Full-TCAD Device Simulation of CMOS Circuits with a Novel Half-Implicit Solver

Ding Gong and Chen Shen
Cogenda Pte Ltd, Singapore. Email: {gongding, shenchen}@cogenda.com

***Abstract*—In this paper, we report full-TCAD device simulation of CMOS circuits with 24 transistors, 1.67 million mesh nodes, and simulation time of 4.5 hours. Simulation of this scale and speed is now enabled by an improved half-implicit algorithm for solving the Shockley equations, and fine-grained tuning for parallel computation efficiency. The solver algorithm is described first, followed by its accuracy and performance benchmarks, and finally its application on large scale problems.**

I. Introduction

TCAD device simulation was traditionally considered as the tool for analyzing a single device, or at best, small circuits such as an inverter or an SRAM cell. Simulation of circuits structures with greater than 1 million mesh nodes was impractical due to scaling difficulties:

- The metal wires in the circuit have much higher conductivity than the semiconductor regions. The difference in conductivity greatly degrades the condition number of the Jacobian matrix. In non-trivial circuits, floating nodes and feedback loops often occurs, which greatly degrades the condition number of the Jacobian matrix. In our tests, condition number as high as 10^{20} is commonly observed, which is impossible to solve with iterative solvers.
- In order to cope with the ill-conditioned matrix, direct solvers are required to get reasonable convergence. Direct solvers are memory-hungry and do not enjoy much speed-up in parallel computation. Since solving the matrix constitutes up to 90% of the total computation time, this becomes the bottle-neck of TCAD device simulation.

While it is difficult to develop a ”silver-bullet” solver that is suitable for all large-scale TCAD simulations, in this paper, we report an algorithm that is suitable for the transient simulation of CMOS devices and circuits.

II. Improved Half-Implicit Solver Algorithm

The Half-Implicit algorithm on transient semiconductor device equations was first proposed by Mock [1] and later improved by Polsky [2]. As opposed to the classical implicit solving algorithm [3], the half-implicit method starts with replacing the Poisson equation by its mathematically equivalent total-current continuity equation. It then solves the carrier-continuity equation and the total-current continuity equation sequentially, instead of as coupled-equations. This produces a non-self-consistent charge error, leading to inaccurate solution and sometimes oscillations. Polsky introduced a modified Poisson's equation to correct the electrostatic potential [2]. In practice, the accuracy and stability was still unsatisfactory. Despite these problems, due to the smaller size of equations to be solved, it has the potential to handle large-scale TCAD simulations.

To further improve the half-implicit algorithm, we propose to additionally correct the carrier concentration. The improved algorithm is illustrated in Figure 1, in which the proposed new step is highlighted.

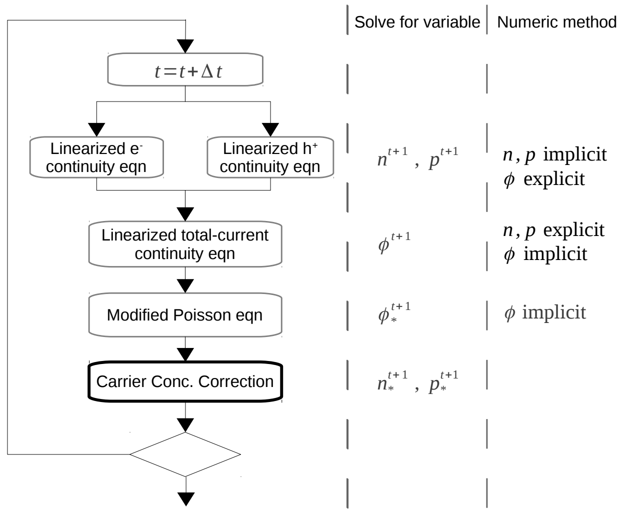


Fig. 1: Flow chart of the proposed half-implicit algorithm. Compared to Polsky's algorithm, a carrier-concentration correction step is added in each time step.

In each time step, the linearized electron and hole continuity equations

$$\frac{n^{t+1}-n^t}{\tau} = -\frac{1}{q_0}\nabla\cdot\left(\mu_n n^{t+1}\nabla\phi^t - D_n\nabla n^{t+1}\right) + G - R \tag{1a}$$

$$\frac{p^{t+1}-p^t}{\tau} = +\frac{1}{q_0}\nabla\cdot\left(\mu_p p^{t+1}\nabla\phi^t + D_p\nabla p^{t+1}\right) + G - R. \tag{1b}$$

is first solved. Note that on the right-hand side, the carrier concentration of the current step is used, and the potential of the previous time step is used. Therefore, the equation is implicit on n and p, but explicit on ϕ, hence the name half-implicit.

Next, the total-current continuity current equation (2)

$$\begin{aligned}\frac{\nabla\cdot\varepsilon\nabla\phi^{t+1} - \nabla\cdot\varepsilon\nabla\phi^t}{\tau} &= -\nabla\cdot\left(\mu_n n^{t+1}\nabla\phi^{t+1} - D_n\nabla n^{t+1}\right)\\ &\quad -\nabla\cdot\left(\mu_p p^{t+1}\nabla\phi^{t+1} + D_p\nabla p^{t+1}\right),\end{aligned} \tag{2}$$

which is a variant of the Poisson's equation, is solved. In this step, the equation is implicit on ϕ, but explicit on n and p.

In order to correct for the non-self-consistent charge introduced by the de-coupled solution of (1a), (1b) and (2), we first solve the modified Poisson's equation (3)

$$\begin{aligned}\nabla \cdot \varepsilon \nabla \phi_*^{t+1} &= -q_0 \left(N_D - N_A + p^{t+1} - n^{t+1}\right) \\ &\quad + \alpha \left(n^{t+1} + p^{t+1}\right)\left(\phi_*^{t+1} - \phi^{t+1}\right) \qquad (3)\end{aligned}$$

proposed by Polsky [2], where α is an empirical parameter very close to $1/V_T$. In the next time step, we use ϕ_*^{t+1} instead of ϕ^{t+1} as the corrected electrostatic potential.

The key proposal of this study is to further correct the carrier concentration (highlighted step in Figure 1), and use the corrected n_*^{t+1} and p_*^{t+1} in the next time step. Two correction schemes have been found to be effective and efficient.

Scheme I: The corrected carrier concentrations are given by

$$n_*^{t+1} = n^{t+1}\left[1 + \alpha\left(\phi_*^{t+1} - \phi^{t+1}\right)\right] \qquad (4a)$$

$$p_*^{t+1} = p^{t+1}\left[1 - \alpha\left(\phi_*^{t+1} - \phi^{t+1}\right)\right], \qquad (4b)$$

where n^{t+1}, p^{t+1} are the carrier concentration solved from the carrier continuity equation, ϕ^{t+1} is the potential solved from the total-current equation, and ϕ_*^{t+1} is the corrected electrostatic potential obtained from the modified Poisson equation. This correction does not guarantee conservation of charge, but the error in charge is in practice negligible.

Scheme II: The carrier continuity equation (5a), (5b) is solved again to obtain the corrected charge

$$\begin{aligned}\frac{n_*^{t+1} - n^t}{\tau} &= -\frac{1}{q_0}\nabla \cdot \left(\mu_n n_*^{t+1} \nabla \phi_*^{t+1} - D_n \nabla n_*^{t+1}\right) \\ &\quad + G - R \qquad (5a)\end{aligned}$$

$$\begin{aligned}\frac{p_*^{t+1} - p^t}{\tau} &= +\frac{1}{q_0}\nabla \cdot \left(\mu_p p_*^{t+1} \nabla \phi_*^{t+1} + D_p \nabla p_*^{t+1}\right) \\ &\quad + G - R. \qquad (5b)\end{aligned}$$

This scheme guarantees charge conservation, but involves solving additional equations, and hence is slower.

It can be verified through spectral analysis that the proposed scheme I and II both provide better numerical stability than Polsky's algorithm. In particular, Scheme II is free from numerical oscillations caused by the non-self-consistent charge.

III. Accuracy and Performance Benchmarks

The proposed algorithm was implemented in the Genius 3D device simulator [4], which is fully parallelized in both equation assembly and the solving. Since all equations are linear, in each time step, several linear matrices of rank N are solved in sequence, where N is the number of mesh nodes. A fully-parallelized, fully-implicit solver is also available in Genius as the baseline algorithm.

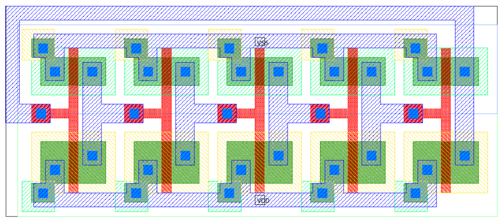
Unless otherwise noted, in the implicit solver, the inexact Newton method (with potential damping) is used to solve the nonlinear equation, and the parallel sparse direct solver MUMPS [5], [6] is used as the linear solver. On the other hand, in the half-implicit case, the same direct solver is used for the total-current continuity equation, and the preconditioned GMRES iterative solver is used in the other equations. Most simulations are run on a workstation with 2×Xeon E5620 processors, using 8-way parallelism.

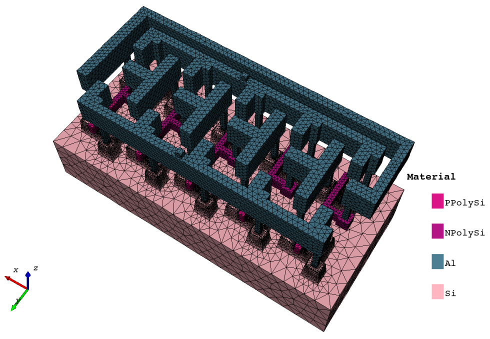
TABLE I: Performance and Accuracy comparison between implicit and half-implicit algorithms, tested on the 5-stage ring-oscillator with 250K mesh nodes.

	Implicit	Half-implicit		
		Polsky	Scheme 1	Scheme 2
Run time (min)	3,691	535	562	909
Speed-up	–	6.9×	6.6×	4.1×
RAM used (GB)	23	11	11	11
Stage delay (ps)	28.18	31.39	29.41	29.28
Error in delay (%)	–	+11.4	+4.36	+3.9

A 5-stage Ring-Oscillator(Fig. 2), among other circuits, was used for accuracy and performance benchmark. The 3D RO model was built from the mask layout with a generic CMOS technology, and has 251,026 mesh nodes. Figure 3 shows the simulated waveform of the RO. Table I compares the simulation run time and the accuracy (measured by the difference of stage delay) w.r.t. the traditional implicit solver.

The half-implicit algorithm is known to require smaller time steps [2], but each time step takes much shorter time, and the total simulation time is shorter. It is seen that the proposed improved half-implicit solver achieves 4-7x speed-up for the 250K-node problem. Greater speed-up can be expected for larger problems. The error in rise time and stage delay is within 5% (or 1.2ps).


(a) Mask



(b) Mesh

Fig. 2: a) Mask layout, b) TCAD model of a 5-stage ring-oscillator.

IV. Scaling Performance in Parallel Computation

The proposed algorithm scales up well in parallel computation. In Figure 4, one can see that scaling only starts to degrade

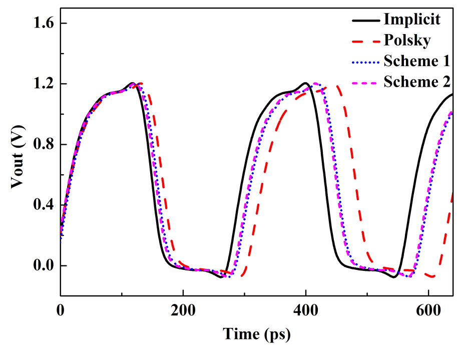


Fig. 3: Two periods of the simulated waveform of the 5-stage ring-oscillator, using the implicit and various half-implicit algorithms. Simulated stage delay is shown in Table I.

at 36 cores (on 3 computing nodes), and scaling does not yet saturate at 96 cores.

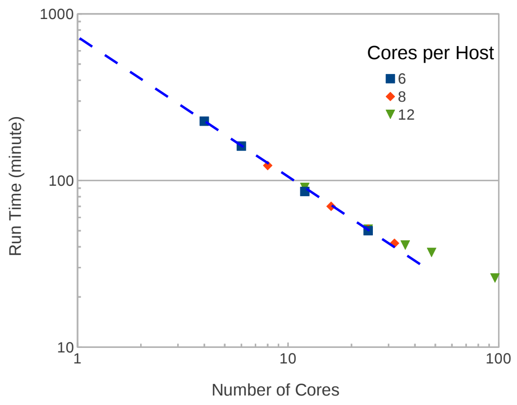


Fig. 4: Parallel scaling performance of the proposed algorithm on a device with about 300K mesh nodes. Simulation time scales well with increased number of CPU cores used in the simulation up to 96 cores.

The run time of TCAD simulation is dominated by two modules in the simulator, 1) the assembly of the equations (and the Jacobian matrix), and 2) solving the linear equations. In traditional TCAD simulators using the implicit algorithm, the latter is the eclipsing factor, constituting up to 90% in the total run time. It is seen from II that the implicit solver in Genius also follows this pattern.

The above two components have very different scaling properties. In a well-tuned parallel simulator, the equation assembly module shows near-perfect parallel scaling, i.e. the run time scales linearly with the number of mesh nodes, and inverse proportionally with the number of processors. On the other hand, the linear solver shows very poor scaling. The time cost of linear solvers is $O\left(N^{\gamma}\right)$, with $1.8 < \gamma < 2.2$, and does not scales up very well in massively parallel computation.

The excellent scaling performance of the half-implicit solver stems from the fact that it sequentially solving several smaller systems of equations, rather than solving a large one. As shown in Table II, in the half-implicit algorithm, the linear solver takes much less proportion in the total run time, the time expensed in equation assembly and linear solver modules are more balanced. The desirable scaling property is preserved in a larger problems as well, as shown in Table III. As a result, the half-implicit algorithm is able to fully take advantage of parallelism and scale up nicely to very large problems containing more than 1 million mesh nodes.

TABLE II: Run time of a simulation on an inverter circuit with 137K mesh nodes using the half-implicit and the implicit algorithm, and broken down to the individual modules in the solver.

Module	Half-Implicit Time (s)	Half-Implicit Percent	Implicit Time (s)	Implicit Percent
Equation Assembly				
Carriers Eqns	355	27%		
Total Current Eqn	368	28%		
Correction Eqn	21	2%		
Total	744	57%	6206	29%
Linear Solver				
Carriers Eqns	33	3%		
Total Current Eqn	455	35%		
Correction Eqn	17	1%		
Total	505	39%	15062	71%

TABLE III: Breakdown of the run time on a larger SRAM circuit with 580K mesh nodes.

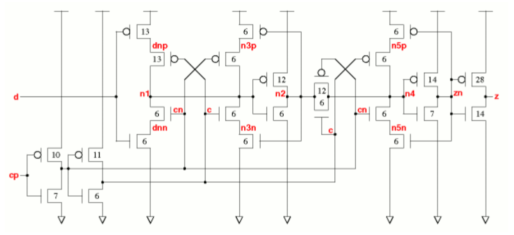
Module	Time (s)	Percent
Equation Assembly		
Carriers Eqns	2293	27%
Total Current Eqn	2362	28%
Correction Eqn	104	2%
Total	4759	57%
Linear Solveer		
Carriers Eqns	208	2%
Total Current Eqn	2613	31%
Correction Eqn	95	1%
Total	2916	35%

V. D-Flipflop with 1.67 Million Mesh Nodes

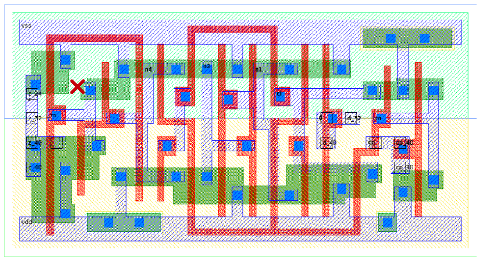
D-flipflop is one of the most complex circuits in a CMOS standard cell library. To test the scalability of the proposed algorithm, a TCAD model of a master-slave D-flipflop circuit with 24 transistors, 1,673,519 mesh nodes, and 10,659,866 tetrahedral mesh elements is constructed (Figure 6), for Single-Event Upset simulation (Figure 7). The active silicon region (with 300nm from the surface) contains majority (about 1 million) of the mesh nodes.

An 95.35 MeV $^{17}\mathrm{Cl}^{11+}$ ion strikes at a transistor in the slave-stage. The energy deposition of the ion in the device is simulated using Gseat, a particle simulator based on the GEANT4 library [8].

Transient simulation of 3000 ps is performed with 252 time steps, with the particle-generated carriers included as a generation term, to study the circuit's single-event upset (SEU) characteristics [9]. Four computing nodes are used; each nodes has two Xeon 5670 6-core processors, providing a total of 48 cores. The peak memory consumption is approximately

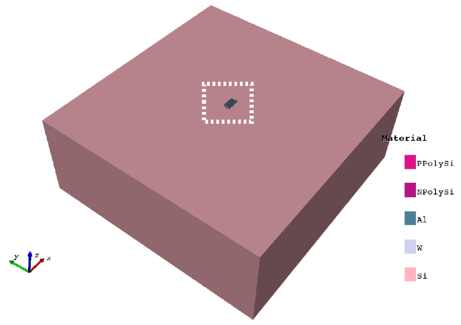


(a) Schematic

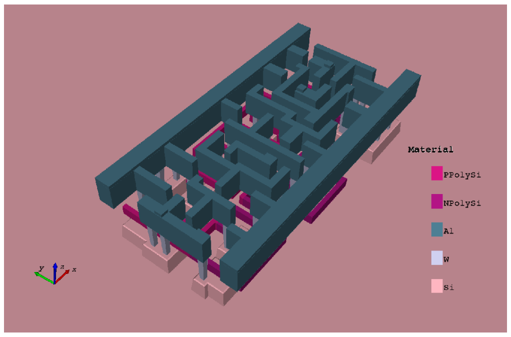


(b) Mask layout

Fig. 5: a) Circuit diagram, b) mask layout [7] of a positive-edge D-Flipflop circuit; The position where the ion hits the device is highlighted in the mask layout (X).



(a) Full TCAD model



(b) Zoom-in view

Fig. 6: a) the full TCAD model and b) the zoom-in view near the active region of the circuit. The large substrate is needed for accurate simulation of the spreading electrostatic potential caused by the incident heavy ion (funneling effect).

190GB, and the run time is 268 minutes. Simulation of this scale with the traditional implicit algorithm is projected to require over 1TB memory and take 3-4 days to complete, but the authors do not have the computing resource to verify this estimation.

VI. Conclusion

In this paper, full 3D TCAD simulation of non-trivial CMOS circuits is demonstrated to become practical. For the first time, TCAD device models with $>$ 1 million mesh nodes can be completed in several hours. The half-implicit solving algorithm is shown to offer 5x or more performance improvement, with $< 5\%$ error in the estimation of circuit timing, and scales well in parallel computation.

Extensive tests have been carried out on the numerical stability of the proposed algorithm. In the month prior to this writing, over 10,000 transient simulations were performed on SRAM cells of several technologies for SEU cross-section evaluation, no convergence failure has been seen.

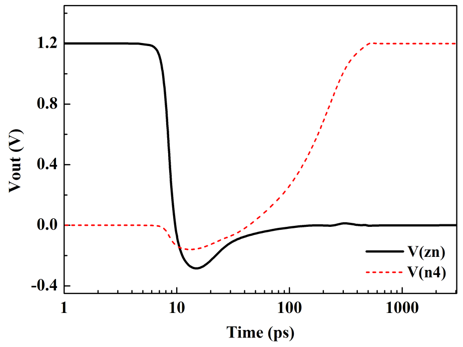


Fig. 7: Simulated voltage waveform of the zn and n4 nodes in the D-Flipflop, under 95 MeV $^{17}Cl^{11+}$ ion strike. Flip of the logic state is observed.

References

[1] M. S. Mock, “A time-dependent numerical model of the insulated-gate field-effect transistor,” *Solid-State Electron.*, vol. 24, pp. 959–966, 1981.

[2] B. S. Polsky and J. S. Rimshans, “Half-implicit difference scheme for numerical simulation of transient process in semiconductor devices,” *Solid-State Electron.*, vol. 29, pp. 321–328, 1986.

[3] S. Selberherr, A. Schtz, and H. Ptzl, “MINIMOS – a two-dimensional mos transistor analyzer,” *IEEE Trans. on Electron Dev.*, vol. 27, pp. 1540–1550, 1980.

[4] Cogenda Pte Ltd, “Genius device simulator: User guide,” [Online]. Available: http://www.cogenda.com/article/download.

[5] P. R. Amestoy, I. S. Duff, J. Koster, and J.-Y. L’Excellent, “A fully asynchronous multifrontal solver using distributed dynamic scheduling,” *SIAM J. of Matrix Analysis and Applications*, vol. 23, pp. 15–41, 2001.

[6] P. R. Amestoy, A. Guermouche, J.-Y. L’Excellent, and S. Pralet, “Hybrid scheduling for the parallel solution of linear systems,” *Parallel Computing*, vol. 32, pp. 136–156, 2006.

[7] G. Petley, “Dfnt1 standard cell family,” [Online]. Available: http://www.vlsitechnology.org/html/cells/vsclib013/dfnt1.html.

[8] K. Amako, J. Apostolakis, H. Araujo, and et al, “Geant4 developments and applications,” *IEEE Trans. on Nucl. Sci.*, vol. 53, pp. 270–278, 2006.

[9] P. E. Dodd, “Physics-based simulation of single-event effects,” *IEEE Trans. on Dev. and Mat. Rel.*, vol. 5, pp. 343–357, 2005.