Expression-level Parallelism for Distributed Spice Circuit Simulation

Dylan Pfeifer
Electrical and Computer Engineering
The University of Texas at Austin
Austin, Texas, U.S.A.

Andreas Gerstlauer
Electrical and Computer Engineering
The University of Texas at Austin
Austin, Texas, U.S.A.

***Abstract*—Distributed system-level simulation among coordinated, heterogeneous simulators requires communication and synchrony to preserve event causality. Once achieved, multiple coordinated, distributed instances of a single simulator not originally written for internal parallelism can be used to conduct the expression-level parallel execution of a model partitioned into subsystems, such that each subsystem is assigned to an individual simulator. Using a Kahn Process Network simulation backplane for coordination, and a custom Xspice TCP/IP socket device for interfacing, expression-level distributed simulation was applied to observe a decrease of up to 1/52 times the transient analysis time of the same circuit in a single Ngspice instance, without modifying the Ngspice kernel or host execution environment. Up to 128 independent Ngspice instances were coordinated in parallel with this method, with a selectable tradeoff in speed versus accuracy.**

Keywords: distributed Spice simulation, heterogeneous cosimulation, Kahn Process Networks, cosimulation backplanes, expression-level parallelism

I. INTRODUCTION

Distributed circuit simulation techniques can be used to coordinate the parallel simulation of circuit subsystems across multiple, process-independent instances of a single simulator not normally supporting internal parallelism. This is possible if the simulator offers a communication interface at the model expression-level to communicate with other running instances, or to communicate with an arbitrating software agent such as a simulation backplane [1]. However, since each concurrent simulator may advance time independently (*local virtual time)* [2], a coordination solution must both communicate signal values and enforce event causality, such that the *local causality constraint* (LCC) is observed. The LCC requires that concurrent simulators process external events in time step order [2]. Techniques addressing causality in distributed simulation are covered in depth in the literature in the field of Parallel Discrete Event Simulation (PDES) [3], and discrete event/continuous-time cosimulation [4].

If a coordination solution is achieved, it can be applied to conduct the multiple-simulator, parallel execution of a Spice circuit at the model expression level for the speed up of an otherwise long-duration, sequential transient circuit analysis. This is important for modern VLSI device counts, where Spice simulation becomes "typically infeasible for designs larger than 20,000 devices" [5]. Where parallel techniques can be employed, the non-linear increase in transient analysis time per device count for sequential executions can be reduced by conducting the Spice model-evaluation phase or the matrix-solving phase of each time point iteration in parallel. Reported solutions, however, require hardware acceleration (such as offloading to a GPU [5] or FPGA [6]), may not parallelize both the model-evaluation and the matrix-solve phases [6], or require changes to the Spice kernel (all cases reviewed). With the expression-level approach, however, both model-evaluation and matrix-solving phases are parallelized in software, without modifying the Spice kernel [7] or host environment, at the cost of a selectable tradeoff in execution speed versus accuracy.

For distributed simulator coordination, we used the SimConnect/SimTalk infrastructure [8] to speed up the transient analysis time of a counter circuit of more than three thousand transistors. We wrote an Xspice [9] user TCP/IP socket device, which allowed concurrently running Ngspice instances to exchange node information with the SimConnect simulator backplane. Parallelism was employed across 2 to 128 concurrently running Ngspice instances, for speedup to 52x at less than 10 percent error of measurement, and up to 17x with less than 1 percent error of measurement. Percent error of measurement was arbitrarily reduced, at the expense of longer simulation time, by increasing the resolution of the interpolated event (IE) data types exchanged between the Ngspice instances.

II. RELATED WORK

Parallel execution is not new to Spice circuit simulation, due to its high internal data potential parallelism [6] in the model-evaluation and matrix-solving phases of each time point iteration. Techniques exploiting this through executive means (parallel CPUs, GPU or FPGA offloading, or matrix algorithm techniques) are covered in [5], [6], and [10]-[13], but these methods require modifying the Spice kernel, expensive hardware, and most importantly are not generally ported to other simulators that have no studied internal parallelism. We term this post-model level of internal simulator parallelism, "execution-level parallelism." That is, the parallelism is not carried out at the model description layer (the expressive level), but rather at the model execution level, where it occurs "behind the scenes" to some degree from the view of the model writer.

Execution-level techniques are powerful though, reporting up to an 18x speedup of Spice 3f5 benchmarks in [6], without loss of accuracy, by offloading the model evaluation phase onto an FPGA. In another approach [5], the expensive BSIM3 transistor model evaluation steps, which "may comprise about 75% of the SPICE runtime [5]," are offloaded to a GPU for parallel execution, for up to a 4x speedup at the expense of single-precision floating-point accuracy (GPU execution) verses double-precision floating-point accuracy (BSIM3 model code). In the matrix-solving phase, domain decomposition methods can be employed to achieve a very large performance increase (up to 870x reported simulation speedup in [13]), but scaling limits the approach at around 400k nodes as the execution is hosted in only a single SPICE3/HSPICE instance. Finally, with multiple technique advancements and supercomputer parallel CPU execution, Sandia National Laboratories' Xyce simulator [14] "demonstrates good speedups (24x on 40 processors)," but results may be limited to "sufficiently large circuits" [6]. These are by no means an exhaustive set of speed up reports, but rather show that significant increases may be obtained at the execution-level through hardware acceleration and software techniques, if they are available and the Spice kernel is modified.

For heterogeneous, distributed simulation, the "backplane" technique ([1], [15]-[17]) has been employed to solve the simulator coordination problem. Simulator backplanes are arbitrating software agents that distribute information across connected simulators and potentially control simulator time advancement. An *interface* must be constructed for each simulator that connects to the backplane to implement the coordination API. The SimConnect/SimTalk client-server backplane architecture described in [8] implements the dynamics of a Kahn Process Network (KPN) [18], such that the tokens of the KPN are *interpolated event* (IE) data types. Interpolated events are defined in [8] and summarized in section III. With this coordination, simulators are synchronized through dataflow, rather than explicit time step control, lessening the backplane API complexity. Additionally, simulators have no process awareness of one another (per the rules of a KPN), only awareness of their input and output FIFOs, which offers easier management as the number of simulators increases.

III. DISTRIBUTED PARALLELISM

We define "expression-level parallelism" to start at the model description layer. At this layer, the description is inspected for points of partition, at which nodes the circuit is expressed as new, independent subcircuits with communication interfaces. Each new subcircuit is assigned to an independent simulator. The entire model then simulates in coordination over the distributed, coordinated instances of the single simulator normally hosting the non-partitioned model. In this way, if a communication interface is offered at the model description layer, parallel execution can be gained for simulators not normally supporting internal parallelism. The cost is the additional communication overhead between simulators (both a computation and latency cost), and the burden of coordinating distributed simulators with independent versions of time advancement (local virtual time).

As an example of a partitioned distribution, Figure 1 illustrates the coordination of eight Ngspice instances connected to the SimConnect server through SimTalk, for a concurrent 8x parallel simulation of the a 128-bit counter described in section IV. The counter is partitioned into subcircuits 16 bits wide, connected at their MSB and LSB nodes via Xspice socket devices.

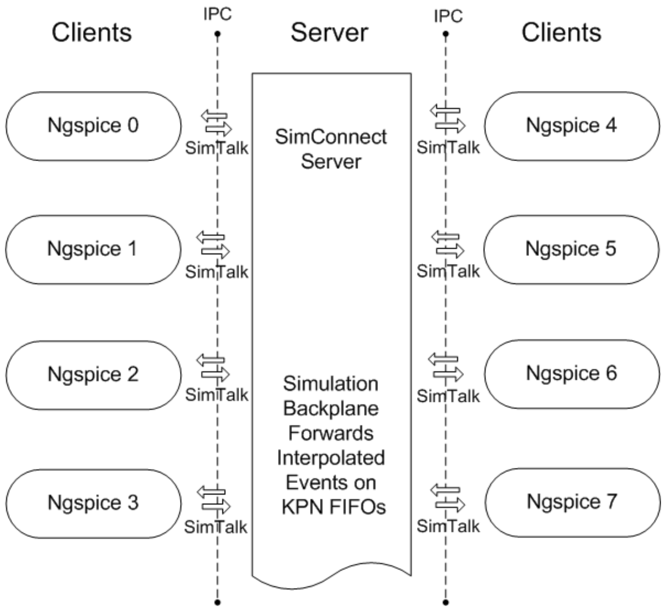

Figure 1. SimConnect/SimTalk relationship for distributed, parallel Ngspice instances

The SimConnect server implements the dynamics of a Kahn Process Network, where the contents of the KPN tokens in are "interpolated event" (IE) objects. Interpolated events, defined in [8], are 3-tuple elements (v, t_m, t_n) from the product set $V \times T \times T$, where { V } is a set of values, and { T } is a set of tags. This nomenclature borrows from the value/tag (v, t) definition of an event covered in [19]. For a given interpolated event (v, t_m, t_n), we define the value v to be constant on the interval [t_m, t_n) specified in the IE, such that the tag set { T } is ordered. { T } is conventionally the real number set $\boldsymbol{R}^1$ in timed, event driven simulations, representing the simulation time stamp of an event occurrence. For an interpolated event (v, t_m, t_n), the range [t_m, t_n) assigns a "stable" time to the signal value v for producers and consumers.

If a simulator consumes an interpolated event, it may assume the value v is constant on the tag range [t_m, t_n), and not need to sample the value again until expiration time t_n. Therefore, an interpolated event encapsulates both communication (the signal value) and synchronization (the start and end time). Mapped to nodes in a Kahn Process Network, simulators consume IEs, run, and produce IEs until the expiration tag of the last consumed IEs, at which point simulators sample their FIFOs again for a new IEs. If input FIFOs are empty, simulators are blocked. Through the blocking read property of KPNs, the *local causality constraint* is observed, because simulators cannot advance in time beyond the expiration tags of IEs on their input FIFOs. Further dynamics of KPN and IEs are detailed in [8].

As a consequence of sampling, there is a tradeoff in speed versus accuracy when using the SimConnect/SimTalk method of IEs for distributed Spice parallelism. Specifically, an IE assigns a stable value for duration to a node voltage, during which local time a consuming simulator can operate on it without re-querying the value. During that time, however, the signal may change, resulting sample-and-hold error for continuous values, or change-delay error for digital values, since the state change information of the digital value is delayed until the start time of the next IE. This speed versus accuracy tradeoff is tunable, however, as explored in section VI.

IV. EXPERIMENTS

Consider simulating a wide-bit asynchronous ripple counter at the transistor level. While ripple counters are impractical as real circuits, due to the rollover delay from maximum value (0xFFF…) to zero, they are simple elementary circuits for conceptualizing or simulating a propagation delay (the rollover delay as the carry bit propagates from bit 0 to bit <n>-1, for counter width <n>). Consider the <n>-bit ripple counter in Figure 2, composed of inverters and positive edge- triggered D flip-flops.

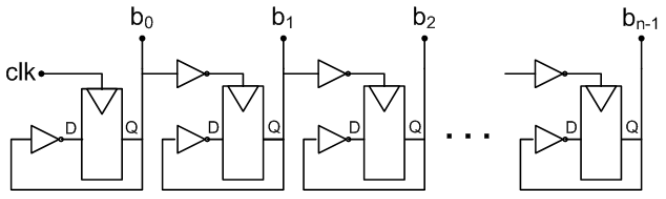

Figure 2. <n>-bit asynchronous ripple counter

The inverters are implemented as a standard pmos/nmos pair, and the D flip-flop at the gate level is implemented according to Figure 3.

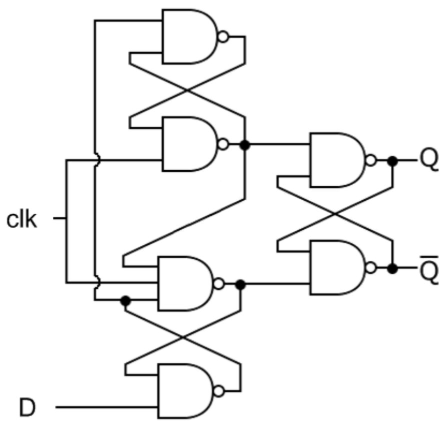

Figure 3. Edge-triggered D flip-flop

Each bit, with two inverters and one flip-flop, consumes 30 MOSFETs, 15 pmos and 15 nmos. The pmos transistors are oversized for symmetric drive strength with respect to the nmos transistors.

Single Instance Simulation

The single-instance counter is simulated in Ngspice [7], the open source distribution of Berkeley Spice version 3 and Georgia Tech's Xspice [9]. Figure 4 shows the increase in transient analysis time as the number of transistors in the circuit increases, per bit width of the counter. The counter is simulated at 4, 8, 16, 32, 64, and 128 bits for 1.5 µs of simulation time.

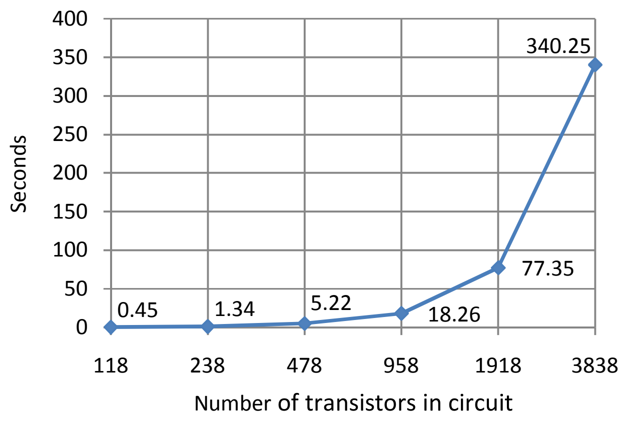

Figure 4. Increase in Ngspice transient analysis time for 1.5 µs of simulation time as counter width increases

From Figure 4, the increase in analysis time per number of transistors is non-linear due to the non-linear increase in model evaluation time and matrix solution time in the Spice kernel as the device count increases. As the number of bits in the counter increases from 64 to 128 bits (1918 to 3838 transistors), for example, the analysis time increases from slightly over a minute to more than five minutes on a single workstation Linux 2.6.16 kernel machine with Intel Xeon 2.93 GHz core. This non-linear increase limits the practicality of simulating complex circuits at the transistor level on the order of modern VLSI transistor counts.

Parallel Simulation

For improvement, the circuit is partitioned at the expression level (the Ngspice circuit deck) into subcircuits <m>-bits wide, where <m> is a power-of-two divisor of 128, and the factor of parallelization. Each subcircuit is then assigned to an independent Ngspice process, coordinated with other Ngspice processes in parallel through SimConnect and SimTalk.

Figure 5 shows an <m>-bit wide subcircuit of the counter, where Xspice user TCP/IP socket devices connect the circuit to its neighboring subcircuits over SimTalk.

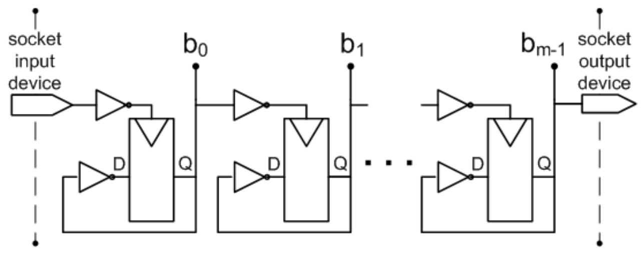

Figure 5. Partitioned subcircuit with socket devices

Between subcircuits, TCP/IP socket devices connect the most significant to least significant bits from one subcircuit to the next. For example, bit 15 of the subcircuit for bits [0:15] is connected to bit 16 of the subcircuit for bits [16:31], and onward through bit 127. The socket device services the SimTalk protocol and delivers IE tokens to the SimConnect backplane, which distributes the IE tokens through KPN FIFOs from signal producer to signal consumer.

V. Results

At 10 ns IE resolution, Figure 6 shows the speedup result per factor of parallelization for the same 128-bit counter for 1.5 µs of transient analysis time.

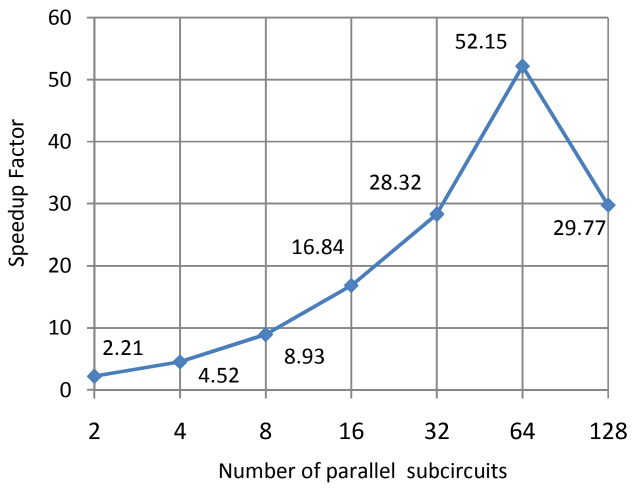

Figure 6. Speedup at 10 ns IE resolution

By dividing the 128-bit counter into two 64-bit subcircuits, we achieve a 2x speedup alone, and then achieve a 52x speedup by subdividing into 64 subcircuits, each 2 bits wide. However, the speedup maximizes at this point, after which it diminishes as the communication overhead per number of Ngspice instances increases. This manifests in the loss of speedup from 64x to 128x parallel in Figure 6. The cost of fixed-resolution IE duration also results in a non-zero percent error of measurement, shown in Figure 7, where the rollover time of the ripple counter across the parallel cases is measured against the rollover time of the non-parallel case.

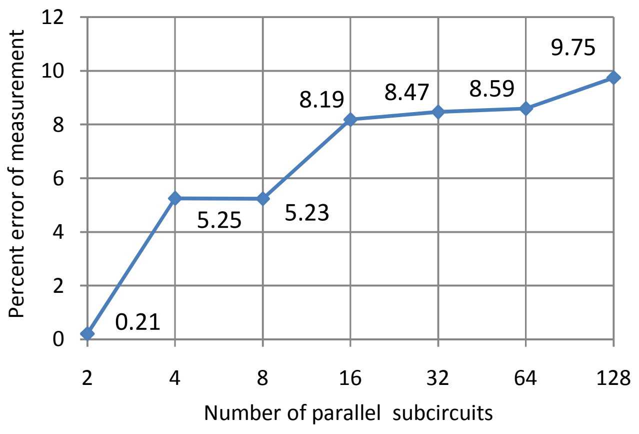

Figure 7. Percent error of measurement at 10 ns IE resolution

The error in measurement occurs because the IEs are of finite duration, during which sample time an event is declared constant. If an IE duration is greater than the rail-to-rail fall time of a circuit inverter, for example, or on the order of it, conveying the information of the inverter's changed state may be delayed up to the duration of the IE, depending on when the inverter output was sampled. Since this delay can continue from one communication node to the next through each parallel instance, it can accumulate at the output at bit 127 where the rollover delay is measured. The sum of accumulated delay can increase as the parallelism increases. This is responsible for the positively correlated relationship in Figure 7.

Increased Resolution

However, if IE resolution increases (from 10 ns to 2 ns) in Figure 8, the percent error of measurement decreases. This is because an inverter fall at communication nodes is sampled every 2 ns, instead of 10 ns. Since the inverter fall time is on the order of 10 ns as these transistors were sized, a 2 ns sample results in smaller worst-case delay in observing a rail-to-rail state change on an inverter output. Percent error of measurement drops to below five percent for the 64x and 128x parallel cases in Figure 8, and to below one percent for the 2x to 16x parallel cases, although the speedup decreases.

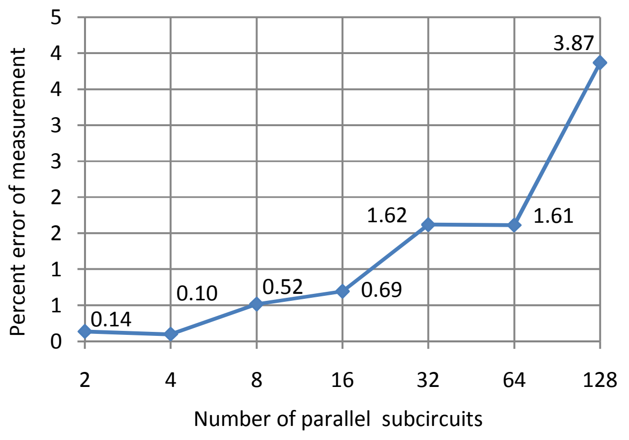

Figure 8. Percent error of measurement at 2 ns IE resolution

The transient analysis time for the same degree of parallelism increases as resolution increases, shown in Figure 9, due to the increased communication rate with the SimConnect server (higher resolution results in smaller Ngspice time steps, resulting in more IE tokens through the KPN FIFOs).

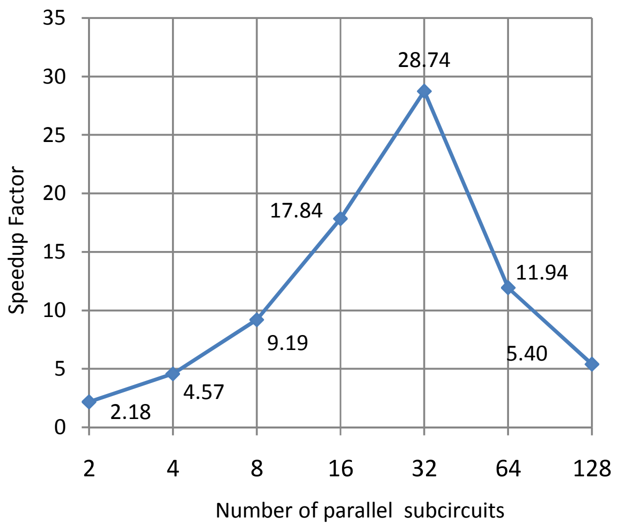

Figure 9. Speedup at 2 ns IE resolution

VI. Discussion

Significant transient analysis time decrease (17x at less than 1 percent error) may be achieved by partitioning the 128-bit counter into subcircuits each 128/<n> bits wide, where <n> is a divisor of 128 and the desired factor of parallelism. Maximum speedup occurs at a parallelization factor per IE resolution, beyond which, as parallelization increases, speedup decreases due to increased communication and load on the SimConnect backplane.

This cost of communication also occurs as IE resolution increases. However, percent error of measurement can be reduced arbitrarily per degree of parallelization by increasing the IE resolution, as shown in Figure 8. This speedup in transient analysis time for the same 128-bit counter is achieved without modifications to the Ngspice kernel, or the execution host, making it different than execution-level parallelization schemes. In this method, the partitioning is performed at the circuit expression-level, in multiple Ngspice decks spread over independent simulators, so both the model-evaluation and the matrix-solving phases occur in parallel.

Choosing an appropriate degree of parallelization and IE resolution automatically is not yet suggested by this work, since it is highly circuit dependent (the partitioning should look for nodes of loose coupling or signal feed-forward cutsets). For accuracy, IE resolution should be on the order of the maximum frequency content of the communicated signal to minimize accumulated delay of rise or fall time information due to sampling. In the examples of Figures 7 and 8, decreasing the IE duration to one fifth (2 ns) of the circuit inverter rail-to-rail fall time (approximately 10 ns) decreased the percent error of measurement for each degree of parallelism by more than one-half for the 4x through 128x parallel cases.

With this approach, there will always be tradeoffs between degree of parallelization, total circuit analysis time, IE resolution, and percent error of measurement. However, gains up to 52x transient analysis time at less than ten percent error by this software technique alone, without modifying the simulator or execution host, may be acceptable at some early investigation phases of system-level design (SLD) [20].

VII. Summary and Conclusions

We took the SimConnect/SimTalk KPN and IE distributed simulation scheme [8] and applied it to the expression-level parallel execution of an Ngspice circuit at the transistor level. We observed gains up to 52x in analysis time, at less than ten percent error of measurement, and 17x in analysis time, at less than one percent error of measurement. This was achieved without any modification to the Ngspice kernel or execution host, but by partitioning the circuit at the expression level and distributing the coordinated subcircuits over independent, concurrent instances of Ngspice. Coordination was achieved through IE tokens exchanged with the SimConnect server through SimTalk, implementing the dynamics of a Kahn Process Network. In maximum parallelization, up to 128 individual Ngspice instances were coordinated with the SimConnect server.

We postulate that it may be possible to combine expression-level parallelization and execution-level parallelization for further speedup. For example, if at execution-level, a K-times speedup is achieved, then that same speedup would be achieved individually over <N> separate Ngspice instances, since the speedup is internal to each instance. If at expression-level, though, a J-times speedup is achieved, then combining both, a J times K factor of speedup should be achieved for both techniques (the speedups should multiply, not add). One speedup occurs at the execution-level, another at the expression-level. There will still be an error in measurement due to the usage of IEs with this method at the expression-level (compared to execution-level methods that may or may not introduce error). We also intend to apply this technique to simulators not initially written for parallel internal execution, such that they offer a device-level communication interface, to see if similar speedup gains can be achieved.

References

[1] Schmerler, S., Tanurhan, Y., and K.D. Muller-Glaser, "A Backplane Approach for Cosimulation in High-level System Specification Environments," *Proceedings of the EURO-DAC Design Automation Conference*, pp.262-267, 18-22 Sep 1995.

[2] Zeng, Y., Wentong C., and S.J. Turner, "Causal Order Based Time Warp: A Tradeoff of Optimism," *Proceedings of the 2003 Winter Simulation Conference* , vol.1, no., pp. 855- 863 Vol.1, 7-10 Dec. 2003.

[3] Lungeanu, D., and C.J.R. Shi, "Distributed Event-Driven Simulation of VHDL-SPICE Mixed-Signal Circuits," *Proceedings of the 2001 International Conference on Computer Design*, pp.302-307, 2001.

[4] Gheorghe, L., Bouchhima, F., Nicolescu, G., and H. Boucheneb, "Formal Definitions of Simulation Interfaces in a Continuous/Discrete Cosimulation Tool," *Seventeenth IEEE International Workshop on Rapid System Prototyping*, pp.186-192, 14-16 June 2006.

[5] Gulati, K., Croix, J.F., Khatri, S.P., and R. Shastry, "Fast Circuit Simulation on Graphics Processing Units," *Asia and South Pacific Design Automation Conference*, pp.403-408, 19-22 Jan. 2009.

[6] Kapre, N., and A. DeHon, "Accelerating SPICE Model-Evaluation using FPGAs, *17th IEEE Symposium on Field Programmable Custom Computing Machines*, pp.37-44, 5-7 April 2009.

[7] Nenzi, P., and V. Holger, "Ngspice Users Manual." V. 22, Sep 2010, ngspice.sourceforge.net.

[8] Pfeifer, D. and J. Valvano, "Kahn Process Networks Applied to Distributed Heterogeneous HW/SW Cosimulation." *The 2011 Electronic System Level Synthesis Conference, ECSI*. 5-6 June 2011.

[9] Cox, F.L., W.B. Kuhn, H.W. Li, J.P. Murray, S.D. Tynor, and M.J. Willis, "Xspice User's Manual." Computer Science and Information Technology Laboratory, Georgia Tech Research Institute. Atlanta, December 1992.

[10] W. Dong, P. Li, and X. Ye, "WavePipe: Parallel Transient Simulation of Analog and Digital Circuits on Multi-core Shared-memory Machines," *45th ACM/IEEE Design Automation Conference*, pp. 238-243, 2008.

[11] T. Weng, R. Perng, and B. Chapman, "OpenMP Implementation of SPICE3 Circuit Simulator," *International Journal of Parallel Programming*, vol. 35, no. 5, pp. 493-505, Oct. 2007.

[12] Lee, P.M., Ito, S., Hashimoto, T., Sato, J., Touma, T., and G. Yokomizo, "A Parallel and Accelerated Circuit Simulator with Precise Accuracy," *Proceedings of ASP-DAC 2002. 7th Asia and South Pacific and the 15th International Conference on VLSI Design*, pp.213-218, 2002.

[13] Peng, H., and C-K. Cheng, "Parallel Transistor Level Circuit Simulation Using Domain Decomposition Methods," *Asia and South Pacific Design Automation Conference*, pp.397-402, 19-22 Jan. 2009.

[14] Hutchinson, S., Keiter, E., Hoekstra, R., Watts, H., Waters, A., Schells, R., and S. Wix, "The Xyce Parallel Electronic Simulator – An Overview," *IEEE International Symposium on Circuits and Systems*, 2001.

[15] Amory, A., Moraes, F., Oliveira, L., Calazans, N., and F. Hessel, "A Heterogeneous and Distributed Cosimulation Environment," *Proceedings of the 15th Symposium on Integrated Circuits and Systems Design*, pp. 115- 120, 2002.

[16] Atef, D., Salem, A., and H. Baraka, "An Architecture of Distributed Cosimulation Backplane," *42nd Midwest Symposium on Circuits and Systems*, vol. 2, pp.855-858, 1999.

[17] Sung, W., and S. Ha, "A Hardware Software Cosimulation Backplane with Automatic Interface Generation," *Proceedings of the ASP-DAC Design Automation Conference*, pp.177-182, 10-13 Feb 1998.

[18] Kahn, G., "The Semantics of a Simple Language for Parallel Programming," *Information Processing*, pages 471-475, Stockholm, Sweden, Aug 1974. North Holland, Amsterdam.

[19] Lee, E.A., and A. Sangiovanni-Vincentelli, "Comparing Models of Computation," *Digest of Technical Papers, the 1996 IEEE/ACM International Conference on Computer-Aided Design*, pp.234-241, 10-14 Nov 1996.

[20] Sangiovanni-Vincentelli, A., "Quo Vadis, SLD? Reasoning about the Trends and Challenges of System Level Design," *Proceedings of the IEEE*, vol.95, no.3, pp.467-506, March 2007.