A Perspective on Fast-SPICE Simulation Technology

Michał Rewieński

Abstract This chapter presents an introduction to the area of accelerated transistor-level ('fast-SPICE') simulation for automated verification and characterization of integrated circuits (ICs) from technologist's perspective. It starts with outlining goals, expectations and typical usage models for fast-SPICE simulators, stressing how they differ from regular SPICE tools. It continues with presenting and classifying core technologies typically included in fast-SPICE simulators, which allow them to achieve critical performance and capacity gains. Also, it discusses how different approaches toward the computational problem can be combined to design and implement highly effective and efficient simulation acceleration technologies, including advanced circuit partitioning, and schemes for optimized modeling of post-layout memory circuits and large parasitic networks. Finally, challenges facing fast-SPICE, as well as possible future research areas are briefly outlined.

1 Introduction

Transistor-level electrical simulation of integrated circuits, i.e. SPICE simulation, is a dynamic area of academic and industrial research. Throughout the years the accompanied software development provided electrical engineers with vital tools and methodologies allowing them to characterize and verify new designs prior to their fabrication in silicon. SPICE simulation quickly became an inherent component of the integrated circuit (IC) design process, crucial to successful and cost-efficient production of electronic chips. Naturally, SPICE tools became part of the race to develop increasingly more complex circuits, and were required to match their simulation capabilities with rapidly advancing transistor manufacturing technology (and conversely, it became clear that capacity of simulation software puts a limit on com-

Michał Rewieński
Synopsys Inc., 700 E. Middlefield Rd., Mountain View, CA, USA
e-mail: michalr@synopsys.com

plexity of ICs which can be efficiently produced). Further integration and downscaling of electronic circuits created a need to simulate larger design blocks, as well as entire chips. This meant scaling up simulation capabilities from circuits with thousands of transistors to circuits with millions of transistors, which, as quickly realized, required abandoning the uncompromising approach to simulation accuracy of the initial SPICE tools. Hence, a new class of *accelerated* transistor-level simulators (also called 'fast-SPICE' simulators) emerged. 'Fast-SPICE' tools started to explore various accuracy tradeoffs to achieve incredible performance and capacity gains. Those in turn were made possible by a myriad of novel heuristics, methodologies, and computational techniques.

Discussing fast-SPICE technologies is the focus of this chapter. Yet, presenting a comprehensive summary or even a listing of all the simulation acceleration methods, published in hundreds of papers and patents, lies well beyond the scope of this contribution. Instead, this chapter attempts to take only a perspective on those technologies. The perspective will be that of a fast-SPICE researcher and developer at the same time (which is the author's own), and will allow one to see the inherently multidisciplinary character of fast-SPICE technologies, underline some of the common characteristics of certain successful techniques, and at the same time illustrate general shifts of focus of fast-SPICE simulation over the years.

First, the scene is set up by briefly presenting the computational problem shared by SPICE and fast-SPICE, and the initial approach toward solving it, exemplified by virtually all classical SPICE simulators. Then, typical usage and limitations of SPICE simulators are outlined, followed by motivations behind accelerated, fast-SPICE tools, and presentation of main differences between the two classes of tools. Next, fast-SPICE technologies are discussed with emphasis on their various, often complementary views of the computational problem, and their classification is attempted. A few selected techniques are then described in more detail in order to illustrate how they integrate various approaches and views of the computational problem in order to achieve gains in simulation capacity and performance. Finally, challenges and potential research paths in fast-SPICE simulation are listed.

2 SPICE: Transistor-Level Circuit Simulation

In SPICE simulation transistor models (such as e.g. BSIM4 or Mos11), lumped passive element models (e.g. R, L, C), and voltage and current source models (e.g. using modified nodal analysis (MNA) formulation) are combined with circuit equations (Kirchhoff's current and voltage laws) to describe the physics and topology of the considered integrated circuit design. In mathematical terms, the problem to be solved numerically is a system of differential-algebraic equations (DAEs) in the following form:

$$\frac{dq(\mathbf{v},t)}{dt} = i(\mathbf{v},\mathbf{u},t) \,, \tag{1}$$

where $\mathbf{v} = \mathbf{v}(t)$ is a vector containing nodal voltages and branch currents (for elements applying MNA formulation), $\mathbf{u} = \mathbf{u}(t)$ is a vector of inputs (realized using e.g. voltage sources or digital vector input elements), t is time, $q()$ is an operator describing charge, and $i()$ is an operator describing current. System (1) modeling an integrated circuit is typically highly nonlinear, stiff and, in the case of most fast-SPICE applications, has an extremely large size. Most commonly one is interested in finding transient time-domain solution of (1) in response to a given input signal, therefore the following description will focus on this case.

In order to tackle problem (1) numerically one needs to address the following issues:

1. A numerical integration scheme, along with suitable corresponding time discretization (time-stepping) needs to be applied. Popular and inexpensive choices for stiff systems include backward-Euler, trapezoidal, and second-order Gear methods. For instance, backward-Euler integration will yield an algebraic equation in the form shown below:

$$\mathbf{v}_{t+\Delta t} = \mathbf{v}_t + \Delta t f(\mathbf{v}_{t+\Delta t}, \mathbf{u}_{t+\Delta t}, t+\Delta t) \tag{2}$$

   where $\mathbf{v}_t$ is a (known) vector of nodal voltages at time t, $\mathbf{v}_{t+\Delta t}$ is an unknown vector of nodal voltages at time $t+\Delta t$, Δt is the time-step, $\mathbf{u}_{t+\Delta t}$ is a vector of inputs at time $t+\Delta t$, and $f()$ is a (nonlinear) operator. In the above formulation, the unknown vector $\mathbf{v}_{t+\Delta t}$ is not given explicitly in (2), and needs to solved for.
2. Algebraic equation (2) needs to be solved in order to find $\mathbf{v}_{t+\Delta t}$. It can be rewritten as a problem of finding zeros of a nonlinear operator:

$$F(\mathbf{v}) = 0 \tag{3}$$

   where $\mathbf{v}$ is the unknown vector of nodal voltages at time $t+\Delta t$, and $F(\mathbf{v}) = \mathbf{v} - \mathbf{v}_t - \Delta t f(\mathbf{v}, \mathbf{u}_{t+\Delta t}, t+\Delta t)$. Zeros of the above equation are most often found using some flavor of the multi-dimensional Newton-Raphson method, yet other choices are available such as e.g. successive chords method [8, 16]. An iteration of the Newton method for (3) is given below:

$$\mathbf{v}^{i+1} = \mathbf{v}^i - J_F^{-1}(\mathbf{v}^i) F(\mathbf{v}^i) \tag{4}$$

   where J_F is the Jacobian matrix for operator F, $\mathbf{v}^i$ is an approximation of $\mathbf{v}$ at i-th iteration, $\mathbf{v}^{i+1}$ is approximation of $\mathbf{v}$ at $(i+1)$-th iteration.
3. Finally, a system of linear equations needs to be solved at each iteration of the Newton process (4):

$$J_F(\mathbf{v}_i)\mathbf{x} = F(\mathbf{v}_i). \tag{5}$$

   To this end any direct or iterative linear solver can be applied. In SPICE domain the 'default,' or reference choice of the linear solver is still a direct solver based on LU factorization with an appropriate pivoting scheme.

The above mentioned components: a numerical integration method with a time-stepping scheme suitable for stiff systems of DAEs, robust nonlinear and linear solvers, along with detailed, rigorous device models (specified in terms of mathematical equations relating physical quantities such as voltages and currents, as well as their derivatives) constitute the core of a SPICE simulation engine. This view of the computational engine heavily influenced design of SPICE simulators which were typically implemented as a combination of interacting yet separate numerical components.

2.1 Usage and Limitations of SPICE

The mathematical and numerical framework of SPICE, very briefly outlined in the previous paragraph, has been set up with one main goal in mind: to provide means for high precision electrical modeling of integrated circuits. SPICE tools based on this framework allowed engineers to simulate their design with the highest achievable accuracy. Therefore, SPICE tools naturally dominated the following simulation/modeling areas:

- Characterization of cell libraries and Intellectual Property (IP) circuit blocks;
- Modeling of precision analog components;
- Verification of subsystems along critical paths.

Since achieving highest possible accuracy of the results was a priority (aligned with expectations of SPICE users), a number of restrictions were imposed on the simulation engine. Above all those included: no approximations to device models, uniform time discretization across the entire simulated circuit, and global convergence to the solution at every time-point. This in turn meant constructing and solving a single system of equations representing the entire circuit at every time-point. This synchronous solve often implied taking very small steps even for very latent circuit blocks. The need to solve a single large and stiff linear system, typically meant that direct linear solvers were the only class of methods which could provide reliable performance. (Finding robust preconditioners for iterative methods often proved to be extremely challenging, hence drastically limiting usability for this class of solvers.)

Yet, using direct (sparse) solvers also meant that the cost of solving the linear system was reaching $O(N^{1.5})$ or even $O(N^2)$ for some circuit topologies, where N was the number of nodes in a circuit. Apart from the cost of evaluating complex, equation-based device models (e.g. for MOSFETs), the cost of linear solves remains a critical factor limiting capacity of SPICE simulators. Although this capacity limits for full-accuracy SPICE tools is constantly pushed upwards, its current rough estimate is a circuit with about 100,000 active elements (MOSFETs).

2.2 Need for Accelerated SPICE (‘Fast-SPICE’) Simulation

Higher levels of integration, and shrinking IC manufacturing process technology created a need to simulate and verify at transistor level much larger circuit blocks, well beyond capacity limits of conventional SPICE simulators. In particular two important simulation areas emerged:

- Verification of large-scale digital, memory circuits;
- Mixed-signal, system-on-a-chip (SoC), full-chip functional simulation.

In order to be able to perform such simulations, involving designs with active element count reaching from 100 million to 1 billion MOSFET devices, and millions of parasitic devices, the capacity of SPICE tools had to increase dramatically. For problems of this size the $O(N^{1.5})$ complexity of linear solves became clearly unacceptable, creating a need to modify or abandon some of the paradigms followed by conventional SPICE simulators, such as e.g. single-matrix solve for global solution convergence. At the same time, accuracy requirements for full-chip functional simulations or memory verification runs were clearly not as stringent as for e.g. numerical characterization of cell libraries, allowing for 5–15% error as compared to full SPICE accuracy. IC designers were willing to pay a price for the ability to simulate much larger systems. This created an excellent opportunity to explore various tradeoffs between achievable accuracy, and simulator speed and capacity. Hence, both the need for transistor-level simulations for much larger systems, and more moderate expectations on accuracy for large-scale simulations drove development of a new class of tools: the accelerated transistor-level (‘fast-SPICE’) simulators.

3 Fast-SPICE Technologies

Relaxed accuracy constraints on large-scale simulations enabled development of simulation optimization technologies which by large and far surpassed optimizations considered in traditional SPICE tools, yielding tremendous performance gains reaching 1,000- or 10,000-fold speedups, as well as similar capacity improvements. Hence, it became feasible to perform simulations for circuits with hundreds of millions of devices in a matter of minutes—as illustrated in Table 1, which shows sample performance data for a commercial fast-SPICE tool.

3.1 SPICE vs. Fast-SPICE Techniques

Key fast-SPICE technologies developed to achieve such performance gains differed fundamentally from optimizations found in conventional SPICE engines. In SPICE tools, the only simulation optimizations permitted were those which did not cause any significant loss of accuracy (cf. Fig. 1). Furthermore, due to the assumption that

Table 1 Performance of a commercial fast-SPICE simulator for a sample circuit containing 284 million active MOSFET devices, 290 million passive RC elements, represented using a hierarchical netlist. The test was performed on a workstation with 2.8GHz AMD Opteron processor, using 64-bit precision

	CPU time	Memory cost
Netlist parsing	0 s	18 MB
Identification, optimization	215 s	853 MB
Operating point computation	70 s	395 MB
Transient simulation	674 s	538 MB
Total	959 s	1804 MB

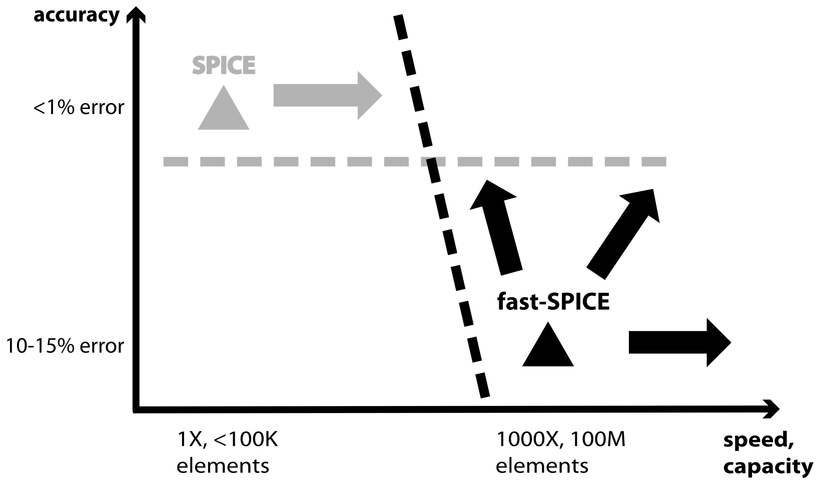

Fig. 1 Triangles show typical 'operating points' for SPICE and fast-SPICE in terms of their capacity, performance and accuracy of computed results. The only allowed SPICE optimizations (indicated with a gray arrow) are those which increase its capacity and speed without sacrificing its high accuracy. On the contrary, in fast-SPICE simulation optimization technologies are allowed to explore various speed/capacity vs. accuracy tradeoffs (cf. black arrows), providing the user with flexibility to select target accuracy or degree of simulation acceleration

SPICE should be able to accurately simulate *any* design, including any novel circuit topology, or unconventional design etc., relatively little effort has been dedicated toward identifying *typical* functional circuit components or blocks, which could potentially be simulated more efficiently. Consequently, the starting point for efforts to improve SPICE was typically a rather narrow view of the computational problem as predominantly a system of equations to be solved. This in turn implied that improvements focused on optimizing components of SPICE engine's mathematical machinery. Enhancements were most often made separately for time-integration schemes, nonlinear solvers, and linear solvers. While over time significant progress has been achieved, yielding gains both in speed and capacity of SPICE tools, the process of improving SPICE in such a manner proved to be very challenging and relatively slow.

During development of fast-SPICE technologies a very different approach toward optimizing the computational problem has been taken. First of all, as already mentioned, those optimizations were allowed to trade off simulation accuracy for speed and capacity, and also permitted direct or indirect control over the desired target level of accuracy (cf. Fig. 1). A no less important consideration was that fast-SPICE tools were to be predominantly used for verification of large systems composed of relatively conventional and robust functional components. This implied that simulation could be aggressively optimized for many well understood circuit blocks. This new approach toward optimizing the simulation dramatically broadened the viewpoint at the computational problem at hand. It was no longer just a system of nonlinear differential equations which was to be optimized, but it became a system of equations describing often well identified functional blocks with known behavior and physical properties. Consequently, instead of narrowly focused SPICE optimizations, fast-SPICE tools contained technologies which took a more *holistic* approach towards optimizing the simulation problem, which was often key to their success. As it will be illustrated in the following sections, fast-SPICE techniques applied, sometimes simultaneously, very different viewpoints in order to achieve tremendous performance and capacity gains.

As a starting point for this discussion let us briefly mention a few 'signature' fast-SPICE optimizations. Then we will go deeper into explaining some general approaches which stimulated their development. The key fast-SPICE technologies are listed in Table 2, which also compares them to solutions used in conventional SPICE. Firstly, accurate, equation-based models for MOSFETs are approximated in fast-SPICE by simplified equations or, most notably, tabularized models for device current, capacitance and charge, yielding 1,000-fold speedups. For passive elements model order reduction (MOR) techniques are applied in order to generate compact models for interconnects. Fast-SPICE also abandoned the requirement for global convergence at each time-point, and global uniform time-stepping for the entire circuit. Instead, 'event-driven' simulation is performed, where solution in a given region is computed only if a change of state is anticipated due to a changing input or load for this region. Also, instead of using global uniform time-step for the entire circuit, fast-SPICE performs 'multi-rate' simulation, where depending on the rate of nodal voltage change in different circuit regions, as well as other criteria, different time-steps are used during numeric integration. Last, but not least, fast-SPICE no longer solves a single system of equations, with a corresponding single, large ma-

Table 2 SPICE vs. fast-SPICE: key optimizations

SPICE	Fast-SPICE
Accurate device models	Approximate MOSFET models, reduction of parasitic networks
Global convergence at each time-point	Event-driven simulation, no global convergence
Global time-step	Multi-rate simulation
Single matrix for entire circuit	Partitioned matrix, hierarchical simulation

trix used during linear solves. Instead, the problem is partitioned into a number of blocks, using carefully designed physical and/or mathematical criteria. Then, linear systems (with corresponding matrices) are solved separately for each block. The final solution is found by performing e.g. some form of relaxation across different blocks, if needed.

All the techniques mentioned above cause systematic accuracy loss, and are therefore specific to fast-SPICE. Apart from them, there were a number of important optimizations initially developed in fast-SPICE domain, which provide performance gains without sacrificing simulation accuracy. Those include:

- High-capacity, high-performance input data storage and processing algorithms for simulation setup phase (i.e. simulator 'front-end'), e.g. efficient methods for processing highly hierarchical netlist representation of the simulated circuit.
- High performance linear system solve technologies, including efficient preconditioners and iterative algorithms, as well as parallel and multi-threaded solvers.
- Automatic, on-the-fly code generation and compilation for faster execution of key simulation blocks.

Although the above mentioned techniques are not fast-SPICE specific, many of them emerged during fast-SPICE development, due to a pressing need for efficient processing of very large input data sets such as hierarchical and non-hierarchical input netlists for full-chip simulations, as well as corresponding back-annotation files, generated by RC extraction tools for post-layout simulation. Many of these software and algorithmic techniques were later transferred to conventional SPICE simulators to increase their capacity and front-end performance.

3.2 Acceleration Technologies: Different Viewpoints

In order to gain more insight on how successful technologies for accelerating transistor-level simulation are designed one should take a look at the general methodology applied during fast-SPICE development, as well as different viewpoints at the computational problem which provide starting points for algorithmic innovations.

3.2.1 Fast-SPICE as Knowledge Database

First of all, fast-SPICE simulators are designed to be much more than conglomerates of various numerical techniques for solving large systems of stiff DAEs. Rather, they are developed as knowledge databases, in which expertise on design, structure, and functionality for many existing IC designs is heavily exploited in order to accelerate simulation while controlling accuracy of the results. Fast-SPICE tools typically include methods which automatically or semi-automatically identify functional circuit blocks of different complexities in order to split the computational problem and to use appropriate solving and modeling techniques and approaches for each block. Also, specialized techniques (some of which will be described in the next section) are implemented in fast-SPICE for certain large-scale functional blocks with well-known functionality and characteristics (such as e.g. power supply networks or memory circuits) in order to achieve cutting-edge simulation speed.

While examining fast-SPICE algorithms, one may distinguish a few complementary viewpoints at the computational problem. Those viewpoints or approaches provide different starting points for simulation acceleration technologies and may be classified as:

- Circuit-based: In this approach knowledge of electrical properties, and functional characteristics for circuit subsystems allows one to develop appropriate optimization and simulation strategy for each block.
- Graph-based: Treating the simulated circuit as a graph allows one to apply high-performance data representation and processing algorithms, as well as efficient identification of blocks with specific topologies.
- Matrix-based: Since, after all, a system of equation is solved, this viewpoint allows one to focus on optimizing matrix representation of the (linearized) problem in order to obtain a system of equations which is easier and faster to solve.

Next, we will mention various techniques which use each of the viewpoints listed above.

3.2.2 Matrix-Based Viewpoint

The ‘matrix-based’ approach has provided a starting point for many methods for accelerating transistor-level simulations. One group of such techniques, analyzes fill-in patterns for matrices representing linearizations of the system describing the considered circuit. Those techniques include:

- Node ordering schemes applied during matrix factorization and linear solves, e.g. minimum degree algorithm, nested dissection (cf. hMETIS [5]) designed to optimize matrix fill-ins;
- Matrix preconditioning techniques, including e.g. incomplete LU or multi-grid preconditioning [12];
- Fill-in based matrix partitioning (cf. Sect. 4.2);
- Parallelization techniques for linear solvers;
- Algorithms for selecting a linear solver (direct or iterative).

The ‘matrix-based’ approach is also applied in many leading Model Order Reduction (MOR) methods for constant positive real submatrices of the system matrix (corresponding to passive, linear portions of the circuit). Such MOR algorithms include Krylov subspace methods [2], or truncated balanced realization [10]. Finally, matrix approach is used in weak fill-in elimination, e.g. in techniques which compute Schur complement to discover weak couplings.

3.2.3 Graph-Based Viewpoint

On the other hand, the 'graph-based' approach stimulated development of the optimization technologies such as:

- Hierarchical circuit data processing and storage, leading to vast improvements in performance and capacity, predominantly in the simulation setup phase.
- Isomorphic matching techniques applied during transient simulation phase, which avoid repeated simulation of sufficiently identical circuit blocks ([1, 15]).
- Effective and efficient methods for topological identification of structure of circuit blocks (such as e.g. memory arrays), leading to methodologies such as hierarchical array reduction (cf. [7], Sect. 4.3).

It should also be mentioned that graph *concepts* are heavily exploited in many other fast-SPICE techniques, such as e.g. minimum degree or graph dissection-based node ordering schemes, as well as MOR methods already discussed above.

3.2.4 Circuit-Based Viewpoint

Last but not least, the 'circuit-based' approach has been used to develop a broad spectrum of simulation optimization techniques. Firstly, automatic or semi-automatic identification, and basic characterization of functional blocks for the simulated circuit allows one to apply optimized modeling approaches to specific block components. In particular:

- For transistors in a block which are found to be operating in digital regime, high performance approximate models (e.g. based on lookup tables) can be applied instead of full-precision equation-based models;
- For entirely linear circuit sub-blocks, model order reduction techniques can be used to generate efficient macromodels;
- Different integration, preconditioning, and solve algorithms can be applied to signal blocks either identified as digital or analog.

Also, the physical or circuit point of view allows one to effectively partition the computational problem. For problem partitioning purposes, fast-SPICE tools most often exploit knowledge on CMOS-based designs. For instance, identifying MOSFET channel connected blocks provides the very simplest partitioning technique. Once functional partitioning is done, sub-systems corresponding to different blocks can be solved separately, different integration time-points and time-steps can be used in each block, and achieving global convergence across different block may be optional. Also, solves for latent blocks may be avoided altogether.

Above we have classified various techniques for accelerated transistor-level simulation by the dominant approach (circuit-, graph-, or matrix-based) towards the computational problem. In fact, highly successful technologies most often *simultaneously* apply all the three discussed views. Next, we will present examples of state-of-the-art fast-SPICE technologies which from the start were developed with such holistic approach in mind.

4 Examples of Fast-SPICE Technologies

This section illustrates how different viewpoints at the simulation problem discussed above are exploited and combined in some advanced fast-SPICE technologies. The selected examples of computational techniques reach beyond the usual methods implemented in the initial fast-SPICE tools, such as methods for generating table transistor models, or simple problem partitioning based on MOSFET channel connected blocks (cf. e.g. [3]). Outlined are comprehensive techniques including an approach for accelerating simulation of parasitic networks, an advanced partitioning methodology, as well as a technique for accelerating simulation of memory circuits. All three technologies not only exemplify the holistic approach to the simulation problem, but also illustrate a general shift of focus in fast-SPICE from MOSFET-centric optimization methods, to acceleration techniques for problems which include simulation of coupling and other parasitic effects at a massive scale. Such effects challenge many of the simple heuristic acceleration techniques applied in older fast-SPICE tools, and point toward broader optimization strategies.

4.1 Optimized Simulation of Parasitic Networks

Fast-SPICE tools have applied various techniques for optimized simulation of passive RC and RLC networks. Those techniques have significantly evolved as parasitic effects became more and more pronounced, and as sizes of extracted R(L)C networks exploded.

Initially, relatively simple reduction techniques using the ‘circuit-based’ view of the RC network were developed. Those techniques applied simple transformations such as splitting floating capacitors, or shifting ground capacitors from one node to another in order to eliminate nodes [14]. Simple physical criteria (e.g. estimated signal delay, relative capacitor value) were used to decide if a transformation could be performed (cf. e.g. [13]).

Later, more advanced techniques exploiting the matrix (or state-space) view of the problem were applied in fast-SPICE in order to reduce the number of nodes in RLC networks. Those included various methods which used state-space projections to generate a more compact matrix for the linear (RLC) network, e.g. Krylov subspace methods, projections based on selecting dominant poles, truncated balanced realization etc. [2, 6, 10]. Such matrix-based techniques were often able to compress linear networks more aggressively than physical, circuit-based approaches.

Yet, both classes of algorithms mentioned above focused their efforts on reducing the number of nodes in a linear network (i.e. model order reduction). Such narrow focus, as well as interpreting the problem only in terms of either an electrical circuit or a state-space system, often rendered those methods unable to generate compressed networks which would result in faster simulations. In particular, matrix-based MOR algorithms at times yielded disastrous results for large linear networks, producing reduced order models which were dramatically slowing down simulations instead of speeding them up [11].

Nevertheless, broader look at the computational problem enabled further advancement of model order reduction (MOR) methods. For one class of recently developed successful MOR techniques (an example of such method is presented in [11]), the focus is shifted from reducing the number of nodes in a linear network to minimizing the number of fill-ins produced during matrix factorization, which is directly connected to the cost of linear solves. This is achieved by incorporating information on factorization into RC network node reduction process. For instance, method [11] employs the same node ordering algorithm during node reduction (when it decides whether a node should be eliminated or not), and during matrix factorizations. The cited method heavily relies on a graph view of the linear network to be reduced, using such graph theoretic concepts as articulation points, as well as approximate minimum degree (AMD) node ordering for reduction. Such comprehensive approach which integrates model order reduction, a graph-based view of the problem, as well as a matrix-based view associated with solving of the resulting linear system, leads to a method capable of creating efficient compressed models for large-scale linear networks with a huge number of driving terminals, yielding large speedups during simulation phase. Also, the applied graph-based approach resulted in a reduction method which is highly scalable, as well as on its own computationally inexpensive to execute. This is particularly important in fast-SPICE applications where potential for reuse of reduced models is very limited.

4.2 Advanced Partitioning Technologies

As already mentioned in previous sections, another key fast-SPICE feature, is partitioning of the computational problem into subproblems (with commensurate partitioning of the corresponding system matrix into submatrices). Efficient partitioning substantially reduces the cost associated with linear solve of a single system, and thereby makes simulation of circuits with tens of millions of nodes tractable.

Early fast-SPICE incarnations most often used the concept of strong feed-forward i.e. weak dependence of voltage (e.g. of a digital gate or a voltage source) on the current through a MOSFET gate (or through a voltage source) to find cut points. Such simple schemes, basically producing partitions grouping channel-connected MOSFET devices, proved extremely effective and efficient for predominantly digital circuits connected to simple power supply consisting of ideal voltage sources.

Yet, as focus of fast-SPICE has shifted from large, pre-layout mostly digital circuits with simple power supplies, to mixed-signal, full-chip simulation which has often included post-layout data, efficient partitioning became much more challenging. For circuits with nonideal power supply networks, including internal power supplies (e.g. for phase-locked loops with charge pumps), or circuits with switched power supply, simple partitioning schemes often produced unacceptably large partitions, yielding inadequate simulation performance. In order to work at all older fast-SPICE tools frequently required user-provided information on nodes which one should partition across, as well as performed synchronous solves between different partitions.

Recently much more advanced partitioning techniques have been proposed, providing efficient, automatic partitioning for large-scale complex mixed-signal designs. Technologies, such as the one presented in [3], apply a much more comprehensive view at the system partitioning problem. In this approach elaborate automatic identification of circuit blocks is first performed, e.g. identification of power supply networks (including internal power supplies), as well as of digital circuit regions. Also the graph-based view is used to perform topological analysis of the circuit in order to identify devices which can (or cannot) be partitioned. Finally, matrix view of the problem is applied. In this view, matrix fill-in patterns are automatically reviewed as well as solve/factorization cost analysis is performed in order to find additional partitioning nodes [3]. Again, this comprehensive approach, combining different views at the computational problem, allows one to effectively partition even strongly coupled blocks, and subsequently simulate them in asynchronous manner. Consequently, the discussed technology outperforms many previously developed schemes, particularly for diverse advanced technology circuits, which include low-power designs, internal power supplies, and flash memory arrays.

4.3 Memory Simulation Acceleration

Another area in which fast-SPICE tools are expected to excel is simulation of memory circuits, including arrays ranging from hundreds of thousands to billions of cells (including DRAM, SRAM, FLASH, CAM etc.). Since memory circuits are encountered in virtually any chip, efficient memory simulation is required in order to achieve acceptable performance in full-chip simulations. This in turn implies application of specialized acceleration techniques which can handle memory circuits of enormous size by optimizing the associated simulation scheme and, most importantly, by reducing memory storage requirements.

On one hand, memory circuits are very challenging to simulate, due to a very large size of the corresponding system. Also, for a typical two-dimensional array topology (cf. Fig. 2) the bandwidth of the corresponding system matrix is $O(\sqrt{N})$, where N is the number of cells in the array. This implies that the cost of computing a simple matrix-vector product is $O(N^{1.5})$, which can be prohibitively expensive for large N.

On the other hand though, memory circuits are easy to simulate thanks to their very regular topologies, as well as the fact that in most cases, very large portions of the memory array remain latent, since both read and write operations typically activate only a very small fraction of the total number of cells at any given time.

Fast-SPICE technologies exploit both specific topology of memory arrays as well as their operating pattern in order to dramatically accelerate their simulation.

Initially techniques for accelerating memory circuit simulations focused on ‘ideal’ pre-layout memory arrays, i.e. arrays which do *not* include parasitic RC elements either inside the cells, or along bitlines or wordlines. Older methods often first identified portions of the array which stayed latent throughout the entire simulations, and subsequently removed those portions to lower memory cost and improve performance of the actual simulation (cf. e.g. Hierarchical Array Reduction [7]). Those techniques also frequently required a designer to manually specify the cell subcircuit, as well as heavily relied on efficient hierarchical representation of the memory circuit. Consequently, the methods were most suited for simulations which verified only portions of the entire memory array.

Subsequently, much more advanced technologies were developed, such as Spice Optimized for Arrays [4]. This method also merges various approaches toward the computational problem, yielding highly efficient memory models. In this technique graph-based approach is used to *automatically* detect a wide range of cell and memory topologies. Then, typical operating patterns of memory circuits are exploited to replace the entire array consisting of N cells, with $O(\sqrt{N})$ models, each model representing an entire row or column of cells. Such approach yields critical reduction of memory usage (from $O(N)$ to $O(\sqrt{N})$), and corresponding performance improvements. Since performance of this optimization technology is also independent of

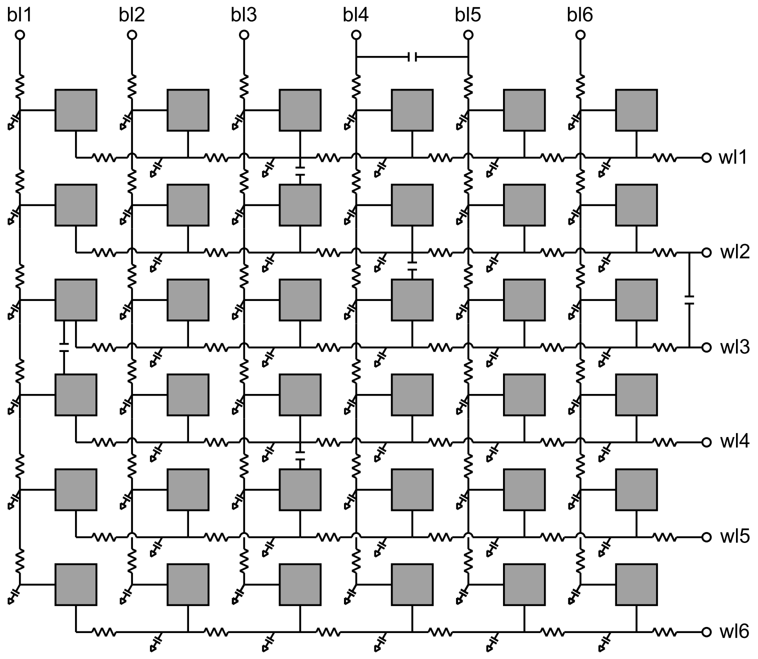

Fig. 2 Model post-layout memory array topology. Grayed boxes represent memory cells. Cells are connected with each other through bitlines and wordlines which are distributed RC lines. Also capacitive couplings are present between different cells, bitlines and wordlines

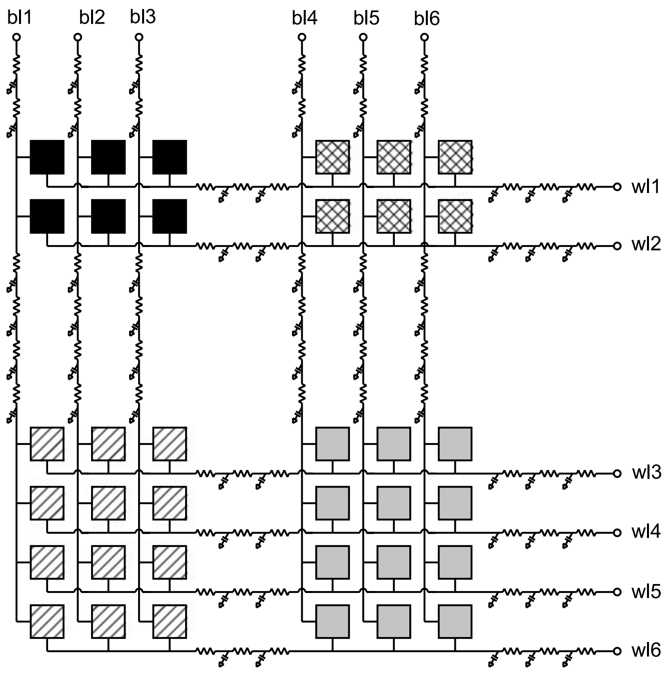

Fig. 3 Cells are automatically reconnected to different nodes located along distributed RC bitlines and wordlines according to information on signal delays along the lines. Consequently, groups of cells which connect to the same bitline or wordline nodes are formed (marked with different patterns). Such groups or subarrays can be efficiently modeled using techniques for pre-layout memory arrays such as Spice Optimized for Arrays [4]

the simulated memory access pattern, it makes the method well suited for extensive memory verification runs [4], which simulate accessing e.g. 100% of cells.

Verification of *post-layout* memory arrays has brought simulation challenges to yet another level. The new issues included:

- Huge number of RC elements inside the array, modeling delays along bitlines and wordlines, and parasitic effects inside the cells;
- Numerous capacitive couplings between bitlines, wordlines and different memory cells (cf. Fig. 2);
- Irregularities in topology of the memory schematic due to varying distribution of parasitic RC elements across different array sub-blocks and cells;
- Varying cell transistor parameters across different cells.

Consequently, simple acceleration techniques based on isomorphism, hierarchical simulation, and idealized array topology for pre-layout memories could not be readily applied in the post-layout case.

Effective technology for accelerated simulation of post-layout arrays required combining numerous state-of-the-art optimization methods, as well as—again—applying diverse views at the computational problem. One such technique, has been

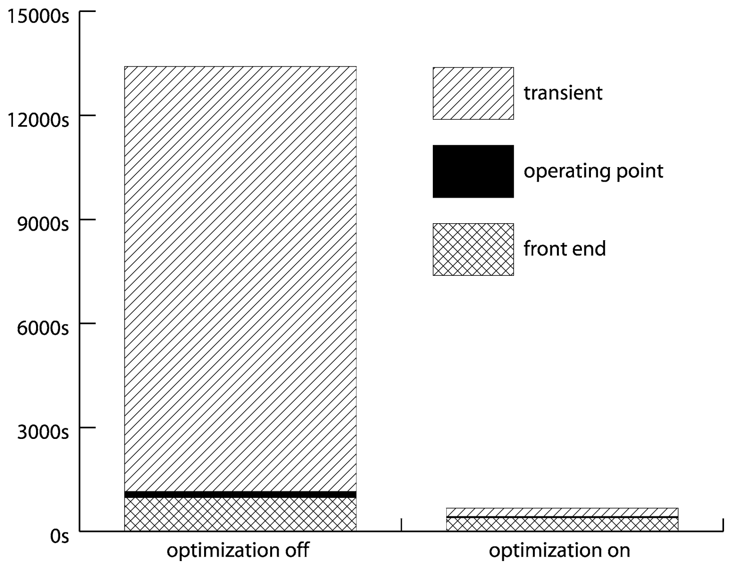

Fig. 4 Reduction of runtime for different simulation phases thanks to post-layout memory optimization technology [9]

presented in [9]. Firstly, it extends automatic graph-based cell and array topological detection to post-layout case. Then, in order to make use of techniques for 'ideal' memory circuits such as [4], it reconnects cells to different nodes located on distributed RC bitlines and wordlines, and hence ideal sub-arrays are formed (cf. Fig. 3). Each of these sub-arrays can be effectively optimized using e.g. method [4]. In order to reconnect cells along bitlines and wordlines while maintaining desired accuracy target, rigorous yet compact models for bitlines and wordlines are automatically generated using matrix-based RC optimization (MOR) algorithms. Fast approximate simulation of read and write operations using those compact bitline and wordline models yields information on feasible cell grouping. Furthermore, automatic optimizations of capacitive couplings between bitlines, wordlines and cells, as well as optimizations of parasitics located inside cells are applied [9].

The comprehensive optimization approach discussed above results in substantial performance gains for simulations verifying post-layout memory circuits. Figure 4 shows runtime reduction when optimization [9] is applied during simulation of a memory circuit with 0.5 million SRAM cells, and 2.6 million parasitic elements (created during post-layout RC extraction performed for the entire memory array). Runtime is reduced at each simulation stage: the setup phase ('front end'), computation of the operating point, and transient simulation, yielding an overall 50X speedup. No less significant are savings in memory required to complete the simulation, as shown in Fig. 5.

Additional advantages of newer memory optimization technologies ([4, 9]) over older approaches become even more apparent when large portions of memory array

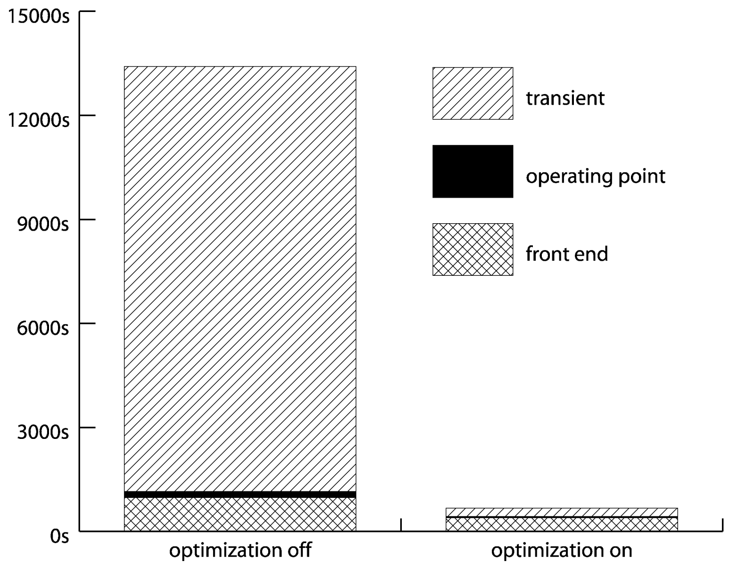

Fig. 5 Reduction of memory usage for different simulation phases thanks to post-layout memory optimization technology [9]

are tested, i.e. when the verification pattern involves accessing (reading/writing) a large percentage of the total number of cells. Figure 6 compares runtimes for a series of simulations with varying percentage of the total number of accessed cells, for an older acceleration technology and the new one [9]. The tests were performed for the same memory circuit as the one described above. Analogous comparisons for memory usage are presented in Fig. 7. First of all, the new technology yields approximately 17-fold speedup over the older solution, for an input pattern which exercises all the cells. Even more importantly, it is apparent from Fig. 7, that while for the older solution the simulator memory usage increases with the increasing percentage of exercised memory cells, it stays constant for the new technology. Hence, the new solution is particularly well suited for simulations which perform extensive memory verification.

5 Challenges of Fast-SPICE and Future Research

Previous sections discussed how different approaches toward the problem of computational modeling of an integrated circuit at transistor level lead to various simulation acceleration technologies embedded in fast-SPICE tools. It has also been shown that combining those approaches or viewpoints toward the simulated system leads to very effective and efficient techniques, which address some of the key problems

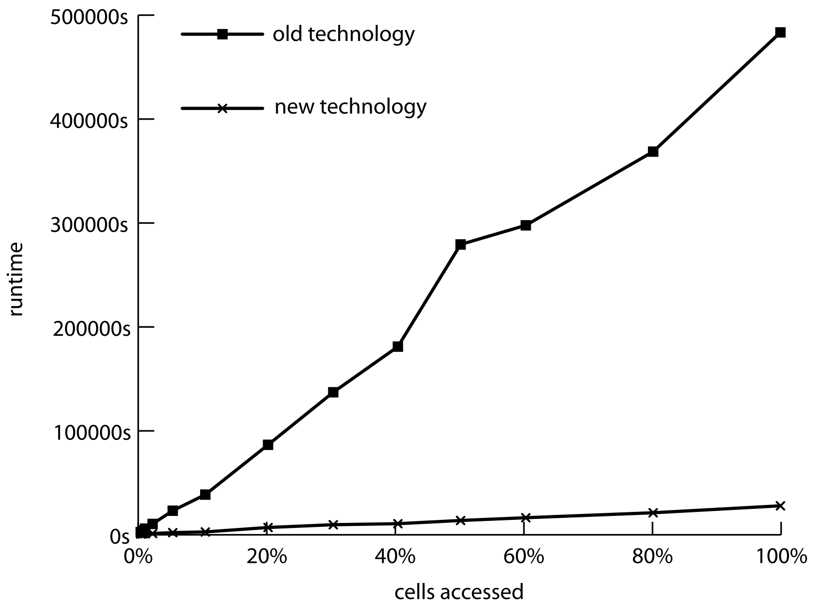

Fig. 6 Comparison of runtimes between simulations applying different memory acceleration techniques—an old one, and a new one [9]. Runtimes are given for a series of simulations with different percentage of the total number of cells accessed

including simulation of large parasitic networks and memory circuits, and effective partitioning of the computational problem.

Despite enormous progress of technology for accelerated transistor-level simulation, new problems keep emerging at an astonishing rate, continually challenging developers and researchers in the fast-SPICE area. Firstly, new challenges arise due to advancing CMOS process technology which makes electrical simulation increasingly more complex. In particular:

- Complexity of MOSFET models keeps increasing, along with variability in transistor model parameters;
- Simulated circuits become increasingly parasitic-dominated, and include massive capacitive couplings which challenge many traditional fast-SPICE heuristics and acceleration methods;
- Ever growing portions of the simulated circuit exhibit analog behavior which requires more careful modeling;
- Power supply networks become increasingly more extensive and complex;
- Architectures for memory circuits become more complicated;
- Element count in full-chip simulations keeps increasing.

Apart from the factors mentioned above, which make the system of equations to be solved larger and more complex, progress in manufacturing technology and IC design also change simulation needs of IC designers. There is a growing demand for simulation capabilities which go beyond traditional fast-SPICE scope, including:

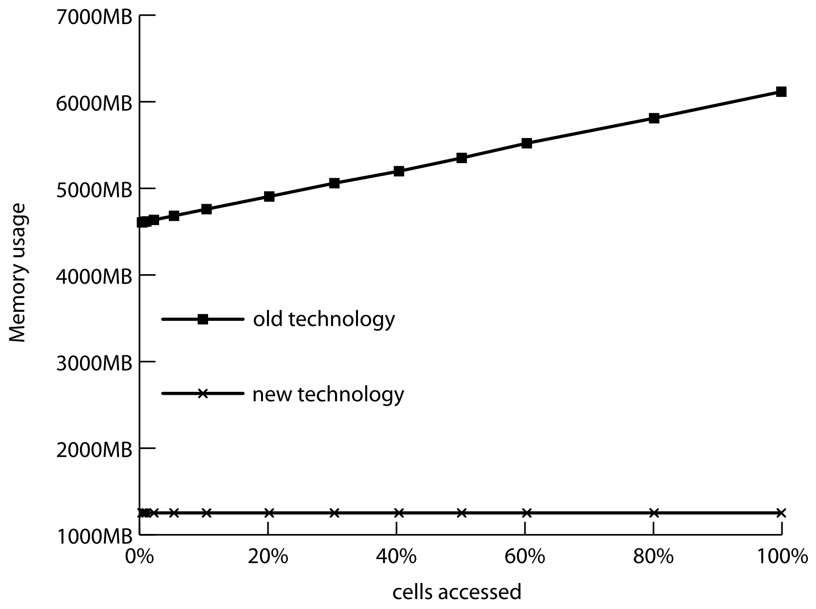

Fig. 7 Comparison of memory usage between simulations applying different memory acceleration techniques—an old one, and a new one [9]. Memory footprints are given for a series of simulations with different percentage of the total number of cells accessed

- Statistical fast-SPICE circuit simulations;
- Chip power-up simulations;
- Reliability analysis, including simulations for current density, electromigration, IR drop, device aging;
- Near-SPICE accuracy in full-chip simulations.

In order to satisfy more sophisticated fast-SPICE simulation needs, as well as growing costs of computations associated with more complicated and larger systems, substantial research efforts are expected on a number of topics. Firstly, more attempts will be made to bridge accuracy gap between SPICE and fast-SPICE tools. Next, tradeoffs associated with new hardware architectures (GPUs, multi-core architectures) will be further investigated. Another focus area will concern techniques for efficient simulation of power supply networks, as well as power-up effects. Finally, fast-SPICE developers are expected to include even more automation in the simulators, in order to improve their ease of use and robustness of the results. As a result of those efforts, a new generation of fast-SPICE tools will undoubtedly emerge, once again pushing performance and capacity limits, and challenging many of today's simulation paradigms.

Acknowledgements The author would like to express his gratitude to Kevin Kerns and Mayukh Bhattacharya for their mentoring throughout the years and for all the valuable discussions, which provided a lot of inspiration for writing this text. Lastly, the author would like to thank Sylwia Rewieńska for her help with preparing this chapter's drawings.

References

1. Cadence White Paper (2004), Using Hierarchy and Isomorphism to Accelerate Circuit Simulation, available at http://w2.cadence.com/whitepapers/5084_AccelCircuitWP_FNL.pdf
2. Celik M, Pileggi L, Odabasioglu A (2002), IC Interconnect Analysis, Kluwer Academic Publishers, Boston
3. Kerns KJ, Bhattacharya M, Rudnaya S, Gullapalli K (2007), Automatic, Hierarchy-Independent Partitioning Method for Transistor-Level Circuit Simulation, patent application submitted to U.S. Patent and Trademark Office (pending)
4. Kerns KJ, Peng Z (2008), SPICE optimized for arrays, U.S. patent no. 7,324,363
5. Karypis G, Kumar V, hMETIS 1.5: A hypergraph partitioning package, Technical Report, Department of Computer Science, Univ. of Minnesota, available at http://www.cs.umn.edu/~metis
6. Kerns KJ, Yang AT (1998), Preservation of passivity during RLC network reduction via split congruence transformations, IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 17, no. 7, pp. 582–591
7. Synopsys NanoSim User's and Reference Guide, 2004
8. Peng Li, Pileggi L (2002), A linear-centric modeling approach to harmonic balance analysis, in Proc. Design, Automation and Test in Europe Conf., pp. 634–639
9. Rewienski M, Kerns KJ (2007), Optimization of Post-Layout Arrays of Cells for Accelerated Transistor Level Simulation, patent application submitted to U.S. Patent and Trademark Office (pending)
10. Phillips JR, Silveira LM (2005), Poor Man's TBR: A simple model reduction scheme, IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 24, no. 1, pp. 43–55
11. Rommes J, Schilders WHA (2010), Efficient Methods for Large Resistor Networks, IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 29, no. 1, pp. 28–39
12. Saad Y (2003), Iterative Methods for Sparse Linear Systems, 2nd ed., SIAM, Philadelphia
13. Sheehan BN (1999), TICER: Realizable reduction of extracted RC circuits, in Proc. IEEE/ACM Int. Conf. Comput.-Aided Design, pp. 200–203
14. Sheehan BN (2007), Realizable Reduction of RC Networks, IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 26, no. 8, pp. 1393–1407
15. Tcherniaev A, et al. (2003), Transistor level circuit simulator using hierarchical data, U.S. patent no. 6,577,992
16. Zhao Li, Shi C-JR (2006), SILCA: SPICE-accurate iterative linear-centric analysis for efficient time-domain simulation of VLSI circuits with strong parasitic couplings, IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 25, no. 6, pp. 1087–1103