A New Distributed Parallel Event-driven Timing Simulation for ECO Design Changes

Seiyang Yang, Doohwan Kwak, Jaehoon Han,
Dept. of Computer Eng.,
Pusan National University
Busan, Korea
e-mail: syyang@pusan.ac.kr

Namdo Kim,
Infra Design Technology,
Samsung Electronics Co.,
Kyunggi-do, Korea
e-mail: nd.kim@samsung.com

***Abstract* - Prediction-based distributed parallel event-driven hardware description language simulation on multi-core computing platforms is a new promising approach to boost simulation performance. Traditional distributed parallel event-driven simulation methods suffer heavy synchronization and communication overhead for transferring the signal data among local simulators, which could easily nullify most of the expected simulation speed-up from parallelization. In our approach, the signal data to be transferred is predicted first in each local simulation independently. No synchronization and communication incurs when the prediction succeeds, and the actual signal data transfer with synchronization and communication among the local simulators occurs only when the prediction fails. Therefore, as far as prediction accuracy remains high, the high simulation speed-up from the parallelization can be anticipated from the approach. In this paper, we have proposed the prediction-based distributed parallel event-driven timing simulation for a series of design changes in typical ECO flow. We also have performed the preliminary experimentation in actual design changes, obtained high prediction accuracy with real designs from industry and achieved significant speed-up gain from the proposed parallelization.**

Keywords—distributed parallel simulation; synchronization; communication; partitioning; simulator; simulation; verification; EDA

I. INTRODUCTION

Simulation has still remained the most popular verification method in chip designs because of ease of use, low cost, 100% signal controllability and observability, etc. Specifically, event-driven Hardware Description Language (HDL) simulation is the most common technique used for functional and timing simulations [1]. However, event-driven simulation suffers from very low performance for complex design objects because of its inherently sequential nature. In chip designs, this has gotten much worse in gate-level simulation than Register Transfer Level (RTL) simulation because the number of simulation objects to be dealt with is much larger at gate-level than at RTL. But, the use of gate-level functional or timing simulation is still quite active and even increasing nowadays for many important reasons [16][17]. Some of them include verification requirement for designs having many asynchronous clocks, limitation of static functional and timing verification methods such as equivalence checking and static timing analysis, variability of deep sub-micron processing technology, etc. Therefore, event-driven HDL simulation is heavily used for both functional and timing verification. Usually, once the bug is found and fixed after a simulation run, another new simulation run is required with a new HDL code or netlist to ensure that the bug is correctly removed and no new bug is accidently brought. This process is iterated until the designers or verification engineers believe no more bugs exist in the design. In this sense, simulation in the design flow is a highly repeated process before and after a series of continuous design changes.

Distributed parallel event-driven HDL simulation has been proposed to alleviate the low performance of sequential simulation [2][3][4]. Unfortunately, it has been not successful because of: i) difficulty in design partitioning; ii) heavy synchronization and communication overhead among modules imposed by the distributed environment, especially in gate-level timing simulation; and iii) load balancing among the distributed simulation jobs.

This paper consists of following; first we briefly mention the previous work and motivation in Section II, and explain our unique and noble approach to distributed parallel simulation in Section III. In next main section, we propose the prediction-based distributed parallel event-driven timing simulation for a series of design changes in the typical ECO (Engineering Change Order) flow, and claim that the performance of gate-level timing simulation could be greatly improved from the proposed approach. In Section V, we have performed some preliminary experiments with real SOC (System On Chip) designs from industry for demonstrating the expected benefit, followed by the conclusion and future work in Section VI.

II. PREVIOUS WORK AND MOTIVATION

The area of distributed parallel simulation is rich in literature. All known works concern traditional distributed parallel simulation, which is based on physical partitioning of the design into multiple sub-designs, assigned to individual local simulators. This simulation concept has been known since late 1980s as Parallel Discrete Event Simulation (PDES) [9]. The main issues in PDES are partitioning, synchronization and granularity. There are basically two types of

 ISBN: 978-1-61208-308-7

synchronization methods in distributed parallel simulation: *conservative* (lockstep based) and *optimistic* (rollback based). These two types differ in the way modules of the partitioned design communicate during simulation for synchronization. Their performance varies with the design and partition strategy, but usually the optimistic method is faster [9][10][11][12]. Most recently, Chatterjee [13] and Zhu [14] proposed the distributed parallel event-driven gate level simulation using general purpose GPUs (Graphic Processing Units). However, it could only handle gate level zero-delay simulation. It is known that it is not effective for the gate level timing simulation [15]. In conclusion, these methods are not practical, do not scale, and have performances depending heavily on optimal partitioning [18].

Recently, some Electronic Design Automation (EDA) vendors have introduced parallel event-driven HDL simulators for multi-cores [5][6][7][8]. Parallel HDL simulation with multi-core technology looks more promising than the original distributed parallel HDL simulation. In multi-core parallel simulation, (inter-module) communication can be accomplished by a straightforward fast memory read/write. However, the expected speed-up was observed only for a special class of designs, such as BIST (Built in Self Test) logic just for gate-level zero-delay simulation, and the speed-up curve quickly saturates with the number of available cores. The problem becomes particularly difficult for large number of cores, which quickly increases the global communication and synchronization overhead among partitioned sub-designs. It mainly comes from the fact that the difficulty of partitioning for distributed parallel simulation lies in simultaneously considering the reduction of the synchronization and communication overhead and load balancing among distributed simulation jobs. Due to the huge design size, much heavier synchronization and communication overhead, etc., the problem even gets much worse for gate-level timing simulation, where the stronger demand for high simulation performance exists, but its speed is even slower than that of gate-level zero-delay simulation at least by one order of magnitude.

In summary, the success of traditional distributed parallel event-driven simulation on multi-core strongly depends on such "ideal" partitioning, which itself is a known intractable problem and therefore is impossible to apply to complex industrial large designs [18]. This is the main reason that multi-core parallel event-driven HDL simulators are not popular in the design community these days although there has been a great demand for increasing gate-level simulation performance on multi-core platforms.

III. PREDICTION-BASED DISTRIBUTED PARALLEL EVENT-DRIVEN HDL SIMULATION

Yang [3][4] had proposed a new promising approach to boost up the simulation performance, prediction-based distributed parallel event-driven HDL simulation on multi-core computing platforms. This approach is based on *predicting* input and output stimulus that need to be applied to module(s) in each local simulation (We will call each of individual simulation in distributed parallel simulation local simulation). How to accurately predict input and output values is explained in the next section.

The predicted input values are stored in local memory and applied to the input ports of a local module assigned to a given simulator. Then, the actual output values at the output ports of that module are compared on-the-fly with the predicted output values, also stored in a local memory. Figure 1 shows an example design consisting of two modules dependent on each other inputs. Simulating the modules in parallel requires predicting inputs for each of the two sub-modules.

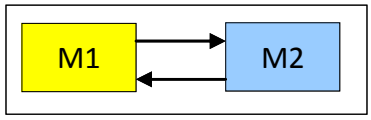

Figure 1: An example design with two dependent sub-modules

Figure 2 shows two sub-modules being simulated in parallel on two cores. Each sub-module uses predicted inputs by default, while its actual outputs are compared against the predicted outputs (stored earlier in local memory). A multiplexer at each sub-module selects between the predicted inputs and the actual inputs. Note that, when both sub-modules access their actual inputs from the other sub-module, synchronization and communication overhead incurs.

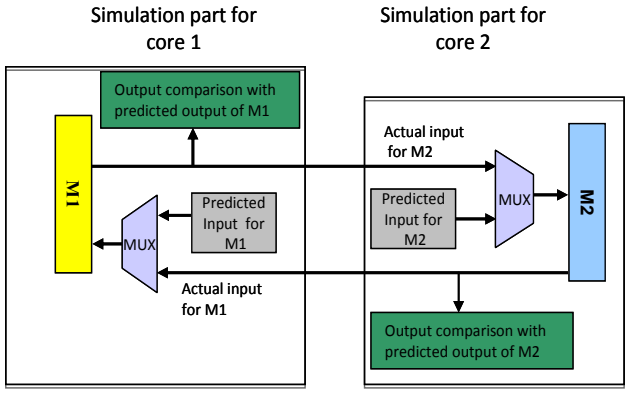

Figure 2: Conceptual diagram of prediction-based distributed parallel event-driven HDL simulation

As long as the prediction is correct, communication and synchronization between two local simulations is completely eliminated. We call this phase of simulation the prediction phase. Only when the prediction fails, the actual input values, coming from the other local simulation, are used in simulation; we call this phase of simulation the actual phase. When prediction fails, each local simulation must roll back to the nearest checkpoint and is restarted from that point. This is possible by generating *checkpoint,* i.e., saving simulation state

 ISBN: 978-1-61208-308-7

or design state, during the simulation in the prediction phase. Note that, when parallel simulation enters the actual phase, it should try to return to the prediction phase as early as possible to attain the maximum speed-up. This is done by continuously comparing the actual outputs of all local simulations with their predicted outputs and counting the number of matches on-the-fly. If the number of matches exceeds a predetermined value, the simulation is switched back to the prediction phase. Therefore, it is obvious that prediction accuracy is the most critical factor in this approach. Prediction accuracy near 100% will give almost linear speed-up even when the number of processor cores increases.

IV. PREDICTION-BASED DISTRIBUTED PARALLEL EVENT-DRIVEN TIMING SIMULATION FOR ECO DESIGN CHANGES

As being mentioned in the previous section, having an accurate prediction data is imperative in the prediction-based distributed parallel event-driven HDL simulation. In [3], Yang had proposed one way to obtain the accurate prediction data, i.e., from the earlier simulation with the higher abstraction model. For example, if the prediction-based distributed parallel simulation is for gate-level simulation with a gate-level netlist, the prediction data could be gathered from the earlier RTL functional simulation with the higher abstraction model, i.e., Verilog RTL design, from which the netlist is going to be synthesized.

In this paper, we propose another way to obtain the accurate prediction data in the design flow. When we apply this prediction-based distributed parallel timing simulation at gate level for verifying design changes in ECO flow, our idea is to get it from signal dump from the earlier simulation before design change. For example, for doing the distributed parallel simulation with the example design in Figure 1 on two cores, all input and output ports of two sub-modules M1 and M2 are registered for signal dump. While simulating the design before design change, signal dumping is performed for those input and output ports of sub-module M1 and M2, and saved as the prediction data for later simulation after design change. Let's assume that a bug in sub-module M1 is revealed from the 1st simulation, and its HDL source or netlist is modified for fixing it. Then, prediction-based distributed parallel timing simulation deployed for the 2nd simulation after design change will use this saved signal dump on the input and output ports of sub-modules M1 and M2 as the prediction data.

As mentioned earlier, simulation is the highly iterated activities before and after a series of continuous design changes in the entire design phase. The application of distributed parallel HDL timing simulation to verify design changes in ECO flow could result in a profound benefit for reducing the total simulation time of all simulation runs, due to this repeating nature. Again, the key factor for boosting the simulation performance from the prediction-based distributed parallel simulation is the prediction accuracy. Intuitively, higher prediction accuracy for the prediction-based distributed parallel simulation in ECO flow is expected from less design change. For example, very high prediction accuracy could be expected from timing-only variant design changes keeping same functionality. But, there are other factors to consider. First, although there is a design change inside a module, it may only internally affect the module, but not at its output port and beyond. In other words, the effect of the change could not be propagated to any of its output port. If this is the case, the prediction accuracy of signal dump from the simulation before design change is 100%. This is the best scenario for prediction-based distributed parallel simulation. Second, a design change inside a module could affect any of its output port, but not its input port. This is the case when there is no feedback connection from its output to its input, possibly through some other module(s), gates(s), etc. For this case, the prediction accuracy of input signal dump from the simulation before a design change is still 100%. But that of output signal dump is not 100%. Third, a design change inside a module could affect both its output and input port, when there is some feedback connection from its output to its input. For this, the prediction accuracy of signal dump from the simulation before design change is not 100%. This might be the worst scenario, and its prediction accuracy cannot be 100%. But, if its accuracy is still high, we could achieve the substantial speed-up from the distributed parallel simulation.

Most design change(s) in ECO flow at the back-end design stage seldom changes the functional behavior of the corresponding sub-module globally in the entire simulation scenario. In fact, many ECO design changes are adjustments only for timing, e.g. buffer substitution, insertion, or removal, functionally equivalent but timing different cell substitution, cell re-placement, re-routing, etc. All of these ECO design change examples do not alter the functional behavior at all. Therefore, we intuitively know that the prediction accuracy for prediction-based distributed parallel event-driven timing simulation could be quite high in typical ECO flow.

Besides reducing communication and synchronization overhead, another important factor to consider in distributed parallel simulation is load balancing. In fact, significant speed-up from distributed parallel simulation is only possible by satisfying these two conditions simultaneously. Another benefit of applying the prediction-based distributed parallel event-driven timing simulation for ECO design changes is that a good load balancing could be easily known from a simulation run before design change. That is, while the prediction data is being collected during the simulation run before design change, the simulation load profiling for major design sub-blocks in design is also carried out. In this case, the prediction data being collected is the signal dump of all input and output ports of those major design sub-blocks.

As any ECO design change, especially timing variant but function invariant design change seldom or never alter the simulation behavior drastically. The simulation load profiles

 ISBN: 978-1-61208-308-7

before and after design change should be pretty much the same. Therefore, as the simulation load of a new simulation run after design change could also be accurately predicted from that of the simulation run before design change, a good load balancing as well as the low communication and synchronization overhead could be anticipated from the prediction-based distributed parallel event-driven timing simulation in ECO flow.

In series of successive ECO design changes, all remaining simulation runs except the 1st simulation run can utilize the signal dump of all input and output ports of the major design sub-blocks as the accurate prediction data for prediction-based distributed parallel simulation. To execute the 1st gate-level timing simulation run also in parallel fashion, the prediction data could be brought from the gate-level functional simulation, i.e. gate-level zero-delay simulation. Therefore, the entire timing simulation runs in ECO flow could be run in parallel, and we could expect that the total simulation time for entire ECO design changes is greatly reduced.

In the next section, we have performed some preliminary experiment to justify our claims with real designs from industry.

V. PRELIMINARY EXPERIMENTATION

In this section, we provide some interesting preliminary experimental results for measuring the prediction accuracy and estimated speed-up from the proposed approach in ECO design change with real industrial designs. The ECO design changes are confined to timing variant but functional invariant. The Verilog simulator we used for the experiment is one of leading commercial Verilog simulators. The first test design in Table I is a BIST design. Here, a design change is a cell replacement for adjusting the timing while keeping the original functionality.

For parallel simulation, the design is partitioned into 5 pieces according to the simulation activities. To do this, the profiling feature in the commercial Verilog simulator had been used.

First, we have measured the prediction accuracy from the ECO design change. The measuring procedure is the following; i) while running the simulation before the corresponding ECO design change with an original design, the prediction data is collected by dumping the signal values on all input and output ports of all modules at the partition boundary, ii) the ECO design change is performed, iii) while running the simulation after the corresponding ECO design change with a modified design, the actual data is collected by dumping the signal values on all input and output ports of same modules at the partition boundary, and iv) comparing the prediction data with the actual data. Note that this comparison should be event-by-event basis. The resulting prediction accuracy in this case is 99.9%. This means that during 99.9 % of the total simulation time each local simulation can be run independently without incurring any communication and synchronization. The communication and synchronization among five local simulations is required only for 0.1 % of the total simulation time. Other additional factors to be considered are checkpoint overhead, and rollback and restart overhead. In this design, only a single rollback is needed. By considering all these, the expected speed-up from the proposed method in the specific ECO design change is 5.12. It is pretty surprising and almost too good to believe.

TABLE I. EXPERIMENTAL RESULT 1

Design Name	Prediction Accuracy (%)	# of Partitions	Expected Speed-up from the proposed method
BIST	99.9	5	5.12

At first glance, it seems this 5.12x speed-up is by no means possible from any distributed parallel simulation with 5 partitions. This is true for any traditional distributed parallel simulation methods that always require communication and synchronization during the entire simulation time. Then, how about for the proposed prediction-based distributed parallel simulation method? The answer is it is possible. The reason is the following. When each local simulation is run independently in the prediction mode, it is possible that even the speed of the slowest local simulation is greater than *(the speed of the non-parallel original simulation)/(# of partitions)*. In this design, the non-parallel original (single core) gate-level timing simulation takes 10,916 sec (wall clock time). In the proposed prediction-based distributed parallel simulation, the slowest local simulation running in the prediction mode only takes 2,130 sec., which is shorter than a fifth of 10,916 sec. We think this comes from the fact that the smaller design avoids virtual memory trashing, which leads to low CPU utilization and degrades the system performance. Note that the unavoidable heavy communication and synchronization nullify this potential benefit in the traditional distributed parallel simulation. We believe that this is a very interesting and important finding that largely differentiates the proposed approach from others.

TABLE II. EXPERIMENTAL RESULT 2

Design Name	Prediction Accuracy (%)	# of Partitions	Expected Speed-up from the proposed method
Mobile AP	99.6	8	Not Available

The second test design is a state of art mobile Application Processor (AP) design from the industry. Again, a design change is a cell replacement for adjusting the timing while keeping the original functionality. For parallel simulation, the design is partitioned into 8 pieces according to the simulation activities. However, due to the security reason, unfortunately we have only measured the prediction accuracy from the ECO design change for the second design. It is shown in Table II.

 ISBN: 978-1-61208-308-7

Like the first design, the very high prediction accuracy has been observed for the second design, too.

From the experiment with two real designs from the industry, we strongly believe that our prediction-based distributed parallel HDL simulation is very effective for boosting the simulation performance at least for timing-only variant ECO design changes.

VI. CONCLUSION AND FUTURE WORK

HDL simulation is a very iterated process before and after a series of design changes. Prediction-based distributed parallel HDL simulation is a new promising approach to parallelize the simulation. Its effectiveness heavily relies on the prediction accuracy. As near 100% accuracy can eliminate most of synchronization and communication overhead, the speed of parallel simulation could significantly be increased. In this paper, we have applied the prediction-based distributed parallel event-driven HDL timing simulation for ECO design changes in chip designs.

We have experimentally shown that in the timing-only variant design changes the accurate prediction data for the distributed parallel simulation after design change could be obtained from the earlier simulation before design change, and contribute to the large decrease of communication and synchronization overhead. Therefore, almost linear speed-up from the parallelization could be anticipated. In the future, we would like to extend the application scope of this approach to the function variant design changes as well as timing-only variant design changes. Our final goal is to implement the prediction-based distributed parallel event-driven HDL simulation method on commercial Verilog simulators, and it is under investigation.

REFERENCES

[1] T. Anderson and R. Bhagat, “Tackling Functional Verification for Virtual Components,” ISD Magazine, November 2000, pp. 26.

[2] P. Rashinkar, P. Paterson, and L. Singh, System-on-a-Chip Verification: methodology and techniques, Springer, 2002.

[3] D. Kim, M. Ciesielski, and S. Yang, “A new distributed event-driven gate-level HDL simulation by accurate prediction,” Design and Test Europe (DATE 2011), March 2011, pp. 547-550.

[4] T. B. Ahmad, N. Kim, B. Min, A. Kalia, M. Ciesielski, and S. Yang, "Scalable Parallel Event-driven HDL Simulation for Multi-Cores," Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design, (SMACD 2012), Sept. 2012, pp. 217-220.

[5] IUS Multicore datasheet, Cadence Design Systems (http://www.cadence.com).

[6] VCS Multicore datasheet, Synopsys (http://www.synopsys.com) .

[7] SimCluster datasheet, Avery Design Automation (http://www.averydesign.com)

[8] MP-Sim datasheet, Axiom Design Automation (http://www.axiomda.com)

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

[10] A. Gafni, “Rollback Mechanisms for Optimistic Distributed Simulation Systems,” SCS Multiconference on Distributed Simulation, Vol. 3, July 1988, pp. 61-67.

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

[12] D.M. Nicol, “Principles of Conservative Parallel Simulation,” Proceedings. of the 28th Winter Simulation Conference, 1996, pp. 128–135.

[13] D. Chatterjee, A. DeOrio, and V. Bertacco, “Event- driven gate-level simulation with general purpose GPUs,” Proceedings. of Design Automation Conference (DAC09), June 2009, pp. 557-562.

[14] Y. Zhu, B. Wang, and Y. Deng, “Massively Parallel Logic Simulation with GPUs,” Article No. 29, ACM Trans. On Design Automation of Electronic Systems, June 2011.

[15] K. Chang and C. Browy, “Parallel Logic Simulation: Myth or Reality?” Computer, Vol. 45, No. 4, April 2012, pp. 67-73.

[16] Gate-level Simulation Methodology, Whitepaper, Cadence Design Systems (www.cadence.com), 2013

[17] VerificationBlog (http://whatisverification.blogspot.com/2011/06/gate-level-simulations-necessary-evil.html)

[18] L. Li and C. Tropper, “A design-driven partitioning algorithm for distributed Verilog simulation,” in Proc. 20th International Workshop on Principles of Advanced and Distributed Simulation (PADS), June 2007, pp. 211–218.

 ISBN: 978-1-61208-308-7