A Parallel and Accelerated Circuit Simulator with Precise Accuracy

Peter M. Lee, Shinji Ito+, Takeaki Hashimoto+, Junji Sato,Tomomasa Touma+, and Goichi Yokomizo
Semiconductor and IC, Hitachi, Ltd
+Hitachi ULSI Systems Co., Inc.
E-mail: lee-peter@sic.hitachi.co.jp

Abstract

We have developed a highly parallel and accelerated circuit simulator which produces precise results for large scale simulation. We incorporated multithreading in both the model and matrix calculations to achieve not only a factor of 10 acceleration compared to the de facto standard circuit simulator used worldwide, but also equal or exceed the performance of timing-based event-driven simulators with the accuracy which matches that of SPICE-based circuit simulation. For example, a 89K element DRAM CAS circuit simulation can be performed in under 38 minutes with timing accuracy error as little as 7 ps.

1. Introduction

Conventional SPICE-based simulators have been the workhorses of circuit design to predict performance from early to late stages of the design. These simulators are all based on the simulator SPICE first developed by University of California, Berkeley, in 1975 [1]. There are now many simulators out on the market and also developed internally for simulation within larger organizations.

Although logic products, due to their enormous scale, are based on higher-level simulation using basic libraries, SRAM and DRAM products require an increasingly larger scale of circuit level simulation. Because of this trend, a new breed of timing-based event-driven simulators have been developed which promise from 100 times performance increase while sacrificing very little in accuracy.

However, to be able to use these accelerated simulators, much time must be spent in adjusting options to obtain the required degree of accuracy. In many cases, to achieve accuracy, performance drops so that actual simulation time approaches that of the SPICE-based simulators. Although purely digital circuits may have more tolerance towards error than full analog circuits, SRAM and DRAM circuits have many circuit blocks which are more analog than digital in nature, thus making accuracy crucial in timing investigations. Furthermore, as clock speed goes up and the contribution from parasitic elements (such as interconnect and power supply network R, L, C, extracted from layout extraction tools) increase in the deep sub-micron to sub-sub-micron regime, predicting precise timing becomes more and more difficult with event-driven timing simulators as the entire chip becomes more analog in nature.

In this paper, we introduce an internally developed SPICE-based simulator in which we implement multithreading under the shared memory symmetrical multiprocessing (SMP) concept. We chose the shared memory methodology as opposed to distributed memory methodology, such as that of [2,3], because of the tendency for communication between processor nodes to increase due to the nature of IC circuits, and the fact that multithreading is more generally available on engineering workstations. To achieve this simulation speed up, we implement advanced multithreading in both the model and matrix calculation portions of the simulator. To preserve accuracy, we retain analytical device models and do not use table look-up models, and we do not perform any parasitic RC reduction which can greatly affect transient waveform results within the simulator.

Section 2 describes the multithreading concepts implemented into the simulator. Section 3 shows benchmark results on five benchmark circuits compared with a de facto standard SPICE-based simulator and an event-driven timing simulator. Section 4 gives a brief description of actual use in product design, and finally, Section 5 ends the paper with a conclusion.

2. Multithreading algorithm implementation

Fig. 1 shows the basic configuration of our circuit simulator, which is quite typical of all SPICE-like circuit simulators. There is a pre-processing portion which parses the circuit netlist (which describes the circuit configuration) and loads the appropriate data structures.

Next, the matrix representing the circuit is created. Actual DC and transient analysis occurs next, with repetitive device model calculations and matrix calculations for each iteration, each DC point, and each timestep. Finally, the output routine dumps simulation results to a file.

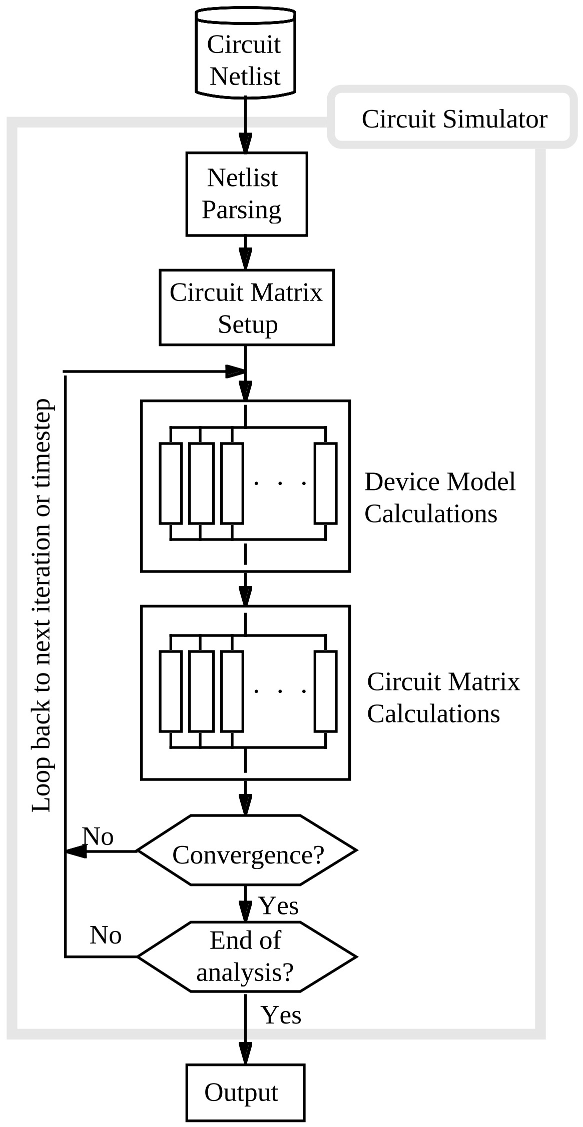

Fig. 1 Basic configuration of our circuit simulator. Multithreading is implemented in the device model calculations and circuit matrix calculations, which occur at every calculation iteration.

The most time consuming portions of the calculations are undoubtedly the model and matrix calculations. When the size of the circuit increases, model calculation time generally increases linearly with the number of devices. However, matrix computational time increases approximately to the power of 2 of the number of nodes in the circuit. Thus, for large-scale circuits, matrix computation will tend to dominate the total simulation time.

As shown in Fig. 1, we thus implemented multithreading in both the model and matrix calculations of our simulator. All multiple threads are generated and administered using the standard POSIX thread library for portability. As simulation accuracy received top priority, we did not implement any table look-up models (we use the full analytical models used in SPICE-based simulators), and we did not implement any algorithms which reduce parasitic RC elements.

To demonstrate the improvements seen for the multithreading implemented in actual circuit simulations, we present three benchmark circuits, one RAS circuit and two DRAM CAS circuits, as summarized in Table 1. All simulations are done using Hitachi's 9000 N4000 8-CPU workstation, which contains 8 PA-RISC 8600 CPU's running at 550MHz.

The next sub-sections explain in more detail the multithread implementations and their effects in speeding up our circuit simulator.

Label	Circuit Type	No. of Elements
Circuit 1	DRAM RAS	38k
Circuit 2	DRAM CAS 1	89k
Circuit 3	DRAM CAS 2	175k

Table 1 List of three benchmarks used for detailed analysis of circuit simulator performance. The circuits are ordered in increasing scale.

2.1 Model multithread implementation

We have implemented multithreading in the model calculations by dividing the calculations evenly among all devices and device types. We reduce the need for synchronization by calculating all types of devices in each thread, and carefully arranging the algorithm so that writing into the conductance matrix for each device do not conflict. Only one synchronization is done before writing into the conductance matrix, and one more synchronization occurs before proceeding to the matrix calculation.

Fig. 2 shows the acceleration factors we achieved for one of our benchmark circuits, a 38K element DRAM RAS circuit. Revision A is as first implemented which is after eliminating conflicts in loading the matrix from different threads. We performed successive revisions B and C, in which further improvements were done to reduce the time to load the conductance matrix and enhance the thread generation step. Although this circuit is fairly small scale compared with the other examples, note we have achieved a maximum of almost 5 speed-up factor in the model calculation section alone for 8 CPU parallel computation. Note that the behavior shows very little saturation, so that more speed-up could be achieved using a higher number of parallel CPU's.

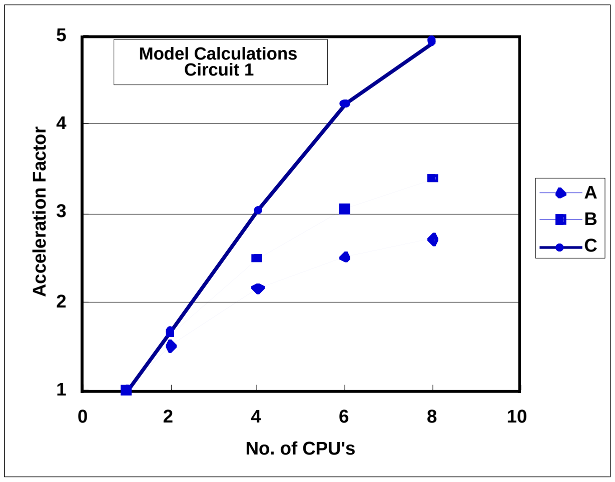

Fig. 2 Acceleration factor versus number of parallel CPU's for model calculations for Circuit 1 showing the effects of improvements A, B, and C, in multithreading and synchronization.

Fig. 3 shows the model calculation acceleration factor for all three circuits Circuits 1, 2, and 3, after all the improvements were implemented. Note that the speed-up factor in this case is fairly independent of the scale of the simulation, as we were able to reduce the overhead due to multithreading and synchronization to a minimum. However, note that the larger scale circuits Circuit 2 and 3 show slightly less saturation behavior, thus promising more speed-up with a larger number of parallel CPU's. Maximum acceleration in this case is achieved by Circuit 2 with a factor of 5.7.

2.2 Matrix calculation multithread (MVA Algorithm) and advanced thread synchronization

For matrix calculation multithreading, we employ an algorithm named Maximal Vectorization Algorithm (MVA) developed earlier in [4] for use in vector-processor-based supercomputers. This is a methodology which pre-determines the calculations necessary for LU decomposition and groups the calculations into sets of independent operations, where operations in one set can be executed in parallel. One "level" denotes one set of calculations which are independent of each other, with the next level being the next set of independent operations to be done in parallel but must occur after the calculations of the previous level, as calculations depend on this previous level. Because parallel calculations at each level must all finish before calculations of the next level start, thread synchronization must occur between each level. The amount of overhead that occurs during synchronization greatly affects the speed-up factor, especially if the original amount of calculations is small. Thus, this method is better suited for larger scale calculations.

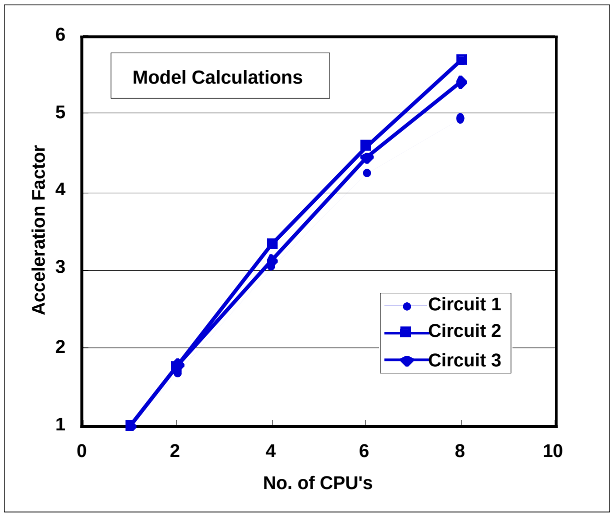

Fig. 3 Acceleration factor of the model calculations versus number of parallel CPU's for Circuits 1 through 3, which are in order of increasing size. The model multithreading has been optimized so that in this case there is not much difference between circuits of different scale. Here we can achieve about 5.7 maximal acceleration factor for 8 CPU's, with little saturation.

In Fig. 4, we show the speed-up obtained in the matrix calculations using the MVA algorithm on Circuit 1, the relatively smaller scale circuit. Here, we improved the method of synchronization from A, B, to C by decreasing the number of thread calls and optimizing the wait time for thread synchronization. Note the large affect on the acceleration results, which show an improvement from a factor of 1.94 to 3.21 at 8 CPU just by changing the synchronization method. Note also that because of the small scale of the calculations, the speed actually decreases for 8 CPU's. Thus, in this case, the optimum simulation occurs at 6 CPU's, which results in a speed-up factor of 3.28.

Fig. 5 shows the results of Circuits 1, 2, and 3 using the optimum synchronization method. Note that acceleration factor increases dramatically as the scale of the circuit increases, from 3.21 to 5.4 in this case for 8 CPU's. Furthermore, the characteristics for Circuit 3 shows less saturation, so that faster performance can be expected with a larger number of parallel CPU's.

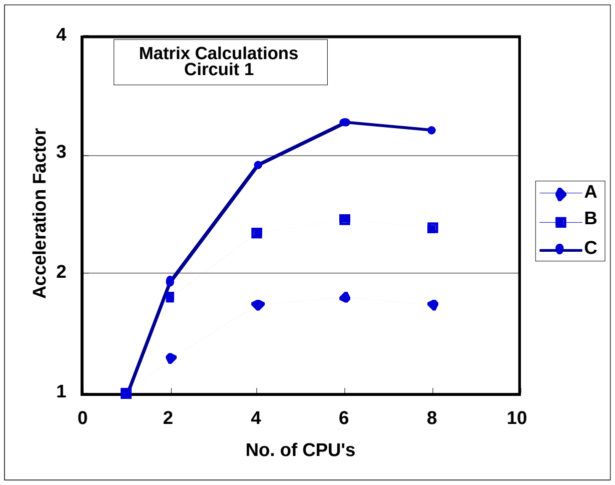

Fig. 4 Acceleration factor versus number of parallel CPU's for the matrix calculation portion using the MVA algorithm. "A" is initial implementation, "B" is first fine-tuning of the multithread synchronization, and "C" is the optimal fine-tuning of the multithread synchronization.

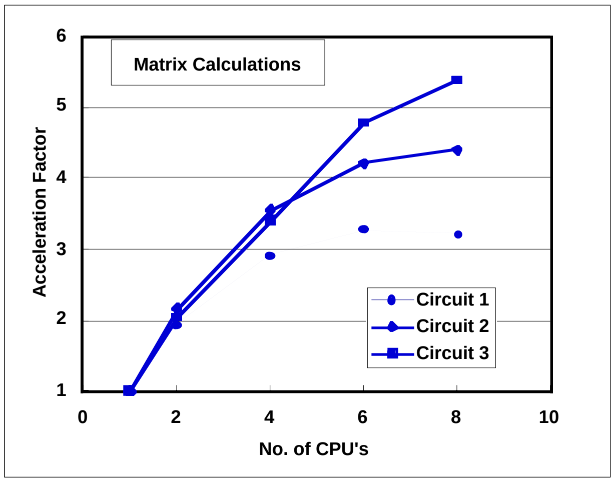

Fig. 5 Acceleration factor of the matrix calculations versus number of parallel CPU's for Circuits 1 through 3, which are in order of increasing size. Note that the speed up is affected by the scale of the circuits, since synchronization between the different levels of the MVA algorithm is present. However, we can achieve almost a factor of 5.4 speed-up for 8 CPU's for the larger circuit, with only some saturation in the characteristics.

2.3 Overall acceleration

Finally, Fig. 6 shows the acceleration factors of the overall simulation. Here the maximum overall acceleration factor is 4.6 for Circuit 2. Slightly more saturation occurs when overall elapsed time is taken into account, since the non-parallelized areas of the simulator outside of the model and matrix calculations start to become visible because of the speed-up of the model and matrix portions seen in the previous sub-sections. Future work will concentrate on further decreasing the elapsed time of this serial portion of the simulator.

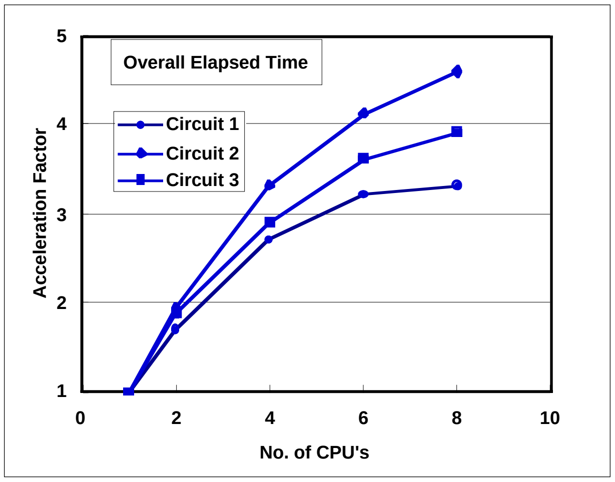

Fig. 6 Overall elapsed time acceleration factor for Circuits 1, 2, and 3. Here we achieve a maximum acceleration factor of 4.6.

3. Performance and accuracy results

This section shows simulation results of our simulator with circuits in the previous section plus two more analog circuit examples, a 4k-element PLL circuit, and a 5.1k-element A-to-D converter (ADC) circuit. Benchmark circuits are listed again in Table 2. Performance is compared with one of the worldwide de facto standard SPICE-based simulators (which we will denote Simulator A) and also with a well-known event-driven accelerated simulator (which we will denote Simulator B). Table 3 shows the nomenclature used.

Label	Circuit Type	No. of Elements
Circuit 1	DRAM RAS	38k
Circuit 2	DRAM CAS 1	89k
Circuit 3	DRAM CAS 2	175k
Circuit 4	Analog PLL	4k
Circuit 5	Analog ADC	5.1k

Table 2 Listing of eight benchmark circuits used for overall performance and/or accuracy evaluation.

Simulator Type	Description
Simulator A	De facto standard circuit simulator
Simulator B	Event-driven timing simulator
Our simulator	Model and matrix multithread circuit simulator

Table 3 Nomenclature used for the three different simulators.

All simulations except for the two purely analog benchmark circuits were done on Hitachi's PA-RISC 8600 based workstations with CPU clock = 550 MHz. As mentioned in the previous section, we performed all non-analog benchmark circuit multithread simulations using the Hitachi 9000 N4000 8-CPU workstation, using the same PA-RISC 8600 CPU running at 550MHz. For the two pure analog simulations, we used an 6-CPU Sun Enterprise 3500 workstation using UltraSparcII chip running at 400MHz for all simulations.

Fig. 7 shows the performance factor of Simulator B (the event-driven timing simulator) and our simulator with respect to the de facto standard circuit simulator Simulator A the five benchmark circuits. Table 4 shows the actual elapsed time of the three simulators. Note that in all cases, the Simulator B and our simulator perform from 7 – 25 times speed-up compared to Simulator A. Note here also that our simulator, which is represented by the darker bars, slightly to substantially exceeds the performance of the event-driven Simulator B for all except Circuit 2 (there is no data for Circuit 5 for Simulator B as it did not predict the correct waveform).

For the performance obtained in Fig. 7, we also compared the timing errors for each of the digital Circuits 1, 2, and 3, as shown in Fig. 8. The waveform used as the standard for comparison was Simulator A using accurate simulation mode (the performance factor of Simulator A in Fig. 7 and the error of Simulator A in Fig. 8 are, in contrast, found by simulating Simulator A in normal mode, which is faster than accurate mode).

Note that the error between Simulator A and our simulator are very comparable. However, the error of the event-driven Simulator B was substantially larger than either of the circuit simulators. Note that by comparing Fig. 7 with Fig. 8, Simulator B is both slower and has larger error than our simulator for Circuit 1 and Circuit 3. Thus, for these two circuits, we judged that no advantage could be gained by fine tuning Simulator B simulations any further.

Circuit 2 was one example where Simulator B was faster than our simulator (32 min. versus 38 min.). We thus tried improving the accuracy of Simulator B for Circuit 2. Increasing accuracy slowed Simulator B until, at the point where Simulator B became slower than our simulator (45 min. versus 38 min.), the error was reduced from 219 ps down to 172 ps. However, this 172 ps error is still much larger than the 7.1 ps error of our simulator. We thus concluded that it was not possible to offer the same performance and accuracy with Simulator B.

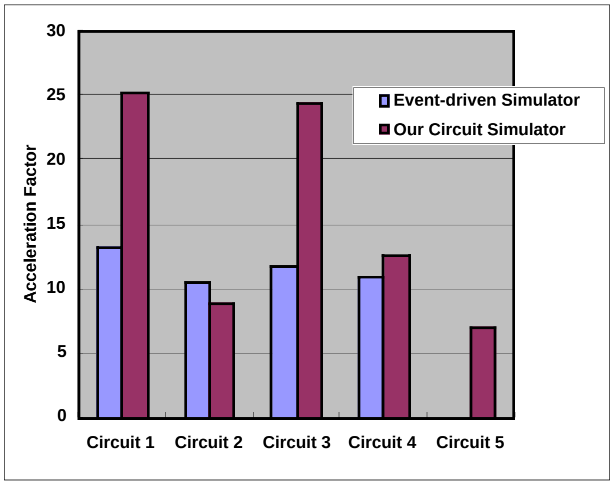

Fig. 7 Simulation elapsed time acceleration factors compared to the de facto standard circuit simulator Simulator A of the event driven timing simulator Simulator B and our circuit simulator. There is no data for Simulator B for Circuit 5 (the ADC analog circuit) as the simulator did not produce correct waveforms.

Label	Simulator A	Simulator B	Our simulator
Circuit 1	91 min.	6.89 min.	3.6 min.
Circuit 2	333 min.	32 min.	38 min.
Circuit 3	2496 min.	214 min.	102 min.
Circuit 4	160.1 hrs.	14.7 hrs.	12.8 hrs. (6-CPU).
Circuit 5	29.1 hrs.	Waveform error	4.2 hrs. (4-CPU)

Table 4 Elapsed time for Circuits 1 – 5 for all three simulators.

For the analog benchmarks Circuits 4 and 5, Table 5 shows the accuracy between Simulators A, B, and our simulator. Simulations were done on the Sun Enterprise 3500 6-CPU workstation. Note that our simulator exhibits exceptional accuracy but with the fastest elapsed time compared with either of the other simulators.

4. Application to products

We have internally released our simulator for usage in design of memory and analog products. In real-product usage, in one site, for a memory product, a set of 9 simulations which required 24 hours was reduced to 1.5 hours, a factor of 16 in speed-up, compared with the previous version of our simulator. We are now having the simulator used for more than 7 products, with a speed-up ranging from a factor of 2 to 16 depending upon the number of parallel CPU's available at the designer's workstations.

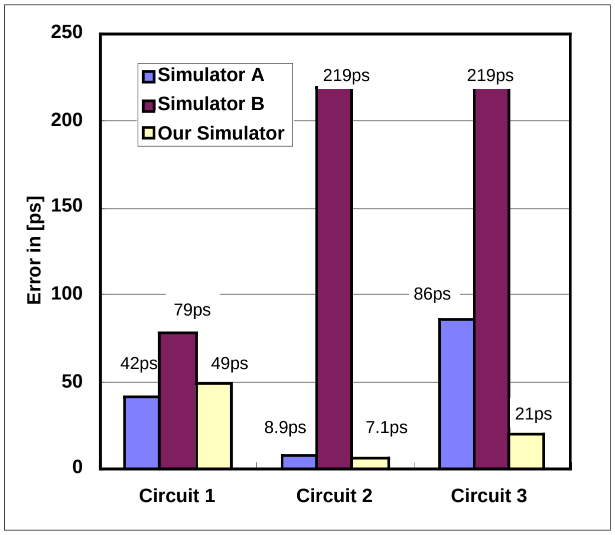

Fig. 8 Error in delay time in picoseconds of the de facto standard circuit simulator Simulator A, the event-driven timing simulator Simulator B, and our circuit simulator.

Label	Error Criteria	Simulator A	Simulator B	Our simulator
Circuit 4	Peak Voltage Error	1.1%	1.4%	1.3%
	Amplitude Error	31.6%	3.9%	3.9%
	Period Error	63.4%	7.75%	8.22%
Circuit 5	Output stepsize	0%	Waveform error	5.3%

Table 5 Accuracy figures for the two analog benchmark circuits.

5. Conclusion

We have developed a highly parallel circuit simulator while preserving the tight accuracy of conventional circuit simulations. We have implemented multithreading in both the model and matrix calculation portions and have achieved maximum acceleration factors of 5.7 and 5.4, respectively, with only slight saturation, for 8 CPU's. For the benchmarks we have listed, we achieved an acceleration factor of 10 or more compared to a widely used de facto circuit simulator, and we show comparable or faster performance than an event-driven timing simulator while retaining the accuracy of normal circuit simulation. For example, we have performed a 89k – element circuit simulation in 38 min. within a propagation delay accuracy of 7 ps.

Acknowledgements

We would like to acknowledge the many people who contributed to the development and evaluation of our circuit simulator. We would like to thank H. Yasushima for the analog benchmark circuit analysis. We would like to thank K. Ono and K. Yuyama of Device Development Center for much advice and for providing the computing environment to do the simulator development and testing, T. Wada and H. Kitajima for providing the initial push to start the development and provide helpful suggestions and feedback in terms of product application, and Y. Okamura and K. Ise for evaluation for analog applications and general help and support. Many thanks goes to H. Kono of Hitachi ULSI Systems Co. for participating in many of the discussions on multi-threading concepts.

References

[1] L.W. Nagel, "SPICE2 – A computer program to simulate semiconductor circuits, " Univ. of California, Berkeley, ERL Memo ERL-M520, May 1975.
[2] K. Hachiya, T. Saito, T. Nakata, and N. Tanabe, "Enhancement of Parallelism for Tearing-based Circuit Simulation," ASP-DAC Technical Digest, 1997, pp. 493 – 498.
[3] T. Kage, J. Niitsuma, K. Teramae, S. Shimogori, and Y. Izuta, "PARACS: A parallel circuit simulator," 5th Karuizawa Workshop on Circuits and Systems, April 1992, pp. 213-224.
[4] F. Yamamoto and S. Takahashi, "Vectorized LU Decomposition Algorithms for Large-Scale Circuit Simulation," IEEE Trans. on CAD, Vol. CAD-4, No. 3, July 1985, pp. 232 – 239.