An Unconditional Stable General Operator Splitting Method for Transistor Level Transient Analysis

Zhengyong Zhu, Rui Shi, Chung-Kuan Cheng
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093
Email: {zzhu,rshi,kuan}@cs.ucsd.edu

Ernest S. Kuh
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, CA 94720
Email: kuh@eecs.berkeley.edu

Abstract— **In this paper, we introduce a general operator splitting method for transient simulation of VLSI circuits. The proposed approach generates special partitions of the circuits and alternates the explicit and implicit integrations between the partitions. We prove that the method is unconditionally stable independent of the step size. The splitting scheme greatly reduces the nonzero fill-ins generated in direct methods like LU decomposition. Orders of magnitude speedup over Berkeley SPICE3 is observed for sets of circuits.**

I. Introduction

With increasing design complexity, huge size of extracted interconnect data is pushing the capacity of transistor level simulation tools to the limits. Direct methods like Gaussian Elimination used in Berkeley SPICE and its variations is prohibitive because of the above linear complexity O($n^{1.5}$) where n is the number of circuit nodes.

In last decade, there is rich literature in the field of circuit analysis to improve the performance of simulation under the rising demand from advanced technologies. Among them, Conjugate Gradient Method and Multigrid Method were used on linear networks [3]–[5]. Model order reduction methods [1], [2] reduced the circuit size by generating stable and passive macromodels in time domain simulation. For timing analysis, Acar et al. [6] introduced a waveform evaluation engine using Successive Chord and macromodeling approach.

At transistor level, Sakallah and Directors [7] saved unnecessary computation by applying different integration method (explicit or implicit) on subcircuits according to their activities. Li and Shi [11] tried to reduce the number and cost of LU decompositions by using low cost integration approximation and Successive Chord Method with approximated device model.

Since direct methods such as LU decomposition remain to be efficient for small circuits with up to tens of thousands of nodes, partition-based simulation methods are widely used in commercial tools [8]–[10]. However, the convergence of those methods cannot be guaranteed and is sensitive to the partition algorithm and propagation order.

The operator splitting method has been adopted to partition the system based on the geometry of the physical adjacency and the locality of the processes. In 1999, Namiki and Ito [16] adopted its special form, the Alternating Direction Implicit (ADI), to simulate a two dimensional electromagnetic wave. They demonstrated the unconditional stability of the finite difference time domain analysis independent of the time step size under the proposed geometric structure. Later, Zheng et al extended the structure to three dimensions [13]. Since then, the method has been applied to tackle huge problems for finite difference time domain analysis [18]. In 2001 and 2003, Lee and Chen proposed TLM-ADI approach [14], [15] for power grid analysis, based on implicit FDTD methods. In 2003, Guo and Tan applied the ADI method to circuit-level power grid analysis [20], which splits power mesh along horizontal and vertical directions and iterates between partitions at each time point till converge with fixed time step size.

In this paper, we present a generalized operator splitting method and demonstrate that the generalized method is unconditionally stable. Following the generalized approach, we partition the circuits using a network splitting algorithm with guaranteed DC paths and alternate the explicit and implicit integrations between the partitions. The splitting algorithm partitions the circuit into structures that produces much fewer nonzero fill-ins during LU factorization. Thus direct methods can remain efficient for large-scale circuits. Unlike [20], the proposed approach has no geometrical constrains and can handle general circuits. We can also prove that there is no iteration needed between partitions at each time point since the operator splitting approach is actually an A-stable numerical integration method and the local truncation error can be controlled by dynamic time step estimation.

The rest of this paper is organized as follows. General operator splitting method and its application on linear and nonlinear circuits are discussed in section II. Section III proves the unconditional stability of the proposed method. Section IV discusses the local truncation error (LTE) estimation and dynamic time step control. Experimental results are then demonstrated in section V. The paper is wrapped up with conclusion and future directions.

II. General Operator Splitting Method

The operator splitting method [17] was first introduced as a technique for solving partial differential equations. The basic idea of operator splitting can be explained with the following initial value problem (IVP) of a simple ordinary differential equation (ODE) [19],

$$\frac{\delta u}{\delta t} = Lu \tag{1}$$

where L is a linear or nonlinear operator and can be written as a linear sum of m suboperators of u,

$$Lu = L_1 u + L_2 u + \cdots + L_m u \tag{2}$$

Suppose $U_1, U_2, \cdots, U_m$ are updating operators on u with respect to $L_1, L_2, \cdots, L_m$ from time step n to time step $n+1$, the operator splitting approach has the form of:

$$\begin{array}{lcl} u^{n+(1/m)} & = & U_1(u^n, h/m) \\ u^{n+(2/m)} & = & U_2(u^{n+(1/m)}, h/m) \\ & \cdots & \\ u^{n+1} & = & U_m(u^{n+(m-1)/m}, h/m) \end{array} \tag{3}$$

where each partial operation acts with all the terms of the original operator.

Our invention introduced in the next subsection generalizes the operator splitting method to graph based modeling. The generalization frees us from the geometry or locality constraints. We prove that the method is unconditionally stable.

A. Formulation

We use a general circuit system to describe our operator splitting method. The circuit contains resistors, capacitors, and inductors with mutual couplings. For linear circuits, the modified nodal analysis using Backward Euler Integration can be expressed as below:

$$\begin{bmatrix} \frac{C}{h}+G & -A^T \\ A & \frac{L}{h}+R \end{bmatrix} \begin{bmatrix} V(t+h) \\ I(t+h) \end{bmatrix} = \begin{bmatrix} \frac{C}{h} & 0 \\ 0 & \frac{L}{h} \end{bmatrix} \begin{bmatrix} V(t) \\ I(t) \end{bmatrix} + U(t+h) \tag{4}$$

where C, L, R, G are the matrices of capacitances, inductances, resistances, and conductances. Matrix A is an incidence matrix linking between the topology of capacitance nodes and inductance branches.Vectors V, I, and U describes the voltages of capacitance nodes, currents of inductance branches, and system inputs. Scalar h is the time step from time t to $t+h$. Note that the four matrices, C, L, R, and G, are symmetric by construction and are positive semidefinite because the elements: capacitances, inductances, resistances, and conductances, are non-active. In addition, we can assume that matrices C and L are positive definite for a nondegenerated case.

The generalized operator splitting formulation allows us to make arbitrary partitions of the circuit. Thus, we have corresponding partitions of matrices A, R, and G, i.e. $A = A_1 + A_2$, $R = R_1 + R_2$ and $G = G_1 + G_2$. By construction, matrices R_i and G_i for $i \in \{1, 2\}$ remain to be symmetric and positive semidefinite. Following the circuit partition, we divide the integration into two half steps and alternates the forward and backward integrations between the partitions as shown in formulation (5). In the first half step, we use forward integration for the subcircuit with matrices A_2, G_2 and R_2. Then, in the second half step, we use forward integration for the subcircuit with matrices A_1, G_1 and R_1. In both half steps, the other partition is integrated by backward implicit integration.

$$\left\{ \begin{array}{l} \begin{bmatrix} \frac{2C}{h}+G_1 & -A_1^T \\ A_1 & \frac{2L}{h}+R_1 \end{bmatrix} \begin{bmatrix} V(t+\frac{h}{2}) \\ I(t+\frac{h}{2}) \end{bmatrix} = \\ \qquad \begin{bmatrix} \frac{2C}{h}-G_2 & A_2^T \\ -A_2 & \frac{2L}{h}-R_2 \end{bmatrix} \begin{bmatrix} V(t) \\ I(t) \end{bmatrix} + U(t+\frac{h}{2}) \\ \begin{bmatrix} \frac{2C}{h}+G_2 & -A_2^T \\ A_2 & \frac{2L}{h}+R_2 \end{bmatrix} \begin{bmatrix} V(t+h) \\ I(t+h) \end{bmatrix} = \\ \qquad \begin{bmatrix} \frac{2C}{h}-G_1 & A_1^T \\ -A_1 & \frac{2L}{h}-R_1 \end{bmatrix} \begin{bmatrix} V(t+\frac{h}{2}) \\ I(t+\frac{h}{2}) \end{bmatrix} + U(t+h) \end{array} \right. \tag{5}$$

If the two left-hand-side matrices correspond to trees or forest structures, a direct matrix inversion will be very efficient to solve those two equations because there is no nonzero fill-ins and the computational cost is linearly proportional to the number of elements.

Let $P_1 = \begin{bmatrix} G_1 & -A_1^T \\ A_1 & R_1 \end{bmatrix}$, $P_2 = \begin{bmatrix} G_2 & -A_2^T \\ A_2 & R_2 \end{bmatrix}$, $S = \begin{bmatrix} \frac{2C}{h} & 0 \\ 0 & \frac{2L}{h} \end{bmatrix}$, and $X = \begin{bmatrix} V \\ I \end{bmatrix}$ then the notation of the two half step of operator splitting formulation (5) can be simplified as:

$$\left\{ \begin{array}{lcl} (P_1+S)X(t+\frac{h}{2}) & = & -(P_2-S)X(t) + U(t+\frac{h}{2}) \\ (P_2+S)X(t+h) & = & -(P_1-S)X(t+\frac{h}{2}) + U(t+h) \end{array} \right. \tag{6}$$

B. Splitting Operation

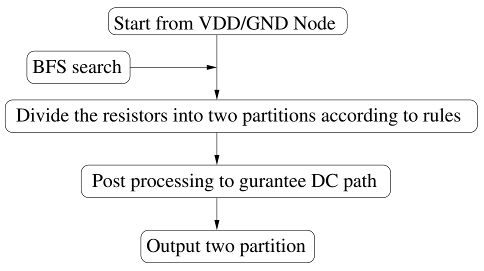
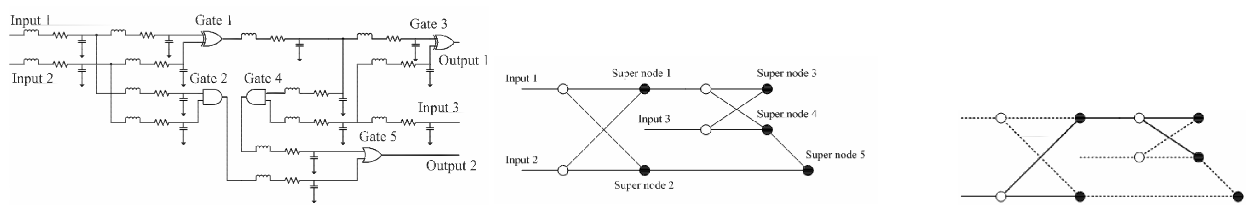

Start from VDD/GND Node
BFS search
Divide the resistors into two partitions according to rules
Post processing to gurantee DC path
Output two partition

Fig. 1. Splitting Algorithm Flow

1) Network Splitting with Guaranteed DC Paths: The performance of direct methods such as LU decomposition can still beat those of iterative methods for small circuits with up to tens of thousands of nodes. Direct methods become prohibitive for large circuits because of the significant amount of nonzero fill-ins generated during factorization. However, it can be proved that the LU decomposition method does not create nonzero fill-ins for circuits in tree/forest structure if nodes elimination always starts from leaves. The elimination order can be captured by ordering algorithms based on minimum degrees. Following this observation, the proposed operator splitting algorithm tries to split the circuits into two partitions in structures close to tree or forests such that the number of nonzero fill-ins is minimized. Even though general circuits may not be able to get optimized partitions in terms of the number of nonzero fill-ins because of its structure limitation or the restriction of DC paths (Splitting algorithm should not generate floating nodes at DC stage), the number of overall nonzero fill-ins is significantly reduced for most circuits.



(a) Example of a Simple Digital Circuit (b) Undirected Graph Representation (c) Splitting Result of The Undirected Graph

Fig. 2. Example for Undirected Graph Representation and Splitting

2) Applications on Linear Circuits: Only resistive connections are considered during partition. Capacitors and inductors are duplicated into both partitions. Resistors are divided into two partitions using graph theory algorithms. In order to obtain DC convergence, some adjustments are needed to ensure that every node in both partitions has a DC path to some voltage source. When solving each partition, the rest of circuit is modelled as equivalent current sources, following the operator splitting formulation (5). Detailed algorithm is discussed in section II-B.1.

3) Applications on Circuits with transistors: We extend the algorithm to handle circuits with transistors. In typical digital circuits, transistors are grouped as various gates. Taking into consideration the nonlinear property of transistor devices and gates, the proposed approach does not split a single transistor or gate into different partitions; instead, each partition has a full-version of all transistor devices. In other words, transistor devices are duplicated in both partitions and solved at every half time point.

The splitting algorithm actually regards each gate as a super node. The details inside each gate are invisible to the splitting algorithm. The same splitting operation for linear circuits is applied on the super node structure instead of the original circuits.

Moreover, the introduction of super node brings some advantages during LU decomposition if all the internal nodes in one gate are eliminated together. Even though each gate may have many internal nodes, it usually has only a few input/output nodes connecting to the outside, which implies that the nonzero fill-ins are confined inside each gate and will not propagate to outside if all the internal nodes in the same gate are eliminated together during LU decomposition. However, the ordinary minimum degree algorithm or the Markowitz Product method used in SPICE3 does not have a global view of the circuit structure and introduce many unnecessary nonzero fill-ins. We modified the ordering algorithm in SPICE3 to incorporate the super node concept.

We define an undirected graph $G = (V, E)$ to represent the circuit structure. There are two kinds of nodes in this graph: super node and branch node. Super node denotes end point of resistors in large linear networks or a single gate. Branch node represents end point of resistors on signal wires connecting gates in the circuit. The edge denotes the resistor branch in the circuit since only resistors are divided into partitions.

Figure 2 gives the undirected graph representation of a simple digital circuit. The original circuit shown in Figure 2(a) is mapped to an undirected graph in Figure 2(b). Figure 2(c) demonstrates one possible splitting result of the graph, where dashed lines and solid lines denote two different partitions.

The splitting algorithm divides the graph into two partitions. The objective is to minimize the total number of non-zero fill-ins of both partitions generated in LU decomposition. The basic idea of our rule-based splitting algorithm is to distribute the edges of every node into different partitions and avoid loops as much as possible, because sparse and tree-like structures produce much fewer nonzero fill-ins.

Figure 1 summarizes the splitting algorithm flow. The input information includes the undirected graph, super node identification and VDD/GND nodes set. There are two main steps, breath first search (BFS) partition and DC path post-processing. In the BFS partition step, we start from VDD/GND nodes simultaneously, go through all the nodes in the graph using BFS and divide the edges of every node into two partitions according to the partition rules defined below. Nodes and edges are associated with labels to record the partition and DC path connection status. Based on the labelling information, the partition rules are defined to benefit DC path available for all nodes. In the post-processing step, we adjust the partition for nodes without a DC path to VDD/GND.

We define seven types of node labels: CONNECTED, UNCONNECTED, ZERO_UNCONNECTED, ONE_UNCONNECTED, ZERO_ONE_UNCONNECTED, ZERO_CONNECTED, ONE_CONNECTED, where "ZERO" and "ONE" represent different partitions. The label describes the partition and DC path status of the node. For example, label "ZERO_UNCONNECTED" means the node has an edge in partition ZERO without a DC path to VDD/GND. Similarly, we define five types of edge labels: UNCONNECTED, ZERO_UNCONNECTED, ONE_UNCONNECTED, ZERO_CONNECTED, ONE_CONNECTED.

There are four partition rules for the splitting process.

1) Branch rule: the edges in one branch belong to the same partition. In the undirected graph, a branch is consisted of edges connected by branch nodes. Branch rule assigns nodes on the same signal wires into one partition, which will accelerate the nonlinear convergence.
2) Degree rule: the edges of node with degree two should be assigned to the same partition. This is because the line structure would not cause many non-zero fill-ins and will be propitious to provide DC path in both partitions.

3) Loop rule: the loop will be avoided in both partitions if possible. Edge loops will potentially introduce certain number of non-zero fill-ins.
4) Balance rule: the edges for each node in the graph will be evenly divided into two partitions. Thus, each partition will be much simpler than the original graph.

As an example, Figure 3 illustrates the step by step splitting process on a 6x6 mesh shown in Figure 3(b). The legend used to represent different types of nodes and edges is given in Figure 3(a). Figure 3(c) - 3(g) reveal the stepwise changes of splitting status for all the nodes and edges in BFS partition stage. Figure 3(h) gives the final splitting result after the post-processing step. Both partitions in the final result have a tree/forest structure, which will greatly benefits the LU decomposition.

III. Unconditional Stability Analysis

For the analysis of the error propagation, we can ignore the inputs in the operator splitting formulation (5). We then combine the two half steps and reduce the two equations (5) to a recursive formula

$$X_{(k+1)} = \Lambda X_{(k)} \tag{7}$$

where $\Lambda = (P_2 + S)^{-1}(P_1 - S)(P_1 + S)^{-1}(P_2 - S)$.

In the proof of the convergence, we use the norm $\|x\|_{s^{-1}} = (x^T S^{-1} x)^{1/2}$. Note that matrix S^{-1} is positive definite because matrix S is positive definite and the inverse of a positive definite matrix remains to be positive definite. We first state the theorem of the unconditional stability. The proof of the statement is assisted by the lemmas which follow the theorem.

Theorem 3.1: The operator splitting formula (5) is stable independent of the step size h

Proof: Let $\rho(\Lambda) = max(|\lambda_i(\Lambda)|)$, where $\lambda_i(\Lambda)$ is the i^{th} eigenvalue of matrix Λ. The proposed operator splitting approach is stable if $\rho(\Lambda) <= 1$.

From Lemma 3.4, we have
$\|(P_1 - S)(P_1 + S)^{-1}x\|_{S^{-1}} \le \|x\|_{S^{-1}}$ and
$\|(P_2 - S)(P_2 + S)^{-1}x\|_{S^{-1}} \le \|x\|_{S^{-1}}$.
Let $\rho(\tilde{\Lambda}) = (P_1 - S)(P_1 + S)^{-1}(P_2 - S)(P_2 + S)^{-1}$
From Lemma 3.2 and 3.3, we can deduce: $\rho(\Lambda) = \rho(\tilde{\Lambda}) \le 1$.

Lemma 3.2: $\rho((P_2 + S)^{-1}(P_1 - S)(P_1 + S)^{-1}(P_2 - S)) = \rho((P_1 - S)(P_1 + S)^{-1}(P_2 - S)(P_2 + S)^{-1})$

Proof: We can derive that $\rho(AB) = \rho(BA)$ if matrix A or B is nonsingular. Thus, we prove the lemma by setting $A = (P_2 + S)^{-1}$ and $B = (P_1 - S)(P_1 + S)^{-1}(P_2 - S)$.

Lemma 3.3: Given a real matrix M, if $\|Mx\|_{S^{-1}} \le \gamma\|x\|_{S^{-1}}$ for all real x, then $\rho(M) \le \gamma$.
The proof can be found in [12].

Lemma 3.4: $\|(P_i - S)(P_i + S)^{-1}x\|^2_{S^{-1}} \le \|x\|^2_{S^{-1}}$ for $i \in \{1, 2\}$ and every real vector x.

Proof:
$\|(P_i - S)(P_i + S)^{-1}x\|^2_{S^{-1}} \le \|x\|^2_{S^{-1}}$ is equivalent to $\|(P_i - S)y\|^2_{S^{-1}} \le \|(P_i + S)y\|^2_{S^{-1}}$ where $y = (P_i + S)^{-1}x$

We expand the inequality expression according to the definition of the norm.

$$y^T(P_i^T - S^T)S^{-1}(P_i - S)y \le y^T(P_i^T + S^T)S^{-1}(P_i + S)y \tag{8}$$

We expand the product terms and cancel the common items on the two sides of the inequality. The expression is reduced to:

$$y(\ P_i + P_i^T)y^T \ge 0 \tag{9}$$

which is true since $P_i + P_i^T$ is positive semidefinite for $i \in \{1, 2\}$.

IV. Local Truncation Error and Time Step Control

Though the general operator splitting approach is A-stable, the local truncation error still need to be controlled below the error tolerance in order to ensure the accuracy. By estimating the local truncation error at each time point, we can dynamically adjust the time step to control the local truncation error.

Given the system equation before numerical integration (10),

$$\begin{bmatrix} C & 0 \\ 0 & L \end{bmatrix}\begin{bmatrix} \dot{V}(t) \\ \dot{I}(t) \end{bmatrix} = \begin{bmatrix} -G & A^T \\ -A & -R \end{bmatrix}\begin{bmatrix} V(t) \\ I(t) \end{bmatrix} + U(t) \tag{10}$$

Let $M = \begin{bmatrix} C & 0 \\ 0 & L \end{bmatrix}$, $N = \begin{bmatrix} -G & A^T \\ -A & -R \end{bmatrix}$ and ignore the input vector U, Equation (10) can be simplified as:

$$\begin{array}{rcl} M\dot{X} & = & NX \\ \dot{X} & = & M^{-1}NX \end{array} \tag{11}$$

Here, we regard the system equations above as circuit state equations for the sake of clarity, which implies that matrix M is non-singular in the following derivation.

The exact analytic solution X with time step h can be derived as below:

$$\begin{array}{rcl} X_{n+1} & = & e^{M^{-1}Nh}X_n \\ & = & (1 + M^{-1}Nh + \frac{h^2(M^{-1}N)^2}{2} + \frac{h^3(M^{-1}N)^3}{6} + O(h^4))X_n \end{array} \tag{12}$$

The general operator splitting approach can also be formulated as:

$$\frac{M}{h}(\hat{X}_{n+1} - X_n) = N_1\hat{X}_{n+1} + N_2X_n \tag{13}$$

where $N = N_1 + N_2$, N_1 represents the the partition applied Backward Euler and N_1 denotes the partition applied forward Euler integration method.

The analytic solution of operator splitting approach is derived as below:

$$(\frac{M}{h} - N_1)\hat{X}_{n+1} = (\frac{M}{h} + N_2)X_n \tag{14}$$

$$\hat{X}_{n+1} = [I + hM^{-1}N + h^2M^{-1}N_1M^{-1}N + O(h^3)]X_n \tag{15}$$

The local truncation error is the difference of operator splitting solution $\hat{X}$ and exact solution X:

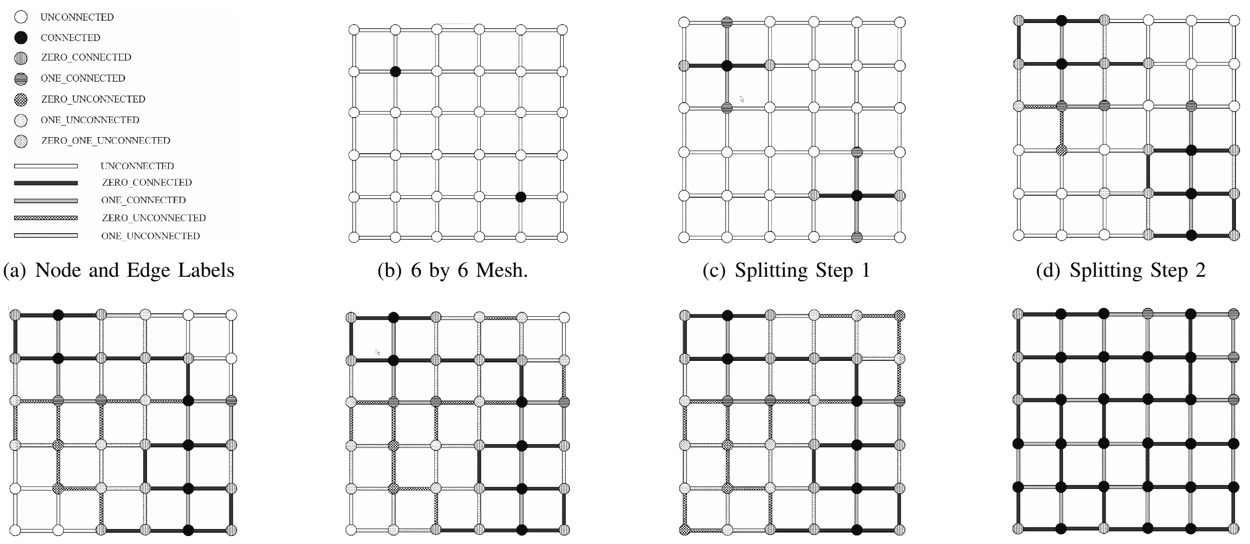
$$LTE = \|h^2M^{-1}(\frac{N}{2} - N_1)\dot{X}_n + O(h^3)\| \tag{16}$$

The local truncation error at each time step should not exceed the error tolerance. If we ignore the high order terms of local truncation error, the time step when forward Euler integration is applied to partition corresponding to N_1 is estimated as:

$$h_1 < \sqrt{\frac{\text{Error Tolerance}}{\|M^{-1}(\frac{N}{2} - N_1)\dot{X}_n\|}} \tag{17}$$

Similarly,the time step when forward Euler integration is applied to partition corresponding to N_2 is estimated as:

$$h_2 < \sqrt{\frac{\text{Error Tolerance}}{\|M^{-1}(\frac{N}{2} - N_2)\dot{X}_n\|}} \tag{18}$$



(a) Node and Edge Labels (b) 6 by 6 Mesh. (c) Splitting Step 1 (d) Splitting Step 2

(e) Splitting Step 3 (f) Splitting Step 4 (g) Splitting Step 5 (h) Final Splitting Result

Fig. 3. 6x6 mesh splitting

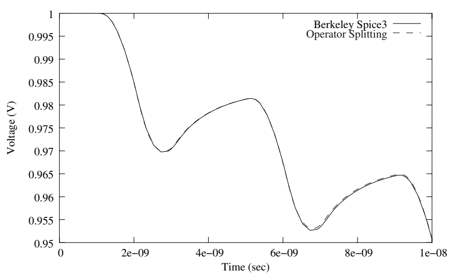


Fig. 4. Transient Response of Circuit3

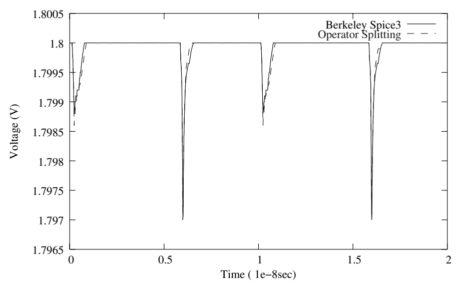


Fig. 5. Voltage Drop of RLC Power and Clock Network Example

And the new time step h is twice of the minimum time step of each partition:

$$h = 2min(h_1, h_2) \tag{19}$$

Our experiment shows that the operator splitting time step size is about 25% of the original SPICE time step size.

V. Experimental Results

The proposed approach is implemented in C programming language. For all circuits presented in this section we compare the proposed approach with Berkeley SPICE3 using BSIM3 models for transistor devices. We did not compare the result with commercial fast simulators because we do not trade any accuracy for speed. Convergence and accuracy are guaranteed. Examples are tested on a Linux machine with 2.6 GHz CPU and 4 Gigabytes memory.

A. Power Network with Nonlinear Current Sinks

We test a number of RLC power networks with size ranging from 11k nodes to 160k nodes. Various transistor gates draw current from the power networks. Those power networks are approximately in mesh structures. The splitting algorithm results in very limited nonzero fill-ins and we observe linear runtime of the proposed method. The CPU runtime is given in Table I. Orders of magnitude speedup (8.1x to 58.2x) against SPICE3 is obtained. The transient waveform circuit3 is given in Figure 4.Since we only replace the LU decomposition procedure inside the SPICE3, other overhead such as device evaluation and dynamic time step control take more than 30% of the total runtime, thus limits the overall speedup.

B. Power and Clock Network

The Power and clock network case contains an RLC power ground network and a two-level H-tree clock. Figure 5 shows the voltage drop at one node of the power network. Transient simulation of 10ns is completed in 649.5 seconds, which is 18.5 times faster than SPICE3 as shown in Table I.

C. Large Power Network Example

This example contains a huge RC power network (0.6 million nodes) with very irregular structure (some nodes have thousands of neighbors). The switching activities that draw current from the power network are modeled as piecewise linear current waveform. Berkeley SPICE3 fails to execute due to the capacity limit. The operator splitting approach finished the transient analysis of 10ns in just 4083 seconds. Figure 6 illustrates the voltage drop of a node on on the power network.

TABLE I
TRANSIENT SIMULATION RUNTIME

Examples	circuit1	circuit2	circuit3	circuit4	Power/Clock	Power Network	1K-cell	10K-cell
#Nodes	11,203	41,321	92,360	160,657	29,100	615,446	10,200	123,600
#Transistors	74	512	1,108	2,130	720	0	6,500	69,000
Simulation Period	10ns	10ns	10ns	10ns	10ns	10ns	20ns	20ns
SPICE3(sec)	602.44	8268.92	39612.32	N/A	12015	N/A	2121	44293
Operator Splitting(sec)	74.64	305.38	681.18	1356.21	649.5	4083.7	415.9	3954.7
Speedup	8.1x	27.1x	58.2x	N/A	18.5x	N/A	5.1x	11.2x

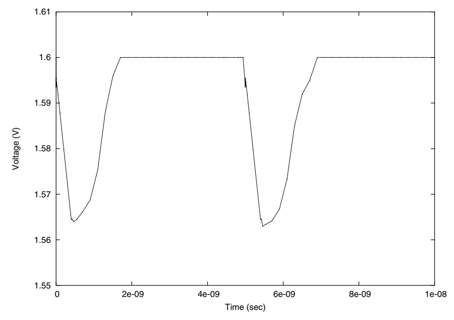


Fig. 6. Voltage drop on the power network

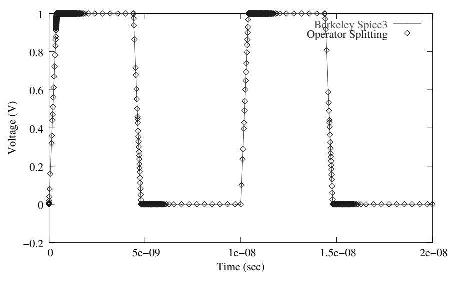


Fig. 7. Transient Waveform of 1K cell Design

D. ASIC designs

Two 1K and 10K cells ASIC designs are tested to demonstrate the proposed approach's ability of handling transistor dominated circuits. The 1k cell circuit has 10,200 nodes and 6,500 transistors. The 10k cell circuit has 123,600 nodes and 69,000 transistors. We assume that ideal power and ground supply is provided in those RC examples. The proposed approach takes 415.9 seconds for 1K cell circuit and 3954.7 seconds for 10K cell circuit to finish 20ns transient simulations. The speedup over SPICE3 is 5.1x and 11.2x for these two examples (Table I). We observe accurate waveform match for both examples. Figure 7 shows the transient waveform of a gate output in the 1K cell design.

VI. CONCLUSION

In this paper, we introduce an general unconditional stable operator splitting method for transistor level transient simulation. Orders of magnitude speedup over Berkeley SPICE3 is observed. Experimental results demonstrate accurate waveform match with SPICE3.

REFERENCES

[1] P. Feldmann, R. W. Freund, "Reduced-Order Modeling of Large Linear Subcircuits via a Block Lanczos Algorithm," DAC, pp. 376–80, 1995.
[2] A. Odabasioglu, M.Celik, and L.T. Pileggi, "PRIMA: Passive Reduced-Order Interconnect Macromodeling Algorithm," ICCAD, 1997.
[3] Z. Zhu, B. Yao, and C.K. Cheng, "Power Network Analysis Using an Adaptive Algebraic Multigrid Approach," DAC, pp. 105-108, 2003
[4] J. N. Kozhaya, S. R. Nassif, F. N. Najm, "Multigrid-like Technique for Power Grid Analysis," ICCAD, pp. 480-487, 2001
[5] T. Chen and C. Chen, "Efficient Large-Scale Power Grid Analysis Based on Preconditioned Krylov-Subspace Iterative Methods," DAC, pp.559-562, 2001.
[6] E. Acar, F. Dartu and L. T. Pileggi, "TETA: Transistor level Waveform Evaluation for Timing Analysis," IEEE Trans. on Computer-Aided Design, Vol. 21, No. 5, May 2002
[7] K.A.Sakallah and S.W.Director,"SAMSON2: An Event Driven VLSI Circuit Simulator," IEEE Trans. on Computer-Aided Design of ICs and Sytems, vol. 4(4), pp. 668-684, October 1985.
[8] www.nassda.com/hsim.html
[9] www.synopsys.com/products/mixedsignal/nanosim
[10] www.cadence.com/products/custom_ic/ultrasim/
[11] Z. Li, C. J. Shi, "SILCA: Fast-Yet-Accurate Time-Domain Simulation of VLSI Circuits with Strong Parasitic Coupling Effects," ICCAD, pp.793-799, 2003
[12] E. L. Wachspress and G. J. Habetler, "An alternating-direction-implicit iteration technique," J. Soc. Ind. and Appl. Math. 8, 403-424(1960)
[13] F. Zheng, Z. Chen, J. Zhang, "Toward the development of a three-dimensional unconditionally stable finite-difference time-domain method," IEEE Tran. Microwave Theory and Techniques, vol 48, No. 9, Sep 2000
[14] Y.-M Lee and C.P. Chen. "Power grid transient simulation in linear time based on transmission-line-modeling alternating-direction-implicit," ICCAD 75-80, 2001.
[15] Y.-M Lee and C.P. Chen. "The power grid transient simulation in linear time on 3D alternating-direction-implicit," Date 2003
[16] T. Namiki and K. Ito, "New FDTD algorithm free from the CFL condition restraint for a 2D-TE wave," IEEE Antennas Propagat. Symp. Dig., pp, 192-195, July 1999.
[17] W. F. Ames, "Numerical Methods for Partial Differential Equations," 2nd ed. New York Academic Press, 1977
[18] W. Sui, "Time-Domain Computer Analysis of Nonlinear Hybrid Systems," CRC Press, 2002.
[19] W. H. Press, S. A. Teukolsky,W. T. Vetterling, "Numerical Recipe in C," 2nd ed. Cambridge University Press, 1992
[20] W. Guo and S. X.-D. Tan, "Circuit level alternating-direction-implicit approach to transient analysis of power distribution networks," Proc. 5th International Conference on ASIC, Oct.2003. pp.246-249.