimental Results

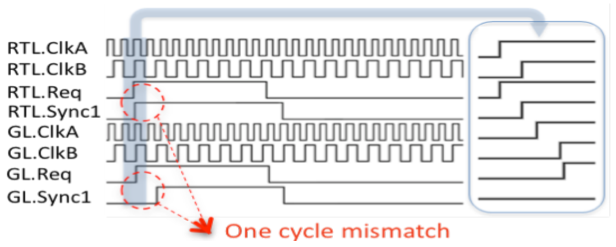
To provide a preliminary proof of concept for the proposed parallel event-driven GL HDL simulation approach using highly accurate prediction, we performed several experiments in gate-level timing simulation and RTL simulation.

For designs having a single clock, the RTL simulation and its corresponding gate-level timing simulation must be 100% cycle accurate at the flipflops boundary. Therefore, in parallel GL timing simulation, our prediction can achieve 100% accuracy without much difficulty. But, as mentioned earlier, one of the challenging questions is whether 100% accurate prediction could be achieved for designs with multiple asynchronous clocks. Normally, in the presence of such multiple clocks the RTL simulation and its corresponding gate-level timing simulation may not be 100% cycle accurate at the flipflops boundary. The discrepancy between RTL simulation and GL timing simulation can be observed on some of the CDC (Clock Domain Crossing) signals when the sending clock and a receiving clock exhibit a specific phase relation. We would like to emphasize that for the designs having multiple asynchronous clocks, gate-level timing simulation is still correct even though it may not be 100% cycle equivalent with respect to its corresponding RTL simulation. Therefore, our approach should produce the same exact result as the traditional GL timing simulation.

We have conducted an experiment with a real design (handshaking logic) commonly used for data transfer between two asynchronous clocks domain. The design, shown in Fig. 4(a), was synthesized using Xilinx Foundation tool and simulated using Cadence NC-Verilog.



(a) Two-Phase Handshaking Logic



(b) Cycle discrepancy between RTL and GL timing simulation

Figure 4. Two-Phase Handshaking Logic.

We observed that the output values of the first flipflop in the synchronizer for REQ signal of the gate-level timing simulation differs only 69 times from that of RTL functional simulation during the entire simulation of 9,460 cycles. This could already provide evidence that RTL design is accurate enough for being an excellent predictor in our parallel event-driven GL timing simulation approach even for designs with multiple asynchronous clocks. One of such situations is depicted in Fig. 4(b). In this design, all cycle discrepancies are observed strictly during a single cycle period only; they are not propagated to the next simulation time.

The performance of the proposed parallel event-driven GL timing simulation using highly accurate inter-module signal prediction can be extrapolated from the data in Table 1. As shown in the table, the RTL simulation of AES and JPEG encoder designs [13] is significantly faster than the SDF delay-annotated GL simulation. Also, as both designs have a single clock, the same cycle accuracy for all flipflops in the design has been observed at both the RTL simulation and GL timing simulation.

Based on these experiments, we are confident that the RTL model is an excellent predictor for our spatial parallel event-driven GL timing simulation. It is at least two orders of magnitude faster, and almost 100% accurate even for designs with multiple asynchronous clocks.

Design	*RTL simulation (sec)*	*GL timing simulation (sec)*	*Ratio*
AES	110	18669	× 167
JPEG encoder	184	47192	× 256

Table 1. RTL and GL conventional timing simulation times

V. Conclusions and Future Work

We propose a novel, radically different solution to greatly reduce or completely eliminate communication and synchronization overhead in a distributed event-driven GL simulation. This is achieved by performing simulation with highly accurate inter-module signal prediction that comes from RTL model. A simple but powerful flipflop-bounded partitioning scheme is introduced to obtain accurate signal prediction considering delay in GL simulation. Since our approach naturally includes comparison with a reference model, we believe that the proposed simulation method provides a "correct by simulation" methodology, which explicitly checks the consistency between RTL and GL models, once the RTL model has met the specification.

In principle, the proposed approach is also applicable to distributed RTL simulation if a higher abstraction model, i.e., Transactional Level (TL) model, is used as an accurate predictor. Our final goal is to increase the performance RTL simulation by our distributed HDL simulation with accurate prediction, which is our future research topic.

Acknowledgment

This work was supported in part by the US National Science Foundation, award no. CCF 0702506 and CCF 1017530.

References

[1] *SimCluster* datasheet, Avery Design Automation (http://www.avery-design.com)

[2] *MP-Sim* datasheet, Axiom Design Automation (http://www.axiom-da.com)

[3] R. Bagrodia, Y. Chen, V. Jha and N. Sonpar. "Parallel Gate-level Circuit Simulation on Shared Memory Architectures." In Computer Aided Design of High Performance Network Wireless Networked Systems,, NSF pp. 170-174, 1995.

[4] H. Bauer, C. Sporrer, and T. Krodel. "On Distributed Logic Simulation using Time Warp." in Proc. VLSI International Conference (IFIP), Edinburgh, 1991.

[5] D. Chatterjee, A. DeOrio and V. Bertacco, "Event-Driven Gate-Level Simulation with GP-GPUs." in Proc. ACM/IEEE Design Automation Conference (DAC'09), San Francisco, CA, pp. 557-562, July 2009.

[6] R.D. Chamberlain, "Parallel Logic Simulation of VLSI Systems.", in Proc. 32nd ACM/IEEE Conference on Design Automation, pp 139-143, 1995.

[7] R.M. Fujimoto, "Time Warp on a Shared Memory Multiprocessor", Transactions of the Society for Computer Simulation, Vol, 6, No. 3, pp 211-239, July 1989

[8] R.M. Fujimoto, "Parallel Discrete Event Simulation." Communication of the ACM, Vol. 33, No. 10, pp 30-53, Oct. 1990.

[9] A. Gafni. "Rollback Mechanisms for Optimistic Distributed Simulation Systems." in Proc. the SCS Multiconference on Distributed Simulation, vol.3, pp 61-67, July 1988

[10] L. Li, H. Huang and C. Tropper, "DVS: An Object-Oriented Framework for Distributed Verilog Simulation", in Proc. of the 17th Workshop on Parallel and Distributed Simulation (PADS'03), 2003.

[11] D. Lungeanu and C.J.R. Shi. "Parallel and Distributed VHDL Simulation", in Proc. Design, Automation and Test in Europe (DATE'00), pp 658–662, March 2000.

[12] D.M. Nicol, "Principles of Conservative Parallel Simulation." in Proc. of the 28th Winter Simulation Conference, pp. 128–135, 1996.

[13] http://www.opencores.org

[14] L. Zhu, G. Chen, B.K. Szymanski, C. Tropper, Tong Zhang "Parallel Logic Simulation of Million-Gate VLSI Circuits" in Proc. 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems - MASCOTS '05, 2005.

[15] D. Kim, M. Ciesielski, K. Shim and S. Yang, " Temporal parallel gate-level timing simulation", in *Proc. High Level Design Validation and Test Workshop (HLDVT)*, pp. 111–116, 2008