SAND2019-0744C

Xyce: Open Source Simulation for Large-Scale Circuits

Eric R. Keiter
Electrical Models and Simulation
Sandia National Laboratories
P.O. Box 5800
Albuquerque, New Mexico 87185–1177
Email: erkeite@sandia.gov

Jason C. Verley
Electrical Models and Simulation
Sandia National Laboratories
P.O. Box 5800
Albuquerque, New Mexico 87185–1177
Email: jcverle@sandia.gov

Heidi K. Thornquist
Electrical Models and Simulation
Sandia National Laboratories
P.O. Box 5800
Albuquerque, New Mexico 87185–1177
Email: hkthorn@sandia.gov

***Abstract*—This paper provides an overview of the open source analog simulation tool, Xyce, which was designed from the ground-up to perform large-scale circuit analysis. Current capabilities of the simulation tool will be discussed, including the analysis methods, device models, and parallel implementation.**

***Keywords*—circuit simulation; Computer-aided design; Algorithm Design and Analysis**

I. Introduction

Accurate analog circuit simulation is an important part of design and verification of electrical and electronic circuits and systems. At modern technology nodes, analog, SPICE-accurate simulation can be a prohibitive development bottleneck. Traditional circuit simulation, such as with SPICE [1], does not scale well beyond tens of thousands of unknowns, due to a reliance on direct matrix solver methods.

A recent driver for efficient large-scale analog simulation is the trend towards digitally-assisted analog/RF, and the development of Systems On a Chip (SOCs). Another driver is the challenge of design, verification, and debugging of mixed-signal systems, which is due to the presence of "analog issues," such as variability, noise/interference, nonlinear analog dynamics, analog waveshapes, and timing/phase lags. Often, for simulation of mixed-signal systems, simulation of the analog portions of the system are a significant bottleneck that cannot be avoided, as verification methods often require analog simulation results as an input.

This paper contains an overview of the open source analog circuit simulation tool, Xyce [2], which was designed from the ground-up to perform large-scale parallel circuit analysis, primarily using message-passing. Current capabilities of the simulation tool will be discussed, including the code design philosophy, analysis methods, and device models. A particular emphasis is placed on parallel algorithm strategies, with some recent parallel scaling results.

II. Circuit Simulation Background

Time-domain (transient) and frequency-domain (HB) analog circuit simulations are an essential, yet expensive, part of the computer-aided design (CAD) process. Traditional circuit simulators, such as SPICE [1], are based on solving a fully-coupled nonlinear differential algebraic equation (DAE)

$$f\left(x\left(t\right)\right)+\frac{dq\left(x\left(t\right)\right)}{dt}=b\left(t\right). \tag{1}$$

where $x(t) \in \mathbb{R}^N$ is the vector of circuit unknowns, q and f are functions representing the dynamic and static circuit elements (respectively), and $b(t) \in \mathbb{R}^N$ is the input vector. This set of DAEs is often nonlinear, and is derived from enforcing Kirchoff's current and voltage laws on a network using formulations such as modified nodal analysis (MNA). The numerical approach employed to compute solutions to equation (1) is predicated by the analysis type. Common analysis types for circuit simulation include steady-state, transient, small-signal frequency domain (generally called AC analysis), and large-signal frequency domain methods such as harmonic balance (HB). A circuit simulation solver flow for transient analysis is shown in Fig. 1. Analogous solver flows exist for every method.

For transient analysis, the set of equations (1), more generally expressed as $F(x, x') = 0$, is solved by numerical integration methods corresponding to the nested solver loop in Fig. 1. This requires the solution to a sequence of nonlinear equations, $F(x) = 0$. Typically, Newton's method is used to solve these nonlinear equations, resulting in a sequence of linear systems (represented as $Ax = b$ in the figure) of the form:

$$(G + Q/\delta t)\delta x = -F \tag{2}$$

that involve the conductance, $G(t) = \frac{df}{dx}(x(t))$, and capacitance, $Q(t) = \frac{dq}{dx}(x(t))$, matrices. For DC (steady-state) analysis, the q terms are not present in equation (1), so the linear system only involves the conductance matrix. The matrices and vectors of Eq. 2 are constructed from device model evaluations, the results of which must be assembled by the simulation framework at each nonlinear solution step.

For harmonic balance (HB), both the f and q terms are present, but the solution is approximated in the frequency domain as a Fourier expansion. After substitution and truncation (to M harmonics), the frequency-domain system is:

$$H_{HB} = \Omega Q\left(X\right) + F\left(X\right) - B = 0, \tag{3}$$

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

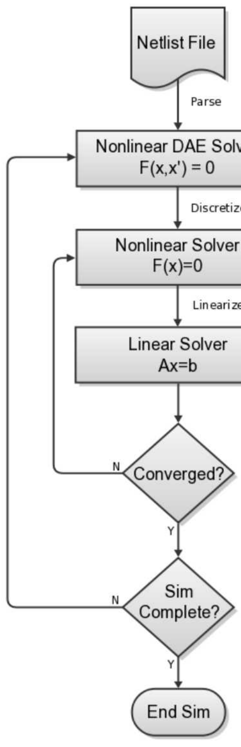

Fig. 1. General circuit simulation flow

where

$$\Omega = \begin{bmatrix} -Mj\omega_0 & & \\ & \ddots & \\ & & Mj\omega_0 \end{bmatrix}, \omega_0 = \frac{2\pi}{T}.$$

Compared to the linear system, of dimension N, generated by transient analysis, HB analysis will generate a linear system of dimension $n(2M+1)$ that is block structured, complex-valued, and possibly dense.

For all types of analysis, device model evaluations and matrix solves comprise the bulk of the simulation cost. As such, any parallel approach should seek to speed up and parallelize these two phases of the calculation. The device evaluation phase of circuit simulation typically scales linearly with problem size. However, matrix solves are another matter. The traditional solution approaches for solving matrices do not scale well beyond tens of thousands of unknowns due to a reliance on a large single matrix that is usually treated by direct matrix solvers [3], [4]. As a result, the runtime scales super-linearly with increasing circuit size. For RF simulation, the can be even worse, because harmonic balance (HB) analysis generates larger block matrices that lack sparsity for nonlinear portions of the problem.

The speed penalty of traditional SPICE-accurate simulation can be mitigated by simulating individual modules. It is also natural to consider techniques that use numerical approximations at various levels throughout the simulator. "Fast-SPICE" simulators [5], [6], [7] rely on circuit-level, hierarchical partitioning algorithms, event-driven simulation techniques, and efficient surrogate models for devices and/or sub-circuits to perform faster, large-scale circuit simulations. While effective in many cases, the numerical approximations inherent to such algorithms can break down for modern feature sizes, especially in post-layout simulations.

The development of inexpensive computer clusters, as well as multi-core technology, resulted in significant interest for efficient parallel circuit simulation [8]. Parallel "true-SPICE" circuit simulation at this point has a rich history, dating back decades [8], [9], [10]. More recent efforts include Frölich [11], who relied on a multi-level Newton approach in the Titan simulator; Basermann [12], who used a Schur-complement based preconditioner; Peng et al. [13] used a domain decomposition approach and relied on a combination of direct and iterative solvers; and Benk et al [14] who also used a domain decomposition approach but with a different partioning method. At this point, parallel "true-SPICE" circuit simulation is also well-integrated into several commercial tools [15], [16].

III. The Xyce Parallel Simulator

Xyce is a SPICE-style analog simulation tool [17] designed to use distributed memory parallelism to address circuit simulation scalability. Additionally, it has been designed with a modular framework, to facilitate integration of new device models, solution algorithms and analysis methods. In this section, the main components and capabilities of Xyce will be described.

A. Device Model Support

Xyce includes legacy SPICE models, industry standard models (BSIM, PSP, MEXTRAM, VBIC, e.g.), and non-traditional models, such as memristors. The extensible design of the device model package has also enabled Xyce to be used to simulate other types of networks, including biological/reaction/neural networks and power grids. To facilitate model implementation, Xyce can be linked to model compilers, including ADMS [18], and the Berkeley Model and Algorithm Prototyping Platform (MAPP) [19], which can translate a model written in a high level modeling language, such as Verilog-A to compilable Xyce C++ code.

B. Analysis Capabilities

Xyce was initially developed to perform transient and DC analysis of analog circuits. The analysis capabilities have been extended to include AC, single-tone and multi-tone harmonic balance (HB), multi-time PDE (MPDE), model order reduction (MOR), direct and adjoint sensitivity analysis, and uncertainty quantification (UQ) such as Polynomial Chaos Expansion (PCE) methods.

C. Code Design

Xyce is written in ANSI C++ using modern software paradigms, which enables Xyce to be a production simulator, as well as a testbed for parallel algorithm research. Xyce uses abstract interfaces and runtime polymorphism throughout the code, which facilitates code reuse and algorithmic flexibility.

Many of the higher-level abstractions, relating to the analysis type or time integration methods, have implementations that are contained in Xyce. However, the lower-level numerical abstractions, related to the nested solver loop in Fig. 1, have interfaces to the high-performance scientific libraries provided by Trilinos [20]. The current Trilinos software stack that is employed by Xyce is illustrated in Fig. 2. Xyce employs these software abstractions to enable adaptation to future parallel paradigms and arithmetic precision strategies with minimal effort. Xyce currently uses MPI with double precision arithmetic through Essential Petra (Epetra). However, future computational platforms may require the use of other parallel paradigms, such as hybrid techniques that combine MPI with threads, to achieve optimal performance. Many of these future computational strategies are discussed in reference [21].

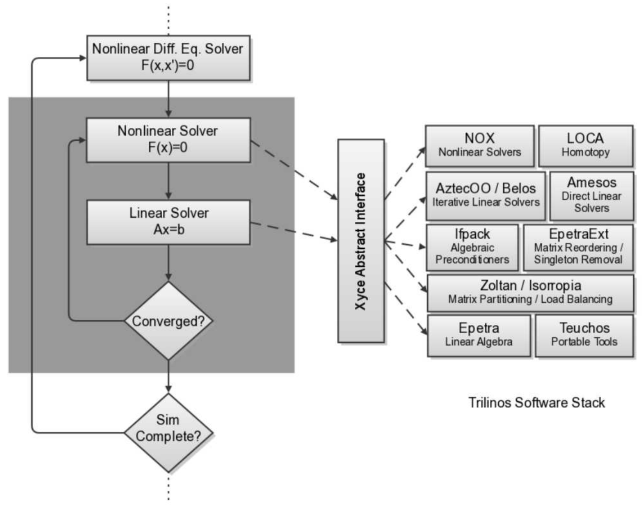

Fig. 2. Nested solver loop interface to Trilinos

D. *Scalable Simulation*

Achieving scalable parallel circuit simulation often comes down to striking the right balance of device distribution and choice of a linear solver. Since the inception of Xyce, the simulator has been designed to use a separate partition for devices and the linear solver, as illustrated in Fig. 3. This is because the cost of evaluating a device model can vary greatly between device types, and balancing that cost across processors can result in a matrix that is challenging for many linear solvers. Furthermore, as the scale of circuits increase—and assuming a reasonable distribution of devices—the dominant cost in the analysis quickly becomes the linear solver. To address this performance bottleneck, new parallel linear solvers and preconditioners [22], [23], [24] have been developed that enable the scalable transient simulation of postlayout ASICs with millions of devices. In the next section, a brief description of each of these methods is given.

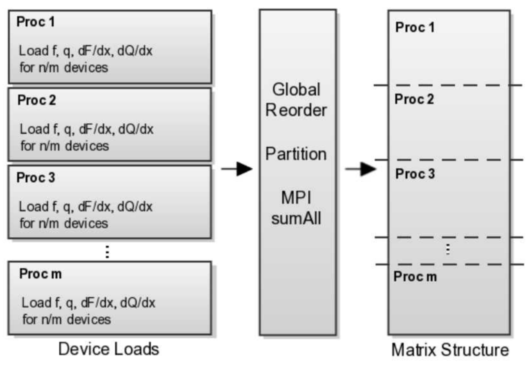
Proc 1
Load f, q, dF/dx, dQ/dx for n/m devices
Proc 2
Load f, q, dF/dx, dQ/dx for n/m devices
Proc 3
Load f, q, dF/dx, dQ/dx for n/m devices
Proc m
Load f, q, dF/dx, dQ/dx for n/m devices
Device Loads
Global Reorder
Partition
MPI sumAll
Proc 1
Proc 2
Proc 3
Proc m
Matrix Structure

Fig. 3. Parallel load balance for device evaluation and matrix structure

IV. Parallel Matrix Solvers

The Xyce simulator has several direct and iterative matrix solvers, ranging from serial direct [3], [4] and parallel direct [24], to parallel preconditioned iterative [22] and even hybrid methods, which combine iterative and direct approaches [23]. Which approach is best for a given circuit will depend upon the size of the problem, as well as the circuit connectivity and respective matrix structure. A high level description of the different matrix solvers in Xyce is given in this section.

A. *Serial and Parallel Direct Solvers*

The default linear solver in Xyce for small-to-medium problems is KLU, a serial direct solver specifically designed for circuit matrices by Davis [4]. More recently, in reference [24], a threaded direct solver, Basker, was presented. This solver relies on hierarchical parallelism and data layouts, and incorporates the first parallel implementation of the Gilbert-Peierls algorithm that is the cornerstone for KLU. In the paper, Basker was compared to KLU, in addition to public-domain parallel direct solvers, such as Pardiso MKL (PMKL). The comparisons were conducted on the Florida sparse matrix collection [25], as well as a large transient Xyce calculation for which existing preconditioned iterative methods had failed. Basker was very competitive on matrices will low fill-in density. On the transient Xyce calculation, a speedup of 5.43x was achieved when using Basker instead of PMKL and 5.22x when using Basker instead of KLU.

B. *Singleton Removal*

A common feature required for the Xyce parallel iterative solvers (described in sections IV-E, IV-D and IV-C) to work effectively is a matrix preprocessing step known as *singleton removal*. Matrices from circuit simulation are sparse, but typically contain dense rows and columns. These structural matrix features are problematic for parallel domain decomposition (DD) methods, as they have the potential to increase communication costs dramatically. The crucial observation [12] is that the dense rows (or columns) correspond to columns (or rows) with one and only one non-zero entry. A histogram illustrating

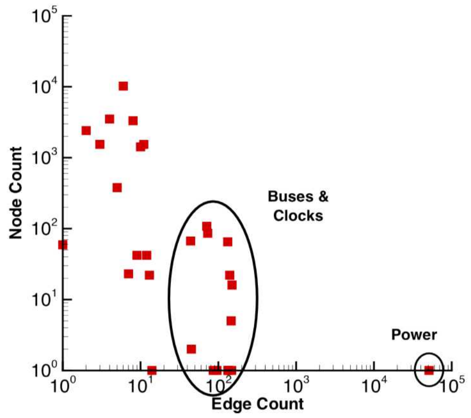

Fig. 4. Example IC connectivity histogram. Highly connected nodes such as power, and moderately connected nodes such as buses and clocks cause communication problems for parallel linear solvers. It is advantageous to eliminate singletons from the matrix graph as an initial step.

the connectivity for an example integrated circuit (IC) is depicted in Fig. 4. For this particular example, the power and ground nodes are connected to more than 10^4 circuit nodes, requiring singleton removal for successful parallel solution.

Singleton removal is a successful strategy, but only when the nodes are connected to ideal sources, and thus can essentially be removed from the system of equations by inspection. In practice, highly connected nodes may be non-ideal due to parasitic elements, and under this circumstance, the singleton removal algorithm can fail. A mitigation strategy designed to preserve singleton removal, based on multi-level Newton methods is presented in reference [17].

C. Block Triangular Factorization Preconditioner

In reference [22], a preconditioner based on block triangular form (BTF) was presented. The block triangular structure is determined by permuting the DC matrix in two steps: first a maximum matching permutation to generate a matrix with a zero-free diagonal, and second a topological sort which finds the strongly-connected components of the associated directed graph. The permutation to BTF is also known as the Dulmage-Mendelsohn decomposition. Once the matrix has been permuted, the block structure is partitioned using a hypergraph partitioner. This preconditioner works well whenever the irreducible blocks (strongly connected components) are small, e.g., for unidirectional circuits; but this property is not satisfied for all circuits, especially ones based on modern process nodes.

D. Hybrid Schur Complement Preconditioner

In reference [23], a solver/preconditioner known as ShyLU was presented. ShyLU is a hybrid solver, which makes use of both course-grained (message-passing) and fine-grained (threading) parallelism. It is also a hybrid solver in the mathematical sense, in that it uses features from direct and iterative methods. ShyLU uses Schur complement techniques to bridge the gap between direct and iterative methods resulting in a solver that is scalable and robust for large-scale circuits. An example of the wide separator ordering used by ShyLU is shown in figure 5. The Schur complement approach used by ShyLU is similar to approaches used by other parallel circuit simulators [12], [26], [14]. In some cases, it is used as the solver directly, and in others it is used as part of a preconditioning strategy. One scaling result from reference [23], for which ShyLU was used as a preconditioner with Xyce is shown in figure 6. The circuit used in this study is a CMOS ASIC consisting of 1.6 million total devices and roughly 2 million unknowns. The test machine was a capacity cluster, with 272 compute nodes, where each node has a 2.2 GHz AMD quad socket/quad core processor and 32GB RAM. When run in serial with the KLU solver this circuit took nearly 2 weeks to complete the simulation. Using optimal partitions with ShyLU the same simulation took about a day.

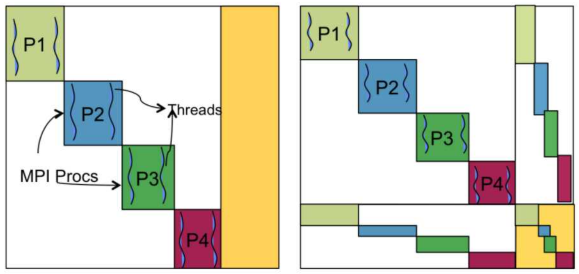

Fig. 5. Graph/Hypergraph based unsymmetric (left) and symmetric (right) ordering of the sparse linear system for parallelism in ShyLU

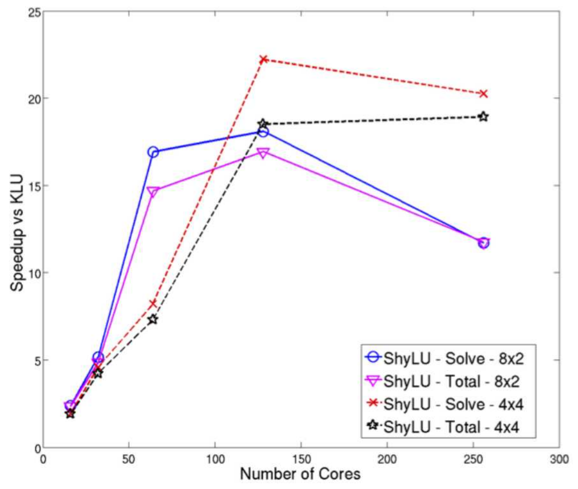

Fig. 6. Speedup of Xyce's simulation time and linear solve time for strong scaling experiments with different configurations of MPI Tasks X Threads per node using ShyLU.

E. Other Preconditioners

In addition to the circuit-specific preconditioners described in sections IV-D and IV-C, Xyce also has available to it, through the Trilinos framework, various standard preconditioning methods. These include incomplete LU preconditioners and block Jacobi preconditioners. Despite being more generalpurpose, when combined with the singleton removal preprocessing they can still be effective for some circuits.

V. Recent Parallel Scaling Examples

In this section, results of a recent parallel scaling study are presented. Two memory circuits of different sizes are considered. The first one is an asynchronus memory circuit consisting of 100K CMOS transistors and the second is a similar memory circuit, but with 1 million CMOS transistors. The circuits are described in pre-layout netlists so no parasitic elements are included. The computer used for these tests is a large institutional cluster at Sandia. This system has Intel Xeon E5-2670 8C 2.6GHz processors, 16 cores on each node, and Infiniband QDR interconnect.

For the smaller 100K device circuit, the most competitive choice for parallelism was to perform the device evaluations in parallel, and the linear solve in serial with KLU. As noted in section III-D, the device evaluation and matrix solve phases have independent parallel partitioning, and one option under this design is to only perform one of the two phases in parallel while leaving the other one in serial. In comparison to iterative solvers, direct solvers are very robust and will usually be faster for smaller matrices, and for the 100K problem, this was the case. With this parallel arrangement, only one of the two major components will speed up the calculation with additional processors, which of course limits the potential overall speedup.

The results for the 100K device circuit using this partial parallel approach are given in Fig. 7 and Fig. 8. The calculations are performed 1,4,8 and 16 cores. Fig. 7 shows a strong scaling result for all the main phases of the simulation, and Fig. 8 shows runtime comparisons for the same phases. Ideal strong scaling on this figure would result in a perfectly straight diagonal line from the lower left had corner of Fig. 7 to the upper right hand corner. The red line represents the cost of the residual load, and the blue line represents the Jacobian load. Combined, these two lines represent the device evaluation phase, with most of the work (all the actual floating point calculations) in the residual evaluation. This phase exhibits good parallel scaling, without much rolloff even at 16 cores. The linear solve, being serial, doesn't speed up in parallel at all and is thus a bottleneck in this calculation. The total simulation, represented by the cyan line shows a speedup of about 2x on 4,8 and 16 cores. This suggests, based on Amdahl's Law that the two phases of the problem are roughly comparable in cost, and this is supported by the serial runtime numbers in Fig. 8.

For the larger 1M device circuit, results are shown in Fig. 9 and Fig. 10. This circuit is large enough that a direct solver was not competitive. The best choice of matrix solver turned out to be GMRES with a block Jacobi preconditioner. Tests were performed for 16, 32, 64, 96 and 128 cores. As the computer used for this study has 16 cores per node, this calculation (unlike the 100K case) made use of the interconnect. The strong scaling result, shown in Fig. 9 exhibits good speedup in all phases of the problem.

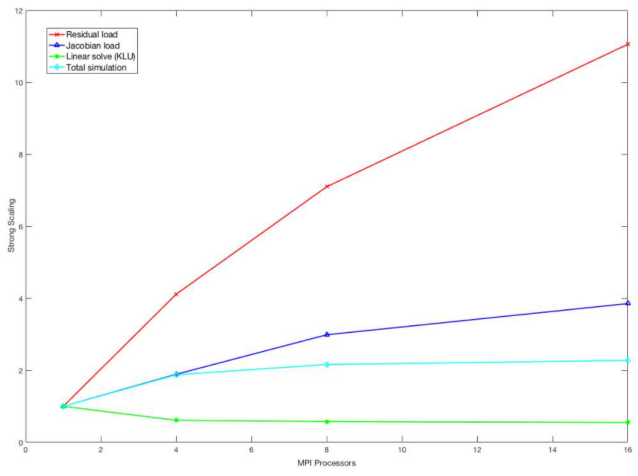

Fig. 7. 100K memory circuit strong scaling

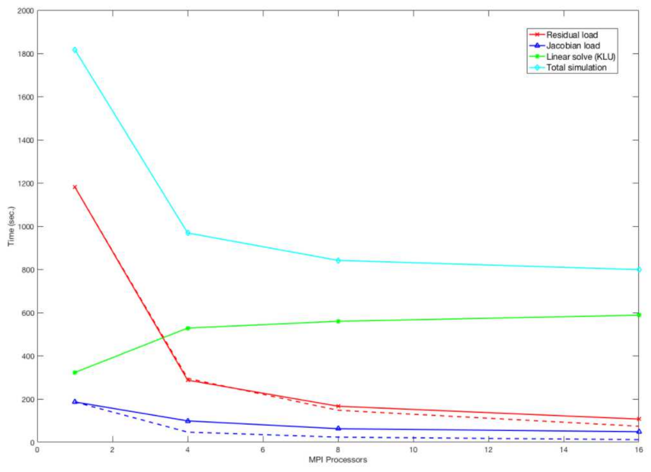

Fig. 8. 100K memory circuit runtime comparison

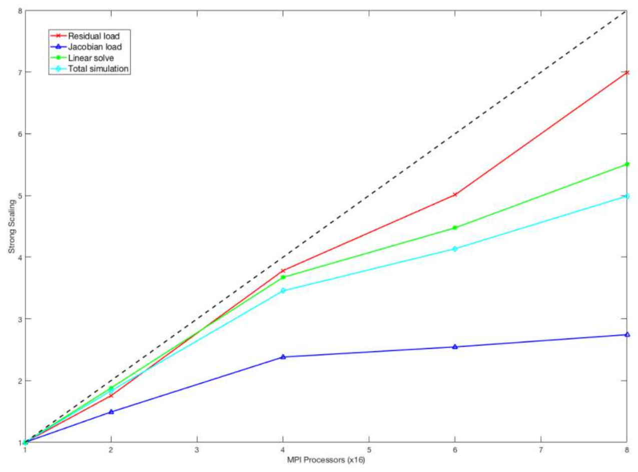

Fig. 9. 1M memory circuit strong scaling

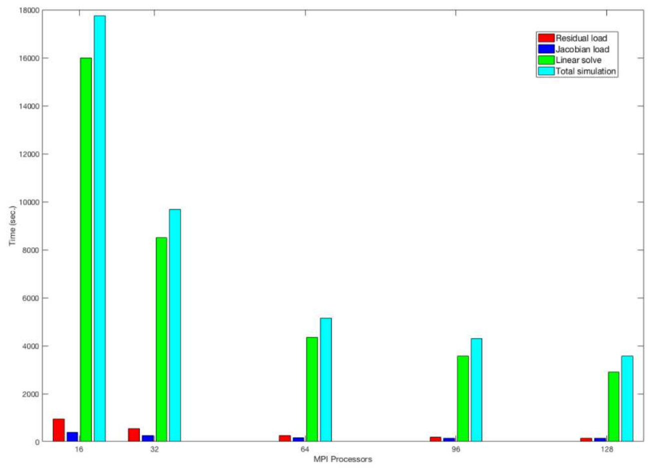

Fig. 10. 1M memory circuit runtime comparison

VI. Conclusion

In this paper, an overview of the Xyce parallel circuit simulator has been presented. With message-passing as the main parallel programming approach, Xyce has a variety of parallel algorithm strategies. These were discussed, with a particular emphasis on the parallel matrix solvers. Finally, some recent parallel scaling results were presented, which used simulations from large memory circuits.

VII. Licensing and Availability

Xyce is open source software, released under the GNU General Public License, Version 3.0, since the release of Xyce 6.0 in 2013. More information about the Xyce project, including software downloads and documentation, can be found on the website: https://xyce.sandia.gov.

Acknowledgment

We wish to acknowledge the Advanced Simulation and Computing (ASC) Project under the National Nuclear and Security Administration (NNSA), and the DARPA Posh Open Source Hardware (POSH) project for funding the development of Xyce.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

References

[1] L. W. Nagel, "Spice 2, a computer program to simulate semiconductor circuits," Tech. Rep. Memorandum ERL-M250, 1975.

[2] *Xyce Parallel Circuit Simulator*, https://xyce.sandia.gov.

[3] K. Kundert, "Sparse matrix techniques," in *Circuit Analysis, Simulation and Design*, A. Ruehli, Ed. North - Holland, 1987.

[4] T. A. Davis, *Direct Methods for Sparse Linear System*. SIAM, 2006.

[5] M. Rewieński, *A Perspective on Fast-SPICE Simulation Technology*. Dordrecht: Springer Netherlands, 2011, pp. 23–42. [Online]. Available: https://doi.org/10.1007/978-94-007-0149-6_2

[6] *Synopsys CustomSim*, https://www.synopsys.com/verification/ams-verification/customsim.html.

[7] *Cadence Spectre XPS*, https://www.cadence.com/content/cadence-www/global/en_US/home/tools/custom-ic-analog-rf-design/circuit-simulation/spectre-extensive-partitioning-simulator-xps.html.

[8] P. Li, "Parallel circuit simulation: A historical perspective and recent developments," *Foundations and Trends® in Electronic Design Automation*, vol. 5, no. 4, pp. 211–318, 2012. [Online]. Available: http://dx.doi.org/10.1561/1000000020

[9] P. F. Cox, R. G. Burch, D. E. Hocevar, P. Yang, and B. D. Epler, "Direct circuit simulation algorithms for parallel processing (VLSI)," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 10, no. 6, pp. 714–725, June 1991.

[10] M.-C. Chang and I. N. Hajj, "iPRIDE: a parallel integrated circuit simulator using direct method," in *[1988] IEEE International Conference on Computer-Aided Design (ICCAD-89) Digest of Technical Papers*, Nov 1988, pp. 304–307.

[11] N. Fröhlich, B. Riess, U. Wever, and Q. Zheng, "A new approach for parallel simulation of VLSI-circuits on a transistor level," *IEEE Transactions on Circuits and Systems Part I*, vol. 45, no. 6, pp. 601–613, 1998.

[12] A. Basermann, U. Jaekel, and M. Nordhausen, "Parallel iterative solvers for sparse linear systems in circuit simulation," *Fut. Gen.. Comput. Sys.*, vol. 21, no. 8, pp. 1275–1284, 2005.

[13] H. Peng and C.-K. Cheng, "Parallel transistor level circuit simulation using domain decomposition methods," in *Proceedings of the 2009 Conference on Asia and South Pacific Design Automation*. IEEE Press, 2009, pp. 397–402.

[14] J. Benk, G. Denk, and K. Waldherr, "A holistic fast and parallel approach for accurate transient simulations of analog circuits," *Journal of Mathematics in Industry*, vol. 7, no. 1, p. 12, Dec 2017. [Online]. Available: https://doi.org/10.1186/s13362-017-0042-z

[15] *ProPlus Solutions NanoSpice^{TM}*, https://www.proplussolutions.com/nanospice-en.php.

[16] *Silvaco SmartSpice*, https://www.silvaco.com/products/analog_mixed_signal/smartspice.html.

[17] E. R. Keiter, H. K. Thornquist, R. J. Hoekstra, T. V. Russo, R. L. Schiek, and E. L. Rankin, "Parallel transistor-level circuit simulation," in *Advanced Simulation and Verification of Electronic and Biological Systems*, P. Li, L. M. Silveira, and P. Feldmann, Eds. Springer, 2011, ch. 1, pp. 1–21.

[18] L. Lemaitre, C. McAndrew, and S. Hamm, "ADMS — automatic device model synthesizer," *IEEE Custom Integrated Circuits Conference*, May 2002.

[19] T. Wang, K. Aadithya, B. Wu, and J. Roychowdhury, "Mapp: A platform for prototyping algorithms and models quickly and easily," in *2015 IEEE MTT-S International Conference on Numerical Electromagnetic and Multiphysics Modeling and Optimization (NEMO)*, 2015, pp. 1–3.

[20] *The Trilinos Project*, https://trilinos.org.

[21] C. Baker, E. Boman, M. Heroux, E. Keiter, S. Rajamanickam, R. Schiek, , and H. Thornquist, "Enabling next-generation parallel circuit simulation with trilinos," in *Workshop on High-Performance Scientific Software (HPSS2011)*, Aug. 2011.

[22] H. K. Thornquist, E. R. Keiter, R. J. Hoekstra, D. M. Day, and E. G. Boman, "A parallel preconditioning strategy for efficient transistor-level circuit simulation," in *2009 IEEE/ACM International Conference on Computer-Aided Design - Digest of Technical Papers*, 2009, pp. 410–417.

[23] H. K. Thornquist and S. Rajamanickam, "A hybrid approach for parallel transistor-level full chip circuit simulation," in *High Performance Computing for Computational Science - VECPAR 2014*, M. Dayde, O. Marques, and K. Nakajima, Eds. Springer, 2014.

[24] J. Booth, S. Rajamanickam, and H. Thornquist, "Basker: A threaded sparse lu factorization utilizing hierarchical parallelism and data layouts," in *Proceedings of the 2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. ACM, 2016, pp. 673–682.

[25] T. A. Davis and Y. Hu, "The university of florida sparse matrix collection," *ACM Trans. Math. Softw.*, vol. 38, no. 1, pp. 1:1–1:25, Dec. 2011. [Online]. Available: http://doi.acm.org/10.1145/2049662.2049663

[26] C. Bomhof and H. vanderVorst, "A parallel linear system solver for circuit simulation problems," *Num. Lin. Alg. Appl.*, vol. 7, no. 7-8, pp. 649–665, 2000.