A Fast Two-pass HDL Simulation with On-Demand Dump

Kyuho Shim, Youngrae Cho*, Namdo Kim*, Hyuncheol Baik, Kyungkuk Kim,
Dusung Kim, Jaebum Kim*, Byeongun Min*, Kyumyung Choi*, Maciej Ciesielski**, Seiyang Yang

Dept. of Computer Eng., Pusan National University, Korea,
* Samsung Electronic Corp., Korea, ** Logic-Mill Technology, LLC, USA
syyang@pusan.ac.kr

Abstract - *Simulation-based functional verification is characterized by two inherently conflicting targets: the signal visibility and simulation performance. Achieving a proper trade-off between these two targets is of paramount importance. Even though HDL simulators are the most widely used verification platform at the RTL and gate level, their major drawback is the low performance in verifying complex SOCs, especially when the high visibility over the design under verification is required.*

This paper presents a new, fast simulation method as an effective way to achieve both high simulation speed and full signal visibility. It is based on an original two-pass simulation approach. During the 1st pass, with the simulation running at full speed, a set of design states is saved periodically at predetermined checkpoints. During the 2nd pass, another simulation is performed, using any of saved checkpoints and providing 100% signal visibility for debugging. Our method differs from the traditional simulation snapshot approach in the amount and the way the design state is saved. Experimental results show significant speed-up compared to existing traditional simulation methods while maintaining 100% visibility.

1. Introduction

In order to enhance the verification efficiency, EDA vendors and many semiconductor companies have resorted to verification acceleration or tried some advanced methodology. As a result, testbench automation, assertion-based verification, simulation acceleration/emulation, formal and semi-formal [1, 2] verification have been regarded as viable solutions for efficient verification methodology. Despite these efforts, HDL simulation is still of a primary verification tool in functional verification. Besides 100% visibility, simulation has many good capabilities, e.g., ease of use, flexibility, low cost, etc. However, as the size of SOCs is getting larger, the simulation performance has been severely lagging. There are several attempts to accelerate performance of simulation. A technique which uses multiple simulators on parallel processors is one of those attempts [3]. HW/SW co-simulation sacrifices some verification accuracy, i.e., cycle-accurate or instruction-accurate, in the SW domain to boost verification performance for the entire design domain [4]. Emulator has emerged as HW-assisted simulation acceleration tool [5, 6]. However, none of those is as widely used as HDL simulators do because each of them has its own drawbacks, e.g. versatility, accuracy, cost, etc.

Most of the designers and verification engineers want to have an ideal HDL simulator with best possible capabilities: having both 100% visibility and high performance. However, these two factors conflict with each other. One of the major factors contributing to the poor performance of simulation is signal dumping throughout the entire simulation. Many experiments have shown that the simulation speed decreases significantly, in the range of 5-20x compared to the one without any signal dumping. However, the root causes of design bugs in design under verification are mostly behind the veil and are hard to predict in advance; and dumping of all signals is crucial and necessary to explore the whole problem space for the purpose of debugging.

This paper presents a smart simulation method which simultaneously addresses two targets, fast simulation speed and high visibility. A smart simulation method adopts a two-pass simulation scheme. Instead of running a single simulation, with or without a dump, based on inaccurate guess, the idea is to perform two consecutive simulation runs; the first contributes to simulation speed and the second one to high signal visibility, without sacrificing simulation speed.

2. Previous Approach and Motivation

2.1 Previous Approach

Marantz [7] proposed a signal reconstruction method for achieving 100% signal visibility. His idea has recently been commercialized by Novas[8]. The basic idea is to save the values of all flipflops, latches, memories, and inputs of a design, or part of the design, during the simulation. Later, a separate reconstruction or interpolation engine computes the values of missing combinational signals on demand. While this approach can provide 100% signal visibility without dumping all signals during the simulation, the simulation time can be reduced to some extent. However, its benefits are quite limited

978-1-4244-1922-7/08/$25.00 ©2008 IEEE

as it requires saving of values of all flipflops, latches, memories and inputs, event by event, during the entire simulation run. Marantz and Novas assume that the number of flipflops and latches is about 5-20% of the total signals in the design. Whether this assumption is universally valid or not, the fact is that, in today's SOC designs, the number of flipflops and latches is much larger than the number of inputs. Dumping of all flipflops and latches is still an expensive proposition, significantly hampering the simulation (we will support this argument quantitatively later). Another problem is its cycle-based reconstruction or the interpolation method. Cycle-based computation is necessary for fast re-calculation of missing signals values on an on-demand basis. However, any timing-related information will be lost so that it cannot be applied to gate-level timing simulation.

Another well-known method is to use the save/restart feature of HDL simulators based on checkpoints. However, this traditional checkpoint method has a problem when the design size increases; the size of simulation state checkpoint file also rapidly increases because traditional checkpoint feature built in the commercial HDL simulators saves the entire binary image of simulation, which can be prohibitively large.

2.2 Preliminary Experiments and Analysis

To justify the above observations quantitatively, we conducted two simple but conclusive experiments. One was to investigate the relationship between the simulation time and the number of dumped signals during the simulation. As shown in Table 1, the simulation time increases almost linearly as the number of dumped signals increases.

Table. 1. Simulation time vs. the size of signal dump

	Percentage of dumped signals(%)	Simulation time (sec)
Design: *AES10*	0 % (No dump)	322
	10 %	662
	20 %	884
	40 %	1,335
	80 %	2,233
	100 % (full dump)	2,687

In the other experiment, we measured the size of the simulation state checkpoint file and the dump overhead for a real SOC industrial design, containing about 18M gates. As shown in Table 2, the size of the simulation state checkpoint file is very large and the simulation time for dumping all signals increases 22 times compared to the simulation without the dump.

Table 2. The dump overhead and the size of simulation state checkpoint file for a large design

Design: *Mobile-SOC*	Dump	Simulation time (sec)	Simulation Checkpoint file size (MB)
	No dump	28,264	240
	Full dump	624,728	

We have observed that many real large SOC designs from industry have significant dumping overhead which increases the simulation time 10 to 20x, or more. This overhead is much higher than we have anticipated. Fig. 1 shows the expected simulation time for such designs when only 5-20% signals are dumped during the simulation run for a design at Table 2.

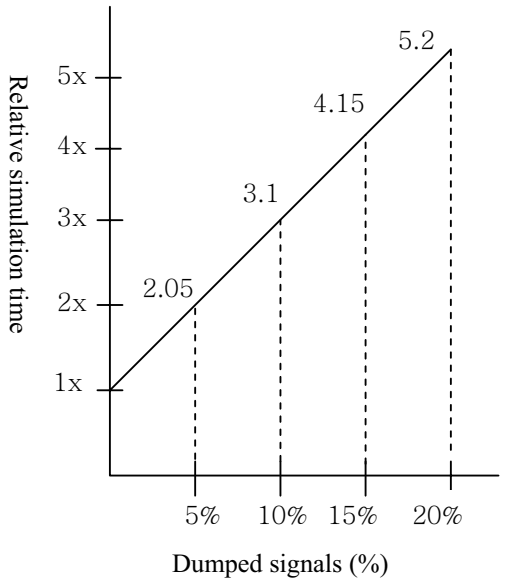


Fig. 1. Estimated simulation overhead for partial dump for a large design with22x simulation overhead for full dump

The conclusion from this simple analysis is that even such a small (5-20%) partial signal dumping is unacceptable as it increases the simulation time significantly for big designs. For example, if the full dump increases the simulation time 22x, even a 10% partial dump will slow down the simulation 3.1x over the simulation with no dump. For 20% partial dump the slowdown is 5.2x. These large overheads for providing an afterward on-demand 100% signal visibility can be hardly justified even if high visibility is absolutely required. However, if support for debugging is not required, the simulation runs with partial dump waste the simulation time and disk space. This is especially true for regression testing where most of simulation runs do not require the

debugging. However, the real problem is that, prior to a specific simulation run with a specific testbench, nobody knows whether the simulation requires debugging or not.

Therefore, an overhead for providing the on-demand 100% signal visibility must be much lower than those in [7, 8] so that the simulation can be run with light-weight partial dump guarantying the subsequent on-demand 100% signal visibility. The approach of partial dumping method in [7, 8] is to dump values of all flipflops, latches, and memories in the design during the entire simulation time. In the next section, we propose a much smarter and efficient way to minimize dumping overhead for partial dump, guaranteeing 100% on-demand visibility afterwards.

3. Smart Simulation for On-Demand Replay

Based on the previous experiments and quantitative analysis, we concluded that in all traditional simulation methods high simulation speed and high signal visibility are impossible to achieve simultaneously. To this end we propose a much better simulation method for achieving 100% signal visibility which does not increase the simulation time significantly over the simulation with no dump, even for a very large design.

We adopt a new checkpoint strategy, which differs from the one built in commercial HDL simulators. Instead of checkpointing the entire simulation image, we generate and use a set of *design state checkpoints*. A design state checkpoint DCP(i) consists of an input sequence IS(i) and a state value of DUV (Design Under Verification) SV(i). The state value of DUV consists of values of all flipflops, latches, and memories in DUV *at a specific simulation time*. Therefore, DUV can be re-simulated with a design state checkpoint by initializing its fliplops, latches, and memories of DUV with SV(i) and applying input sequence IS(i) to DUV.

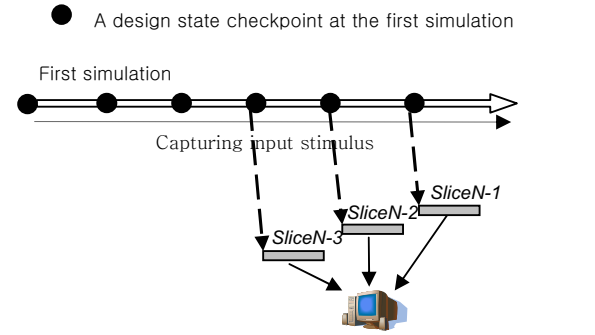
Our simulation method consists of two passes.

Pass 1: During the 1st simulation the values of design states are saved *periodically,* at regular intervals at checkpoints, and input values are stored *continuously*. This simulation is mandatory.

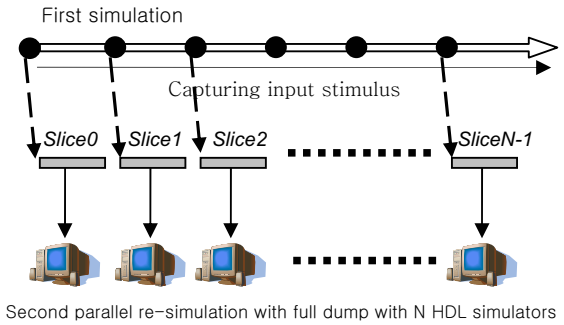
Pass 2: The 2nd simulation, which can re-start from any of the saved checkpoints, is optionally run only when the high visibility of the design signals is required. This 2nd simulation can be run instantly on an on-demand basis.

Fig. 2 shows our two-pass simulation method explained above. There are two major differences between our method and the previous methods [7, 8].

1. First, instead of indiscriminately dumping values of all flipflops, latches, memories, and inputs during the entire simulation time, we only dump the values of all inputs, while the values of all flipflops, latches, and memories are dumped only periodically, at the predetermined checkpoints, during the first simulation run. (We will call values of all sequential elements *state values*, and the values of all inputs *input values*) only. More specifically, the state values are saved only periodically, e.g. every 50,000 *nsec*. The simulation can be re-started from any of these periodically saved state values, using the saved input values as stimulus after initializing the state of the design with the saved state values.
2. Second, instead of running a separate cycle-based reconstruction engine, as in [7, 8], we re-run the second simulation with a conventional HDL simulator. This has an additional benefit that our replay naturally supports timing simulation as well as functional simulation.



(a) Serial Mode Replay



(b) Parallel Mode Replay

Fig. 2. Fast simulation having on-demand replay capability

Since the number of flipflops and latches in a

design is typically much larger than the number of inputs, the benefit of our method is obvious. That is, the dumping overhead for saving those two groups of signals (state values, input values) at the first simulation is greatly reduced, compared to the methods in [7, 8]. Again, this is because the state values are saved periodically during the simulation run and only input values are saved continuously. Therefore, the overhead for generating multiple design state checkpoints in our method is much lower than the overhead for saving state values and input values continuously adopted in [7, 8].

However, the frequency of saving the design state is important and should be carefully determined for following reasons.

1. If the design state saving frequency is too high, then the speed of the first simulation for capturing both state values and input values may become significantly slower.
2. If the design state saving frequency is too low, then the second re-simulation for providing 100% visibility for specific simulation time window will take long time.

Because the frequency of saving the design state can be parameterized as tradeoff between the speed of the first simulation and the second re-simulation, our method based on design state checkpointing is more flexible than the methods in [7, 8]. For example, the frequency of design state saving in the first simulation can be lowered, to say 10 design state checkpoints, during the regression testing where most of the simulation runs do not require signal dumping for debugging. But, it would be raised, e.g. to 1,000 design state checkpoints, during the non-regression testing where many of the simulation runs eventually would require the visibility over DUV, for faster second re-simulation.

For determining an optimum design state saving frequency, we have performed extensive experimentation with a couple of real SOC designs from industry, and obtained a very reasonable answer. We will discuss this later in the experiment section.

After the first simulation, the entire simulation time is divided into a set of time segments (called *slices*) so that the second simulation can be run independently, slice by slice. When the signal visibility is needed for debugging, the second re-simulation run is executed as needed. This second re-simulation run itself can be serial or parallel. Let's assume, for example, that the total simulation time is 10,000,000 nsec, and state saving frequency is 10,000 nsec. That is, the simulation run is divided into 1,000 slices, each slice being allocated 10,000 nsec of the simulation time. If re-simulation of the time window 9,010,200 to 9,110,100 nsec is necessary to support 100% visibility of the design, the first on-demand re-simulation runs from 9,010,000 nsec to 9,020,000 nsec (slice # 902), the second one from 9,020,000 nsec to 9,030,000 nsec (slice # 903), and so on; the last one runs from 9,110,000 nsec to 9,110,100 nsec (slice # 912). All these runs are executed serially, one by one, as shown in Fig. 2(a). However, if several simulator licenses are available (in this case 11 licenses are needed), then all 11 re-simulation runs can be done in parallel, as shown in Fig. 2(b).

Therefore, our method is universally applicable, whether the required visibility window for debugging is wide or narrow, in terms of the simulation time. Namely, the serial mode with a single simulator license will provide 100% visibility for narrow simulation time and the parallel model with multiple simulator licenses will provide 100% visibility for wide simulation time, e.g. 100% visibility for the entire simulation time.

Today, most design companies are using simulation farms where a large number of simulators are simultaneously accessed in multiple design projects. In this simulation farm environment, every designer or verification engineer who had submitted his/her simulation jobs wants to finish them as quickly as possible, and to start the debugging process with some debugging tool (e.g. waveform viewer) as soon as the dump data (e.g. VCD, FSDB, or other waveform files) becomes available. However, when many people are trying to dump all signals for the entire simulation time, a large number of simulators is exclusively owned by those simulation jobs for a long time (for example, for the design in Table 2, one HDL simulator must be exclusively dedicated to dump all signals for about 7 days). In such a simulation farm environment, it is very critical to provide a *fast simulation turn-around-time*, that is, the elapsed time between the simulation start time and the simulation end time with/without signal dump, to each person. For the design at Table 2, the simulation turn-around-time with our method is only about 7.5 hours, while still providing 100% visibility (but, the simulation turn-around-time with conventional simulation for full dump is 7 days = 168 hours). This means the debugging process and more simulation runs after the debugging can start even the same day.

Our approach with a good policy for operating a simulation farm is especially effective for whole simulation job being conducted. An example of good policy is to give the higher priority to the simulation jobs requiring the visibility for narrow simulation time, and the lower priority to the simulation jobs

requiring the visibility for wide simulation time. Under such a policy, the simulation turn-around-time for each person becomes drastically shorter because every simulation run can be finished almost as soon as the simulation run without no dump, while generating a set of design state checkpoints for *possible* replay. This way, the simulator licenses are not exclusively owned by single simulation jobs that require all signals dump, for long time. Later, only those simulation jobs that require the visibility for debugging are re-simulated with our on-demand replay method. Simulation jobs that require the visibility for narrow simulation time can be replayed in serial mode with high priority, and simulation jobs require the visibility for wide simulation time can be replayed in parallel mode in low priority. But, as both of them can finish their simulation jobs and generate the signal dump files drastically sooner than any other traditional methods, the simulation turn-around-times for each simulation can be uniformly and greatly reduced.

3. Experiments

We applied our fast simulation technique with a guaranteed on-demand 100% visibility to a large SOC design (about 18M gate equivalent) and one medium SOC design (about 1.5M gate equivalent) from industry. The RTL simulation was performed on SUN workstation (UltraSPARC) with a Verilog HDL simulator from a major EDA vendor. There is a number of interesting points for observation.

First, the simulation time of the 1st simulation is increased only by 6-13% (depending upon the frequency of saving design state), compared to the simulation time with no dump, for an 18M gate large design. This is much faster than the simulation with all essential signal dump used in [7, 8] for guarantying 100% signal visibility at the re-construction phase. The simulation time of the 1st simulation with all essential signal dump would be increased by 205% (if the essential signal is 5% among all signals in DUV) to 520% (if it is 20% among all signals in DUV) compared to the simulation time with no dump. These numbers clearly show that the essential-signal-based reconstruction method can be prohibitively expensive for current large designs. The simulation time of the 1st simulation is increased by 42-48%, compared to the simulation time with no dump for a 1.5M gate medium size design. The simulation time of the 1st simulation with all essential signal dump would be increased by 21% (if the essential signal is 5% among all signals in DUV) to 84% (if it is 20% among all signals in DUV) compared to the simulation time with no dump. Therefore for a 1.5M gate design the time for our 1st simulation is comparable to that of the simulation with all essential signal dump used in [7, 8].

Table 3. Experiments with an 18M gate design

Design Name: Mobile SOC (RTL)		Simulation time (sec)	Dump file size or Checkpoint file size (M Bytes)
No dump		28,284	0
Full dump		624,728	(approx.) 50,000
Ours (10 slices)	1st simul	29,868	980
	2nd simul	60,214	
Ours (100 slices)	1st simul	30,722	2,107
	2nd simul	5,807	
Ours (1,000 slices)	1st simul	31,868	3,702
	2nd simul	587	

Table 4. Experimentation with a 1.5M gate design

Design Name: MediaPlay-SOC (RTL)		Simulation time (sec)	Dump file size or Checkpoint file size (M Bytes)
No dump		8,559	0
Full dump		43,796	(approx.) 33,329
Ours (100 slices)	1st simul	12,118	1,521
	2nd simul	669	
Ours (1,000 slices)	1st simul	12,694	1,685
	2nd simul	24	

Second, the dump file sizes for the 1st simulation are only 2-7% of the file size for all signals dump for an 18M gate design, and 4.5-5% of the file size for all-signals dump for a 1.5M gate design. These file sizes are also smaller than those for storing essential signals which is between 5% and 20% of all signals.

Third, there is no noticeable simulation time increase when the frequency of design state checkpoint is increased from 100 to 1,000. This fact is justified by analyzing the design state checkpoint file, which consists of two sets, (state values, input values). The size of the file for storing state values is proportional to the design state checkpoint frequency. However the size of file for storing input values is almost independent of the design state checkpoint frequency, and the file size for storing states values becomes similar to the file size for storing input values when the number of slices is 1,000. This is an

important observation because the large number of slices does not increase the simulation time as long as its file size is comparable to the file size for storing input values. As mentioned earlier, the higher the design state checkpoint frequency at the 1st simulation, the faster the 2nd re-simulation that provides 100% visibility over DUV, to the point that it can be done as instant on-demand base. For example, for obtaining 100% signal visibility in DUV for the simulation time (one specific slice among 1,000 slices) takes only 587 sec for the large design, and 24 sec for the medium design (Note that the replay times for different slices could be different in the event-driven simulation). Therefore, the slice by slice 2nd re-simulation can provide instant on-demand signal visibility for the entire simulation time without a long waiting time.

We also would like to point out that the traditional simulation checkpoint method cannot be used because of its large simulation checkpoint image size, e.g. 240MB/checkpoint for an 18M gate design at Table 3. If we used the same 1,000 checkpoints for fast replay, the file size for 1,000 checkpoints would become 240GB, compared to 3.702GB of our design state checkpoint method, and 50GB dump file size of the traditional full dump method. We can easily conclude that the simulation time increases drastically when 240GB simulation image files are written into the disk during the simulation.

Based on our experiments, we can claim that our fast simulation method having instant replay capability is much better for both reducing the simulation time and achieving 100% visibility over any other traditional methods. It also has the smallest disk usage. Parallel re-simulation with multiple HDL simulators over multiple slices even can provide 100% signal visibility over very long, possibly the entire simulation run.

4. Conclusions

This paper has presented an effective, new two-pass simulation method to achieve full visibility without sacrificing simulation speed. It is smart enough to boost the simulation and debugging productivity with no new investment. Also it may reduce the simulation turn-around-time of every simulation job in the simulation farm environment with limited number of HDL simulator licenses. Therefore, for the current SOC design verification that requires much longer simulation cycles, we expect that the proposed approach will yield a very good overall improvement in the verification productivity. Experimental results with two real designs from industry have shown 3.7x and 20x simulation speed-up compared to a conventional full-dump simulation method, while maintaining a 100% visibility.

[References]

[1] R. Kumar, “Formal Verification of Hardware: Misconception and Reality”, WESCON/98, pp. 135-138, Sept. 1998

[2] Yuan Lu, Weimin Li, “A semi-formal verification methodology,” ASIC, 2001. *Proc. 4th International Conference,* pp. 33-37, Oct. 2001

[3] R. M. Fujimoto, “Parallel Discrete Event Simulation,” Communication of the ACM, Vol23, No.10, pp. 31-53, October 1990

[4] J. A. Rowson, “Hardware/software Co-simulation,” *Proc. 31th Design Automation Conference*, pp. 439-440, June. 1994

[5] Palladium Datasheet, Quickturn (http://www.quickturn.com), 2007

[6] Veloce Datasheet Mento Graphics, http://www.mentor.com, 2007

[7] J. Marantz, “Enhanced Visibility and Performance in Functional Verification by Reconstruction,” *Proc. 35th Design Automation Conference*, pp. 164-169, June 1998

[8] Y. C. Hsu et al, “Visibility Enhancement for Silicon Debug,” *Proc. 43rd Design Automation Conference,* pp. 13-18, July 2006