Proceedings of the 3rd World Congress on Electrical Engineering and Computer Systems and Science (EECSS'17)
Rome, Italy – June 4 – 6, 2017
Paper No. EEE 103
ISSN: 2369-811X
DOI: 10.11159/eee17.103

Predictive Parallel Gate-level Timing Simulation with Fast and Accurate Prediction

Jaehoon Han[1], Seiyang Yang[2]
[1]Hanwha Thales
Changwon, Korea
mauser87@nate.com
[2]Pusan National University
Busan, Korea
syyang@pusan.ac.kr

Abstract **-** A promising predictive parallel event-driven logic simulation had been proposed for greatly enhancing the parallel simulation performance, which could be heavily disturbed by synchronization & communication overhead among local simulations. For achieving the maximal simulation performance in the predictive parallel event-driven logic simulation, the high prediction accuracy is the key factor. In this paper, we have enhanced this noble parallel simulation method with fast, yet accurate prediction using incremental simulation. Preliminary experimentation has been performed for common design changes, and shown the effectiveness of the proposed approach.

Keywords**:** parallel event-driven logic simulation; simulation; digital hardware verification; SOC; VLSI

1. Introduction

Event-driven logic simulation suffers from very low performance for complex digital logic designs because of its inherently sequential nature. Especially, this has gotten worse in gate-level functional 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, and worst in gate-level timing simulation. But, the use of gate-level timing simulation is still quite active and even increasing nowadays for many important reasons [1]. Parallel event-driven logic simulation has been proposed to alleviate the low performance of sequential simulation [2][3][4][5][6]. Unfortunately, it had been not successful because of; i) difficulty in design partitioning, ii) heavy synchronization and communication overhead among partitioned modules, especially in gate-level timing simulation, and iii) load balancing among the distributed simulation jobs [7]. A very different distributed parallel event-driven simulation method based on the prediction had been proposed [8]. Afterward, the successive papers published had shown the effectiveness of the method [9][10][11]. In this noble method, the high prediction accuracy could solely achieve the significant simulation performance improvement. In this paper, we have enhanced this predictive parallel event-driven logic simulation method for a series of gate-level timing simulation through much faster, but still very accurate prediction using incremental simulation. Preliminary experimentation has been performed for common design changes, and has shown the effectiveness of the approach.

2. Predictive Parallel Event-driven Logic Simulation

First, the fundamental of predictive parallel event-driven logic simulation will be explained [9][10][11]. We will call each of the simulations for a partitioned module in parallel event-driven simulation as *local simulation*, and the corresponding partitioned module as *local module*. Unlike applying the actual input vales in traditional parallel event-driven simulation, predictive parallel event-driven logic simulation method applies the *predicted* input values to the input ports of a partitioned 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. Note that, when a partitioned module uses its actual inputs supplied from other partitioned modules, synchronization and communication overhead incurs.

In this method, however, as long as the prediction is correct, communication and synchronization among local simulations is completely eliminated. We call this phase of parallel 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 parallel 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 or design state, during the simulation in the prediction phase. Note also that, when this predictive parallel simulation enters the actual phase, it should return back to the prediction phase as early as possible to attain the maximal 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.

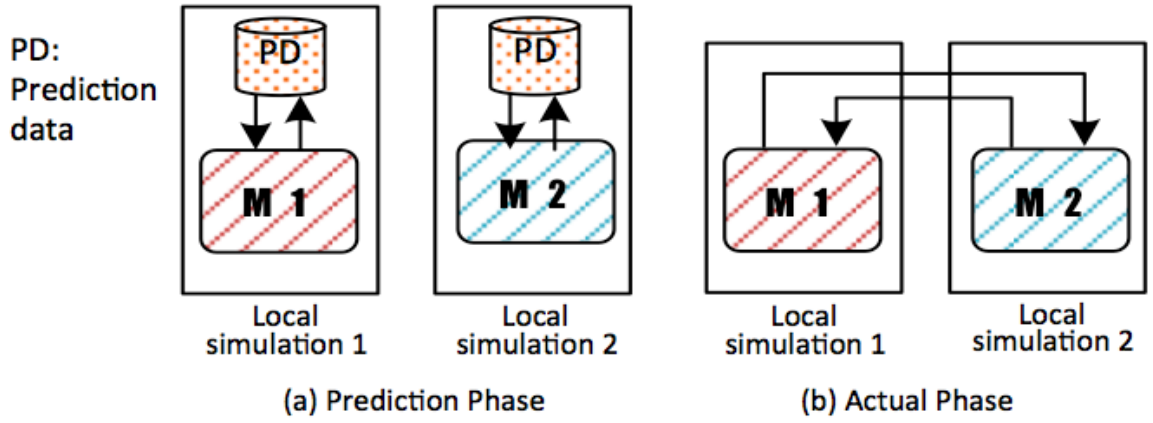

Fig. 1: Two Execution Phases of Predictive Parallel Event-driven Logic Simulation.

3. Predictive Parallel Gate-level Timing Simulation Through Fast and Accurate Prediction

From Chapter 2, it is obvious that the high prediction accuracy is the key factor for achieving the maximal simulation performance in the predictive parallel event-driven logic simulation. For example, as 100% prediction accuracy contributes to zero synchronization and communication overhead among local simulations, even the linear or sub-linear speed-up is possible under the optimal partition.

During the gate-level timing simulation phase, there are three cases to obtain the prediction data, i.e. the values of all inputs and outputs of every local module at the events. The first, although trivial, the 100% accurate prediction data is always obtained from the earlier gate-level timing simulation for repeating predictive parallel timing simulation if there is no design change between these two successive simulations (Case I). The second, the highly accurate prediction data could be obtained from the final gate-level functional simulation or RTL simulation for the first predictive parallel gate-level timing simulation (Case II). This is because all of these simulations are functionally identical. However, the third, the prediction data from the earlier gate-level timing simulation might not be accurate enough for the current predictive parallel timing simulation if there is at least one design change because it could easily alter the simulation behavior (Case III). Each of this design change is either functional-oriented or timing-oriented. If the design change alters its functional behavior of the design (and very possibly its timing behavior too), it is called functional change. By contrast, the timing change alters its timing behavior only.

For all of these three cases, the prediction data of the current predictive parallel gate-level timing simulation is obtained from the previous simulation, which could be RTL or gate-level functional simulation, or gate-level timing simulation (Note that any previous simulation itself also could done as predictive parallel fashion). Therefore, next question to be answered is how we could avoid or enhance the prediction accuracy when that of the prediction data taken from the previous simulation is not sufficiently accurate. The possible inaccuracy could be classified as either timing-oriented or functional-oriented. To overcome these two types of prediction inaccuracies, we deploy two different tactics. One is avoidance, and the other enhancement.

3.1. Avoidance of Timing Inaccuracy

The timing-only inaccuracy in prediction data could be avoided by the constrained partition if we only allow the legal partition such that the every output of local modules must be the output of flipflops. In such a partition, the exact timing behavior of prediction data could be easily augmented to its correct functional behavior by the simple and fast static timing analysis method because the exact delay calculation involves only the propagation delay of flipflops and the single path wire delay. We also claim that this constraint for partition is easily achievable because this is one of the normal design constraints in sub-module designs for many reasons, e,g. fast and easy complete static timing analysis, on these days.

3.2. Enhancement of Functional Accuracy

Functional inaccuracy mostly comes from the functional design change. Especially, the functional design changes made during the gate-level timing simulation phase are usually for removing hidden hard functional bugs unfound unfortunately at the earlier design verification phase, e.g. RTL simulation or gate-level functional simulation. Therefore, if there is any functional design change between two successive simulations, it is very likely that the accuracy of prediction data obtained from the previous simulation is not high enough for the current predictive parallel gate-level timing simulation due to the functional discrepancy. In such a situation, we need to obtain the new prediction data having higher accuracy through the regeneration. Obviously, there must be a tradeoff between the accuracy of the new prediction data and the regeneration time (Note that the total simulation time in the predictive parallel simulation should include the regeneration time).

One extreme, although not making sense at all, is to generate 100% accurate prediction data by running the exactly same one as the current predictive parallel gate-level timing simulation. For fast regeneration of the new accurate prediction data, much more clever way is to run the simulation at the higher abstraction level, i.e. gate-level functional simulation or RTL simulation [11]. Because as both of these higher level simulations should have same functional behavior as the current predictive parallel gate-level timing simulation, we can expect very high functional accuracy of the new prediction data. This is we call the enhancement of prediction accuracy.

However, the regeneration time is strictly bounded by the time for simulation at the higher abstraction level, and so is the total simulation time of the predictive parallel gate-level timing simulation. To speed up the regeneration process more, we propose the incremental simulation, explained in the next section.

3.3. Incremental Simulation

Remember that there is at least one functional design change just before the predictive parallel gate-level timing simulation. Therefore, for simulation at the higher abstraction level regenerating the new prediction data, we may start the simulation at the higher abstraction level with *the corresponding local module(s) only* in which the functional design change is made, not with the entire design. In our proposed incremental simulation, we will call those local modules that are currently being simulated *active local modules*, and so far not being simulated *inactive local modules*. As long as the effect of the design change is not propagated to the output of the corresponding active local module(s), the remaining part of the entire design needs not to be simulated and remained as inactive local modules. However, once the effect is appeared at the output of the corresponding active local module, we need to include those local module(s) which have a connection with the currently active local module(s) to the newly active one(s). In predictive parallel simulation scheme, as the prediction data for every local module is already available, we can easily observe the propagated effect at the output of active local module(s) by comparing its actual output values with its old predicted output values, and checking their difference.

Therefore, through our incremental simulation, we could efficiently speed up the regeneration process for prediction data by allowing the active local modules to grow spatially in the entire design domain as the simulation time temporally increases. Of course, at the worst, the active module(s) eventually may include the entire design. However, it is clear that the proposed incremental simulation never be slower than the traditional one even at the worst case.

4. Preliminary Experimentation

For experimentally proving the effectiveness of our new prediction strategy, we first compare the prediction accuracy of ours and the relative time of prediction data regeneration with those of previous prediction strategies (Table 1), and the relative simulation speeds of five different simulation methods including ours (Table 2). We use three designs, one from industry and other two from OpenCores [12]. A major commercial Verilog simulator from one EDA vendor, which also

supports multi-core for parallel simulation, is used for the experiment. For our gate-level timing simulation experiment with three designs, the higher abstraction level simulation is gate-level functional simulation, i.e. zero-delay simulation, which is an order of magnitude faster on average.

Table 1: Prediction Accuracy (%), and Normalized Prediction Data Regeneration Time.

Design		AC97	atpg	PIC
Original predictive parallel sim.	A[1]	45.3 %	33.0 %	39.9%
	T[2]	N/A	N/A	N/A
Predictive parallel sim. with accuracy enhancement [11]	A[1]	95.1 %	99.5 %	100.0 %
	T[2]	1	1	1
Ours	A[1]	95.1 %	99.5 %	100.0 %
	T[2]	0.2	0.4	0.02

1: A denotes the prediction accuracy. 2: T denotes the normalized prediction data regeneration time.

First, we have shown the prediction accuracy, and relative prediction data regeneration time through the normalization in Table 1. The original predictive parallel simulation (second row) simply uses the original prediction data obtained from the previous simulation before the functional design change for every local module. Therefore, its prediction accuracy is pretty low for all of the three designs, and would hinder the parallel simulation performance from preventing heavy synchronization and communication overhead. The predictive parallel simulation with accuracy enhancement (third row) tries to raise the prediction accuracy through the regeneration of the prediction data by executing higher abstraction level simulation, i.e. gate-level zero-delay simulation, *with the entire design* after the functional design change [11]. The prediction accuracy is drastically improved for all cases, as expected. Ours (fourth row) also has achieved the high prediction accuracy as much as one in [11], but done it much faster through the proposed incremental simulation, i.e. 2.5x to 50x speed-up.

Table 2: Normalized Simulation Execution Time.

Design (No. of cores)	AC97 (5)	atpg (4)	PIC (3)
Traditional sequential sim.	1	1	1
Traditional multi-core parallel sim.	30.8	0.9	16.7
Original predictive parallel sim.	0.7	0.8	0.8
Predictive parallel sim. with accuracy enhancement [11]	0.9	0.5	0.7
Ours	0.4	0.3	0.5

Next, the relative simulation times through the normalization among different simulation methods are compared in Table 2. First, the traditional sequential simulation (second row), and the traditional multi-core parallel simulation (third row) are both given. Note that the performance of multi-core parallel simulation is much worse in two cases, AC97 and PIC, than that of sequential simulation. This is due to the heavy synchronization and communication overhead, which is very common in gate-level timing simulation. From this, without eliminating the synchronization and communication overhead rigorously, we strongly claim that the parallel gate-level timing simulation is a hopeless effort. Meanwhile, the original predictive parallel simulation (fourth row) had improved the simulation performance, but its further speed-up is bogged down by its low prediction accuracy. Predictive parallel simulation with accuracy enhancement (fifth row) had further improved the performance by raising the prediction accuracy through prediction data regeneration [11]. But, this

time its long prediction data regeneration time became the bottleneck. Ours (sixth row) has efficiently removed this obstacle, and achieved the additional noticeable speed-up for all three cases. Therefore, much faster predictive parallel gate-level timing simulation is definitely anticipated with the proposed approach.

5. Conclusion

In this paper, we have proposed the improved predictive parallel gate-level timing simulation method by utilizing the incremental simulation, which could regenerate the new accurate prediction data much faster, but still accurate when the functional design change invalidates the old prediction data from the previous simulation because of low prediction accuracy. The preliminary experimental result already has shown the effectiveness of the approach, and more result will be shown at the conference presentation.

References

[1] Gate-level Simulation Methodology, Whitepaper, Cadence Design Systems, 2013, [Online]. Available: www.cadence.com

[2] R. M. Fujimoto, "Parallel Discrete Event Simulation," *Communication of the ACM*, vol. 33, no. 10, pp. 30-53, 1990.

[3] 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)*, pp. 211-218, 2007.

[4] D. Chatterjee, A. DeOrio, and V. Bertacco, "Event-driven gate-level simulation with general purpose GPUs," in *Proceedings of Design Automation Conference (DAC)*, pp. 557-562, 2009.

[5] Y. Zhu, B. Wang, and Y. Deng, "Massively Parallel Logic Simulation with GPUs," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 16, no. 3, pp. 1-20, 2011.

[6] J. Gross, et al, "Multi-Level Parallelism for Time- and Cost-efficient Parallel Discrete-Event Simulation on GPUs," in *Proc. of 26th ACM/IEEE Workshop on Principles of Advanced and Distributed Simulation (PADS 2012*), 2012.

[7] K. Chang and C. Browy, "Parallel Logic Simulation: Myth or Reality?," *Computer*, vol. 45, no. 4, pp. 67-73, 2012.

[8] D. Kim, M. Ciesielski, and S. Yang, "A new distributed event-driven gate-level HDL simulation by accurate prediction," in *Proc. of Design and Test Europe (DATE)*, pp. 547-550, 2011.

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

[10] S. Yang, et al, "A New Distributed Parallel Event-driven Timing Simulation for ECO Design Changes," in *Proc. of SIMUL-2013*, pp. 169-173, 2013.

[11] S. Yang, et al, "Predictive Parallel Event-driven HDL Simulation with A New Powerful Prediction Strategy," in *Proc. of Design and Test Europe (DATE)*, no. 316, 2014.

[12] Open Cores, [Online]. Available: www.opencores.org