Power Grid Transient Simulation in Linear Time Based on Transmission-Line-Modeling Alternating-Direction-Implicit Method

Yu-Min Lee, and Charlie Chung-Ping Chen
Department of Electrical and Computer Engineering
University of Wisconsin at Madison
1415 Engineering Drive, Madison, WI 53706
(e-mail: yu-min@cae.wisc.edu, chen@engr.wisc.edu)

Abstract— **The soaring clocking frequency and integration density demand robust and stable power delivery to support tens of millions of transistors switching. To ensure the design quality of power delivery, extensive transient power grid simulations need to be performed during design process. However, the traditional circuit simulation engines are not scaled as well as the complexity of power delivery, as a result, it often takes a long runtime and huge memory requirement to simulate a medium size power grid circuit. In this paper, we develop and present a new efficient transient simulation algorithm for power distribution. The proposed algorithm, TLM-ADI (Transmission-Line-Modeling Alternating-Direction-Implicit), first models the power delivery structure as transmission line mesh structure, then solves the transient MNA matrices by the alternating-direction-implicit method. The proposed algorithm, with** *linear runtime and memory requirement*, **is also** *unconditionally stable* **which ensures that the time-step is not limited by any stability requirement. Extensive experimental results show that the proposed algorithm is not only orders of magnitude faster than SPICE but also extremely memory saving and accurate.**

I. INTRODUCTION

The increase in the complexity of the VLSI chips, and the decrease in the feature size of the chips demand larger grids for power distribution. This causes the design and verification of the power networks have become a challenging task. The inferior designed power distribution network can degrade the circuit performance, noise margin, and the reliability. Since the power grids are rapidly becoming a limiting factor in high performance microprocessors, the ability of analyzing power grids efficiently is a critical requirement to obtain a robust design [1], [2], [3], [4].

There are several sources that cause the degradation of the quality of power delivery systems such as IR drop, Ldi/dt drop, and resonance issues. While IR drop can be simply verified by the DC analysis, the Ldi/dt drop issues need to be analyzed by transient simulation due to the differentiation nature of Ldi/dt drop. In order to ensure the design quality of power delivery, extensive transient simulations are required during the design process. However, due to the complexity of on-chip power grids, it is computationally expensive to simulate all the transistors with power delivery structure. To effectively enhance the simulation speed, it has been proposed to decouple the power delivery structure simulation and transistors simulation [5]. That is, it first simulates transistors to get the drown current waveforms and then performs linear simulation with those currents attached as independent current sources. In this way, we can effectively speed up the simulation since there are fewer elements in both circuits and the linear circuit can be simulated efficiently with only one LU decomposition. However, due to the large size and grid nature of the linear circuit, SPICE [6] does not perform well in this type of system and often takes days to complete the full simulation and needs many giga bytes of memory space. Hence, in order to facilitate the design of large scale power grids, it is crucial to develop an efficient transient simulation engine which is capable of performing the full-chip power grid analysis in a reasonable turn around time.

In this paper, we propose to use the TLM (Transmission Line Modeling) [7] method to perform the time-domain simulation since TLM can capture both the IR drop and Ldi/dt drop. TLM is close related to the FDTD (Finite Difference Time Domain) method, which is one of the most popular and powerful computational electromagnetic techniques in the microwave simulation field [8], [9], [10]. The TLM method differs from FDTD in the sense that it utilizes transmission line cells to model the structure and directly solves the voltage and current quantities while FDTD uses Yee cell structure to obtain electric and magnetic fields. Since voltages and currents are the major focus of the VLSI power delivery analysis, TLM method can be applied directly to perform power delivery transient simulation. The TLM method has been successfully applied to analyze the LC networks by Gwarek [11]. Unfortunately, the time step size is restricted by the minimum grid cell size (Courant stability condition as the standard FDTD method [10]). For example, for the VLSI technology with feature size as 0.1 μm and the dielectric permittivity as 4, the Courant limit is close to 0.47 fs. Thus it needs around 2.1×10^6 time steps to simulate an 1-ns period.

To effectively reduce the stability limit requirement, several unconditional FDTD methods [12], [13], [14] have been proposed and showed good potential in the application of on-chip microwave analysis [12]. However, there is still no unconditionally stable TLM method in the literature to the authors' best knowledge. In this paper, we develop an unconditionally stable algorithm, TLM-ADI, which relaxes the time-step constraint enforced by the Courant stability limit of the traditional TLM method. This new time-stepping scheme is based on an innovative alternating-direction-implicit (ADI) method [15]. With this new method, the upper bound of the time step is only limited by the accuracy requirement rather than stability requirement. Thus, it greatly enhances the computational efficiency due to the reduction of number of time steps. Furthermore, the runtime and memory is linear with $O(N)$ (N, the total number of nodes) since at each time step it only solves around $\sqrt{N}$ *tridiagonal* matrix equations with dimension $\sqrt{N} \times \sqrt{N}$.

The TLM-ADI method with *linear runtime and memory requirement* is also *unconditionally stable* which ensures that time-step is not limited by any stability requirement. Extensive experimental results show that our algorithm is not only orders of magnitude faster than SPICE but also extremely memory saving and accurate.

The remainder of the paper is organized as follows. In Section II, we briefly review the finite-difference algorithm. In Section III, we derive the numerical formulation for our proposed method, and two main features, unconditional stability and linear run time, of the proposed method are dressed. In Section IV, numerical experiments are presented. In Section V, the conclusion of this paper is given.

II. POWER GRID MODELING AND SIMULATION WITH THE FINITE DIFFERENCE METHOD

As illustrated in Figure 1, we use transmission line grids to model the power delivery structure. In each wire segment, we

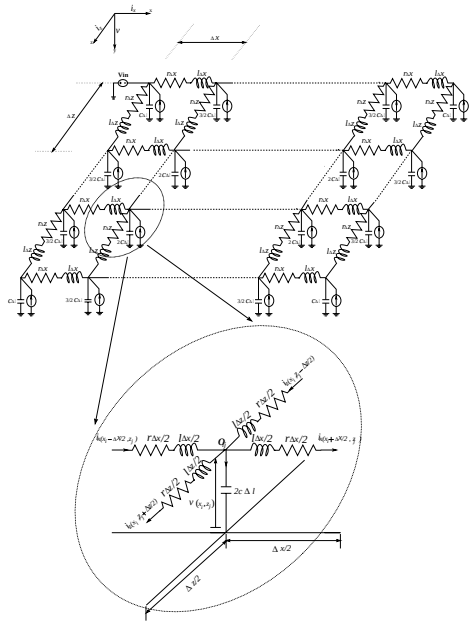

Fig. 1. The transmission line modeling of a power grid structure

use a serially connected resistors and inductors with a ground capacitor to represent it. The parameters r, l, and c are resistance per unit length, inductance per unit length, and capacitance per unit length, respectively. Once the model has been set up, the system matrices are created by the transient nodal analysis. In

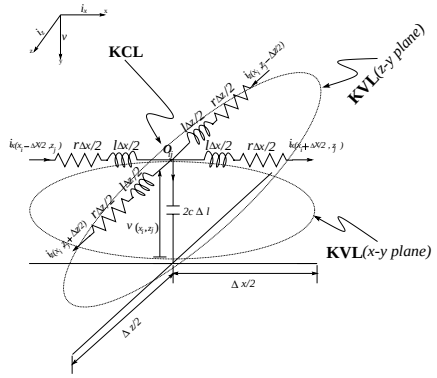

Fig. 2. KCL and KVL for a cell

each cell, as illustrated in Figure 2, by applying KCL at center node, and KVL around the loops of that node in both the $x-y$ and $y-z$ planes, we can write KCL and KVL equations for a node o_{ij} at position (x_i, z_j) as (For simplicity, we ignore independent sources.)

$$\tilde{\mathbf{C}}_{ij}\frac{\partial}{\partial t}\tilde{\mathbf{x}}_{ij} \quad = \quad -\tilde{\mathbf{G}}_{ij}\mathbf{x}_{ij} \tag{1}$$

After assembling the KVL and KCL equations for each cell, the full system equations can be summarized as

$$\tilde{\mathbf{C}}\frac{\partial}{\partial t}\mathbf{x} + \tilde{\mathbf{G}}\mathbf{x} \quad = \quad \tilde{\mathbf{S}}_s(t) \tag{2}$$

where $\mathbf{x}$ is the vector of nodal voltages and currents through the inductors, and $\mathbf{S}_s(t)$ is the vector of voltage and current sources. The system equations are equivalent to the modified nodal analysis (MNA) equations.

Connection Between MNA and Transmission-Line-Equation

Equation (2) can also be expressed in the transmission-line-equation (TLE) formation and hence can be solved by related techniques such as TLM and FDTD methods [8].

For instance, after multiplying the both sides of Equation (1) by $\tilde{\mathbf{C}}_{ij}^{-1}$ and taking limit as $\triangle x \to 0$, $\triangle z \to 0$, and $\triangle l \to 0$ (Here, a uniform internodal distance is assumed, i.e., $\triangle x = \triangle z = \triangle l$.), we can get the general TLE as follows.

$$\frac{\partial v}{\partial t} \quad = \quad \frac{1}{2c}(-\frac{\partial i_x}{\partial x} - \frac{\partial i_z}{\partial z}) \tag{3}$$

$$\frac{\partial i_x}{\partial t} \quad = \quad \frac{1}{l}(-\frac{\partial v}{\partial x} - r i_x) \tag{4}$$

$$\frac{\partial i_z}{\partial t} \quad = \quad \frac{1}{l}(-\frac{\partial v}{\partial z} - r i_z) \tag{5}$$

The basic concepts of finite difference schemes for solving the 2-dimensional TLE are quite simple. First, the domain ($x-z-t$ planes) of the solution is subdivided by a net with a finite number of mesh points ($(x_i, z_j, t_n) = (i\triangle x, j\triangle z, n\triangle t)$ which is represented by "$|^n_{i,j}$" in the rest paper). Then, the derivative at each mesh point is replaced by the finite difference. The finite difference can be done in many ways such as forward-difference, backward-difference, or centered-difference. For example, by using the centered-difference, the $\partial v(x_i, z_j, t)/\partial t_{n+1/2}$ and $\partial i_x(x, z_j, t_{n+1/2})/\partial x_i$ can be approximated as

$$\frac{\partial v(x_i, z_j, t)}{\partial t_{n+1/2}} \quad \approx \quad \frac{-v|^n_{i,j} + v|^{n+1}_{i,j}}{\triangle t} \tag{6}$$

$$\frac{\partial i_x(x, z_j, t_{n+1/2})}{\partial x_i} \quad \approx \quad \frac{-i_x|^{n+1/2}_{i-1/2,j} + i_x|^{n+1/2}_{i+1/2,j}}{\triangle x}. \tag{7}$$

In addition, we can approximate $\partial i_x/\partial t$, $\partial i_z/\partial t$, $\partial i_z/\partial z$, $\partial v/\partial x$, and $\partial v/\partial z$ by the similar way.

Similarly, i_x and i_z can be approximated by the centered-time-average as

$$i_x(x_{i+1/2}, z_j, t_n) \quad \approx \quad \frac{i_x|^{n-1/2}_{i+1/2,j} + i_x|^{n+1/2}_{i+1/2,j}}{2} \tag{8}$$

$$i_z(x_i, z_{j+1/2}, t_n) \quad \approx \quad \frac{i_z|^{n-1/2}_{i,j+1/2} + i_z|^{n+1/2}_{i,j+1/2}}{2}. \tag{9}$$

Plugging the above approximated equations into Equation (3)-(5), (For simplicity, we set $r = 0$ which is the LC circuit.) we get the updating equations [11] as follows.

$$v|^{n+1}_{i,j} = v|^n_{i,j} - \begin{bmatrix} \frac{\triangle t}{2c\triangle x} & \frac{\triangle t}{2c\triangle z} \end{bmatrix} \left(\begin{bmatrix} i_x|^{n+1/2}_{i+1/2,j} \\ i_z|^{n+1/2}_{i,j+1/2} \end{bmatrix} - \begin{bmatrix} i_x|^{n+1/2}_{i-1/2,j} \\ i_z|^{n+1/2}_{i,j-1/2} \end{bmatrix} \right)$$

$$\begin{bmatrix} i_x|^{n+1/2}_{i+1/2,j} \\ i_z|^{n+1/2}_{i,j+1/2} \end{bmatrix} = \begin{bmatrix} i_x|^{n-1/2}_{i+1/2,j} \\ i_z|^{n-1/2}_{i,j+1/2} \end{bmatrix} - \begin{bmatrix} -\frac{\triangle t}{l\triangle x} & \frac{\triangle t}{l\triangle x} & 0 \\ -\frac{\triangle t}{l\triangle z} & 0 & \frac{\triangle t}{l\triangle z} \end{bmatrix} \begin{bmatrix} v|^n_{i,j} \\ v|^n_{i+1,j} \\ v|^n_{i,j+1} \end{bmatrix}$$

The above updating equations are a simple explicit finite difference scheme. They can be easily done since only one unknown variable appears in each difference equation. While it suffers on the Courant stability constraint [10] which is

$$\triangle t \le \frac{1}{\frac{1}{\sqrt{l}c}\sqrt{\frac{1}{(\triangle x)^2} + \frac{1}{(\triangle z)^2}}}. \tag{10}$$

III. TRANSMISSION-LINE-MODELING ALTERNATING-DIRECTION-IMPLICIT METHOD

In this section, we first derive and present the TLM-ADI algorithm for the homogeneous case (r, l, and c are constants). Then its stability is studied analytically. At the end, we generalize the TLM-ADI method to the inhomogeneous case (r, l, and c have different values at different positions), and show that the run time of the proposed algorithm is linear with $O(N)$ (N, the total number of nodes).

A. HOMOGENEOUS CASE

The ADI method is a well known method for solving the partial differential equations (PDE). The main feature of ADI is to sweep directions alternately. Here we derive and present our unconditionally stable algorithm based on the ADI scheme. In contrast to the standard finite difference formulation with only one iteration to advance from the n^{th} to $(n+1)^{th}$ time step, the formulation requires one sub-iteration to advance form n^{th} to $(n+1/2)^{th}$ time step, and a second sub-iteration to advance from $(n+1/2)^{th}$ to $(n+1)^{th}$ time step. For example, considering the KCL equation of (1), every term in the first sub-iteration is effectively discretized at $n+1/4$ as

$$\left.\frac{\partial v}{\partial t}\right|^{n+1/4} = \frac{1}{2c}\left(-\left.\frac{\triangle i_x}{\triangle l}\right|^{n} - \left.\frac{\triangle i_z}{\triangle l}\right|^{n+1/2}\right) \tag{11}$$

where $\triangle i_x = i_x(x+\triangle x/2, z) - i_x(x-\triangle x/2, z)$, and $\triangle i_z = i_z(x, z+\triangle z/2) - i_z(x, z-\triangle z/2)$. Note that the current terms are discretized at time steps n and $n+1/2$, giving an over all effect of $n+1/4$. Thus, the $\triangle i_x/\triangle l$ is evaluated explicitly from known data at time step n, while the $\triangle i_z/\triangle l$ is evaluated implicitly from as-yet known data at time step $n+1/2$. In the second sub-iteration, every term is effectively discretized at $n+3/4$ as

$$\left.\frac{\partial v}{\partial t}\right|^{n+3/4} = \frac{1}{2c}\left(-\left.\frac{\triangle i_x}{\triangle l}\right|^{n+1} - \left.\frac{\triangle i_z}{\triangle l}\right|^{n+1/2}\right). \tag{12}$$

Here, the current terms are discretized at time steps $n+1/2$ and $n+1$, giving an over all effect of $n+3/4$. Thus, the $\triangle i_z/\triangle l$ is evaluated explicitly from known data at time step $n+1/2$, while the $\triangle i_x/\triangle l$ is evaluated implicitly from as-yet known data at time step $n+1$. This ADI scheme can be applied to the KVL equations of(1).

The updating equations are listed below. In this generalized formulation, we have used the conventional semi-implicit formulation to evaluate the voltage and current terms at the appropriate time steps.

Sub-iteration 1: Advance i_x, i_z, and v from time step n to time step $n+1/2$.

$$i_x|^{n+1/2}_{i+1/2,j} = \frac{4l-r\triangle t}{4l+r\triangle t} i_x|^{n}_{i+1/2,j} - \frac{2\triangle t}{\triangle x(4l+r\triangle t)} \begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} v|^{n}_{i,j} \\ v|^{n}_{i+1,j} \end{bmatrix} \tag{13}$$

$$\begin{bmatrix} v|^{n+1/2}_{i,j-1} & v|^{n+1/2}_{i,j} & v|^{n+1/2}_{i,j+1} \end{bmatrix} \begin{bmatrix} \frac{-(\triangle t)^2}{2c\triangle l\triangle z(4l+r\triangle t)} \\ 1+\frac{(\triangle t)^2}{c\triangle l\triangle z(4l+r\triangle t)} \\ \frac{-(\triangle t)^2}{2c\triangle l\triangle z(4l+r\triangle t)} \end{bmatrix} = v|^{n}_{i,j}$$

$$-\frac{\triangle t}{4c\triangle l}(-i_x|^{n}_{i-\frac{1}{2},j} + i_x|^{n}_{i+\frac{1}{2},j}) - \frac{\triangle t(4l-r\triangle t)}{4c\triangle l(4l+r\triangle t)}(-i_z|^{n}_{i,j-\frac{1}{2}} + i_z|^{n}_{i,j+\frac{1}{2}}) \tag{14}$$

$$i_z|^{n+1/2}_{i,j+1/2} = \frac{4l-r\triangle t}{4l+r\triangle t} i_z|^{n}_{i,j+1/2} - \frac{2\triangle t}{\triangle z(4l+r\triangle t)} \begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} v|^{n+1/2}_{i,j} \\ v|^{n+1/2}_{i,j+1} \end{bmatrix} \tag{15}$$

Equation (13) provides an explicit updating expression for i_x since it only depends on known values. Equation (14) provides an implicit updating expression for the voltage component v. Note that the matrix associated with this equation is tridiagonal. It can be efficiently solved by LU decomposition in linear time. After that, we can update i_z by plugging the values of v into Equation (15).

Sub-iteration 2: Advance i_x, i_z, and v from time step $n+1/2$ to time step $n+1$.

$$i_z|^{n+1}_{i,j+1/2} = \frac{4l-r\triangle t}{4l+r\triangle t} i_z|^{n+1/2}_{i,j+1/2} - \frac{2\triangle t}{\triangle z(4l+r\triangle t)} \begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} v|^{n+1/2}_{i,j} \\ v|^{n+1/2}_{i,j+1} \end{bmatrix} \tag{16}$$

$$\begin{bmatrix} v|^{n+1}_{i-1,j} & v|^{n+1}_{i,j} & v|^{n+1}_{i+1,j} \end{bmatrix} \begin{bmatrix} -\frac{(\triangle t)^2}{2c\triangle l\triangle x(4l+r\triangle t)} \\ 1+\frac{(\triangle t)^2}{c\triangle l\triangle x(4l+r\triangle t)} \\ -\frac{(\triangle t)^2}{2c\triangle l\triangle x(4l+r\triangle t)} \end{bmatrix} = v|^{n+1/2}_{i,j}$$

$$-\frac{\triangle t(4l-r\triangle t)}{4c\triangle l(4l+r\triangle t)}(-i_x|^{n+\frac{1}{2}}_{i-\frac{1}{2},j} + i_x|^{n+\frac{1}{2}}_{i+\frac{1}{2},j}) - \frac{\triangle t}{4c\triangle l}(-i_z|^{n+\frac{1}{2}}_{i,j-\frac{1}{2}} + i_z|^{n+\frac{1}{2}}_{i,j+\frac{1}{2}}) \tag{17}$$

$$i_x|^{n+1}_{i+1/2,j} = \frac{4l-r\triangle t}{4l+r\triangle t} i_x|^{n+1/2}_{i+1/2,j} - \frac{2\triangle t}{\triangle x(4l+r\triangle t)} \begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} v|^{n+1}_{i,j} \\ v|^{n+1}_{i+1,j} \end{bmatrix} \tag{18}$$

Equation (16) provides an explicit updating expression for i_z because it only depends on known values. Equation (17) provides an implicit updating expression for the voltage component v. Since the matrix associated with this equation is tridiagonal, it can be efficiently solved by LU decomposition in linear time. Finally, we can update i_x by substituting the values of v into Equation (18).

B. STABILITY ANALYSIS

The general way of verifying the stability of a finite-difference algorithm is to put a sinusoidal traveling wave into the algorithm, and make sure that the propagation gain of this traveling wave is no more than one for all frequencies. By applying the Von Neumann analysis [9] in the LC circuit ($r=0$), we can analytically prove that TLM-ADI method is unconditionally stable.

For each time step n, the instantaneous values of the $\mathtt{i}_x^n$, $\mathtt{i}_z^n$, and $\mathtt{v}_y^n$ in space across the grids are Fourier-transformed into the spatial spectral domain with respect to the x and z coordinates to provide a spectrum of spatial sinusoidal modes. By assuming the wavenumbers k_x, and k_z along the $x-$, and $z-$direction, respectively, these components can be represented as

$$\mathtt{i}_x^n = I_{x_0} e^{-j(k_x x + k_z z)} e^{jwn} \tag{19}$$

$$\mathtt{i}_z^n = I_{z_0} e^{-j(k_x x + k_z z)} e^{jwn} \tag{20}$$

$$\mathtt{v}^n = V_0 e^{-j(k_x x + k_z z)} e^{jwn}. \tag{21}$$

The composite vector in the spatial spectral domain at time-step n is denoted as

$$\mathbf{F}^n = \begin{bmatrix} \mathtt{i}_x^n & \mathtt{i}_z^n & \mathtt{v}^n \end{bmatrix}^T. \tag{22}$$

It can be shown that the Sub-iteration 1 (consisting of the equations (13)-(15) with $r=0$) can be written in the spatial spectral domain in matrix form as

$$\mathbf{F}^{n+1/2} = \mathbf{M_1}\mathbf{F}^n \tag{23}$$

where

$$\mathbf{M_1} = \begin{bmatrix} 1 & 0 & jW_x \\ -\frac{\tilde{W}_x W_z}{Q_z} & \frac{1}{Q_z} & j\frac{W_z}{Q_z} \\ j\frac{\tilde{W}_x}{Q_z} & j\frac{\tilde{W}_z}{Q_z} & \frac{1}{Q_z} \end{bmatrix} \tag{24}$$

$$W_x = \frac{\triangle t}{l\triangle x}\sin\frac{k_x\triangle x}{2}; \qquad \tilde{W}_x = \frac{\triangle t}{2c\triangle l}\sin\frac{k_x\triangle x}{2}$$

$$W_z = \frac{\triangle t}{l\triangle z}\sin\frac{k_z\triangle z}{2}; \qquad \tilde{W}_z = \frac{\triangle t}{2c\triangle l}\sin\frac{k_z\triangle z}{2}$$

$$Q_x = 1 + W_x\tilde{W}_x; \qquad Q_z = 1 + W_z\tilde{W}_z.$$

Similarly, it can be shown that Sub-iteration 2 (consisting of the Equations (16)-(18) with $r=0$) can be written in the spatial spectral domain in matrix form as

$$\mathbf{F}^{n+1} = \mathbf{M_2}\,\mathbf{F}^{n+1/2} \tag{25}$$

where

$$\mathbf{M_2} = \begin{bmatrix} \frac{1}{Q_x} & -\frac{W_x\tilde{W}_z}{Q_x} & j\frac{W_x}{Q_x} \\ 0 & 1 & jW_z \\ j\frac{\tilde{W}_x}{Q_x} & j\frac{\tilde{W}_z}{Q_x} & \frac{1}{Q_x} \end{bmatrix}. \tag{26}$$

Substituting (23) into (25), we get

$$\mathbf{F}^{n+1} = \mathbf{M_2}\mathbf{M_1}\mathbf{F}^n. \tag{27}$$

With the help of package MAPLE^{TM}, we can find the three eigenvalues of the composite matrix $\mathbf{M} = \mathbf{M}_2\mathbf{M}_1$ as follows

$$\lambda = \quad 1, \quad \left(1 - W_x\tilde{W}_x - W_x\tilde{W}_xW_z\tilde{W}_z - W_z\tilde{W}_z \pm 2\sqrt{-W_x\tilde{W}_x - W_z\tilde{W}_z - W_x\tilde{W}_xW_z\tilde{W}_z}\right) / \left(1 + W_x\tilde{W}_x + W_z\tilde{W}_z + W_x\tilde{W}_xW_z\tilde{W}_z\right).$$

By the detail analysis of the above eigenvalues, we are able to prove that the proposed algorithm is unconditionally stable in the following theorem.

Theorem 1: The TLM-ADI algorithm is unconditionally stable.
Proof: The first eigenvalue is unity. For the rest two eigenvalues, the arguments of the square roots in the numerators are all negative numbers. Hence, the square roots are imaginary numbers. By taking the magnitudes of the numerators, we find symbolically that they are exactly the same as denominators. Therefore, the magnitudes of the eigenvalues are unity, regardless of the Δt. If all the eigenvalues of the iterative matrix are less or equal to one, the iterative scheme is unconditionally stable. We conclude that our TLM-ADI algorithm is unconditionally stable. □

C. INHOMOGENEOUS CASE AND LINEAR RUN TIME

Generally, the parameters r, l, and c may have different values at different positions of the circuit. Hence, we extend the TLM-ADI method to handle more general situations, as illustrated in Figure 3. The $R_{x,i\pm1/2,j}$ and $L_{x,i\pm1/2,j}$ are the equivalent resistances and inductances in $x-$ direction for a cell, $R_{z,i,j\pm1/2}$ and $L_{z,i,j\pm1/2}$ are the equivalent resistances and inductances in $z-$ direction for a cell, and $C_{i,j}$ is the equivalent capacitance for a cell. By applying the similar derivation of Equation (13)-(18), we get

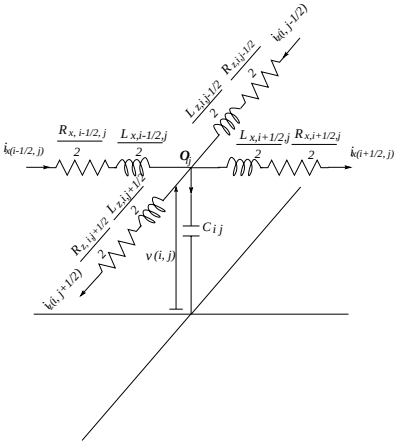

Fig. 3. Inhomogeneous case

the updating equations in matrix forms as follows.

For **Sub-iteration 1**, we divide the set of these N ($N = N_x \times N_z$) nodes by N_x subsets with each one containing N_z points at the $z-$ direction, as illustrated in Figure 4.

$$\mathbf{i}_x|^{n+1/2}_{i+1/2,*} = \mathbf{D}_{xi}\mathbf{i}_x|^n_{i+1/2,*} - \mathbf{D}_{vxi}(\mathbf{v}|^n_{i+1,*} - \mathbf{v}|^n_{i,*}) \quad \forall\, i = 1, \cdots, N_x - 1 \tag{28}$$

$$\mathbf{\Phi}_i\mathbf{v}|^{n+1/2}_{i,*} = \mathbf{v}|^n_{i,*} - \mathbf{D}_{xvi}(\mathbf{i}_x|^n_{i+1/2,*} - \mathbf{i}_x|^n_{i-1/2,*}) - \mathbf{Z}_{zvi}\mathbf{i}_z|^n_{i,*} \quad \forall\, i = 1, \cdots, N_x \tag{29}$$

$$\mathbf{i}_z|^{n+1/2}_{i,*} = \tilde{\mathbf{D}}_{zi}\mathbf{i}_z|^n_{i,*} - \mathbf{Y}_{vzi}\mathbf{v}|^{n+1/2}_{i,*} \quad \forall\, i = 1, \cdots, N_x \tag{30}$$

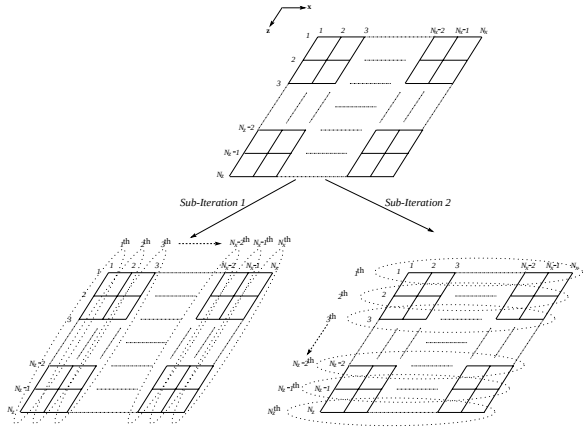

Fig. 4. Mesh points

where $\mathbf{D}_{xi}$, $\mathbf{D}_{vxi}$ and $\mathbf{D}_{xvi}$ are $N_z \times N_z$ diagonal matrices, $\tilde{\mathbf{D}}_{zi}$ is a $(N_z-1)\times(N_z-1)$ diagonal matrix, $\mathbf{Z}_{zvi}$ is a $N_z \times N_z$ *lower-1-banded* matrix (only diagonal and the first lower-subdiagonal terms are nonzeros), $\mathbf{Y}_{vzi}$ is a $(N_z-1)\times(N_z-1)$ *upper-1-banded* matrix (only diagonal and the first upper-subdiagonal terms are nonzeros), $\mathbf{\Phi}_i$ is a $N_z \times N_z$ tridiagonal matrix, $\mathbf{i}_x|^n_{i+1/2,*}$ and $\mathbf{v}|^n_{i,*}$ are $N_z \times 1$ vectors ($n = 1/2, 1, 1+1/2, \cdots$), and $\mathbf{i}_z|^n_{i,*}$ is a $(N_z-1)\times 1$ vector ($n = 1/2, 1, 1+1/2, \cdots$). All definitions of these matrices and vectors are listed in Appendix A.
The computational load for $\mathbf{i}_x|^{n+1/2}_{i+1/2,*}$ is $O(N_z)$ since it only depends on the known values. From Equation (29) and (34), the matrix associated with updating $\mathbf{v}|^{n+1/2}_{i,*}$ is tridiagonal, so the run time is also $O(N_z)$. The work load for $\mathbf{i}_z|^{n+1/2}_{i,*}$ is $O(N_z-1)$ because it depends only on the known values after $\mathbf{v}|^{n+1/2}_{i,*}$ being updated. Therefore, the run time is $O(N_z)$ for each i.
Hence, the computational load for Sub-iteration 1 is $O(N)$.

For **Sub-iteration 2**, we divide the set of these N nodes by N_z subsets with each one containing N_x points at the $x-$ direction, as illustrated in Figure 4.

$$\mathbf{i}_z|^{n+1}_{*,j+1/2} = \mathcal{D}_{zj}\mathbf{i}_z|^{n+1/2}_{*,j+1/2} - \mathcal{D}_{vzj}(\mathbf{v}|^{n+1/2}_{*,j+1} - \mathbf{v}|^{n+1/2}_{*,j}) \quad \forall\, j = 1, \cdots, N_z - 1 \tag{31}$$

$$\mathbf{\Psi}_j\mathbf{v}|^{n+1}_{*,j} = \mathbf{v}|^{n+1/2}_{*,j} - \mathcal{D}_{zvj}(\mathbf{i}_z|^{n+1/2}_{*,j+1/2} - \mathbf{i}_z|^{n+1/2}_{*,j-1/2}) - \mathcal{Z}_{xvj}\mathbf{i}_x|^{n+1/2}_{*,j} \quad \forall\, j = 1, \cdots, N_z \tag{32}$$

$$\mathbf{i}_x|^{n+1}_{*,j} = \tilde{\mathcal{D}}_{xj}\mathbf{i}_x|^{n+1/2}_{*,j} - \mathcal{Y}_{vxj}\mathbf{v}|^{n+1}_{*,j} \quad \forall\, j = 1, \cdots, N_z \tag{33}$$

where $\mathcal{D}_{zj}$, $\mathcal{D}_{vzj}$ and $\mathcal{D}_{zvj}$ are $N_x \times N_x$ diagonal matrices, $\tilde{\mathcal{D}}_{xj}$ is a $(N_x-1)\times(N_x-1)$ diagonal matrix, $\mathcal{Z}_{xvj}$ is a $N_x \times N_x$ *lower-1-banded* matrix, $\mathcal{Y}_{vxj}$ is a $(N_x-1)\times(N_x-1)$ *upper-1-banded* matrix, $\mathbf{\Phi}_i$ is a $N_x \times N_x$ tridiagonal matrix, $\mathbf{i}_x|^n_{*,j}$ and $\mathbf{v}|^n_{*,j}$ are $N_x \times 1$ column vectors ($n = 1/2, 1, 1+1/2, \cdots$), $\mathbf{i}_z|^n_{*,j+1/2}$ is a $(N_x-1)\times 1$ vector ($n = 1/2, 1, 1+1/2, \cdots$). All definitions of these matrices and vectors are listed in Appendix A.
By the similar way, the computational load for Sub-iteration 2 is also $O(N)$.
Now, we can easily prove that TLM-ADI algorithm has a linear run time at each time step.

Theorem 2: The run time of TLM-ADI algorithm is $O(N)$ at each time step, where N is the total number of nodes.
Proof: Two sub-iterations, 1 and 2, need to be performed for each time step. From the above discussion, we know their run times are both $O(N)$. Therefore, the total run time is $O(N)$. □

We summarize our TLM-ADI algorithm in Table I.

TLM-ADI Algorithm

Input= $\mathbf{i}_x|^n$, $\mathbf{i}_z|^n$, $\mathbf{v}|^n$ **Output=** $\mathbf{i}_x|^{n+1}$, $\mathbf{i}_z|^{n+1}$, $\mathbf{v}|^{n+1}$

Begin

Sub-Iteration 1:

$$\mathbf{i}_x|^{n+1/2}_{i+1/2,*} = \mathbf{D}_{xi}\mathbf{i}_x|^n_{i+1/2,*} - \mathbf{D}_{vxi}(\mathbf{v}|^n_{i+1,*} - \mathbf{v}|^n_{i,*}) \quad \forall i = 1, \cdots, N_x - 1$$

$$\mathbf{\Phi}_i\mathbf{v}|^{n+1/2}_{i,*} = \mathbf{v}|^n_{i,*} - \mathbf{D}_{xvi}(\mathbf{i}_x|^n_{i+1/2,*} - \mathbf{i}_x|^n_{i-1/2,*}) - \mathbf{Z}_{zvi}\mathbf{i}_z|^n_{i,*} \quad \forall i = 1, \cdots, N_x$$

$$\mathbf{i}_z|^{n+1/2}_{i,*} = \tilde{\mathbf{D}}_{zi}\mathbf{i}_z|^n_{i,*} - \mathbf{Y}_{vzi}\mathbf{v}|^{n+1/2}_{i,*} \quad \forall i = 1, \cdots, N_x$$

Sub-Iteration 2:

$$\mathbf{i}_z|^{n+1}_{*,j+1/2} = \mathcal{D}_{zj}\mathbf{i}_z|^{n+1/2}_{*,j+1/2} - \mathcal{D}_{vzj}(\mathbf{v}|^{n+1/2}_{*,j+1} - \mathbf{v}|^{n+1/2}_{*,j}) \quad \forall j = 1, \cdots, N_z - 1$$

$$\mathbf{\Psi}_j\mathbf{v}|^{n+1}_{*,j} = \mathbf{v}|^{n+1/2}_{*,j} - \mathcal{D}_{zvj}(\mathbf{i}_z|^{n+1/2}_{*,j+1/2} - \mathbf{i}_z|^{n+1/2}_{*,j-1/2}) - \mathcal{Z}_{xvj}\mathbf{i}_x|^{n+1/2}_{*,j} \quad \forall j = 1, \cdots, N_z$$

$$\mathbf{i}_x|^{n+1}_{*,j} = \tilde{\mathcal{D}}_{xj}\mathbf{i}_x|^{n+1/2}_{*,j} - \mathcal{Y}_{vxj}\mathbf{v}|^{n+1}_{*,j} \quad \forall j = 1, \cdots, N_z$$

End

Note: All symbols are defined in Appendix A.

TABLE I
TLM-ADI ALGORITHM

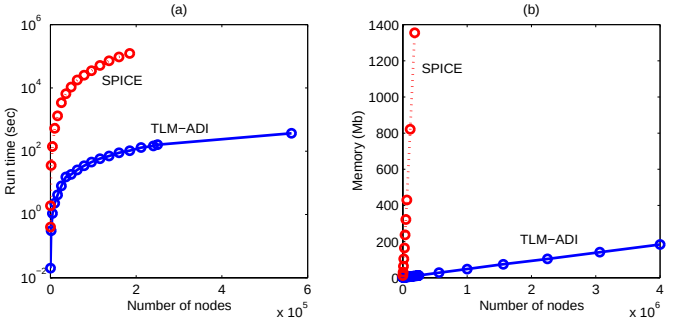

Fig. 5. Compare the (a) run time (b) memory usages between TLM-ADI and SPICE

IV. EXPERIMENTAL RESULTS

In this section, we present several numerical experiments with implementing the TLM-ADI method in C language, and performing on an Alpha workstation with Dual SLOTB 667 MHz Alpha 21264 processors. For simplicity, the experiments are performed on the homogeneous case. We use $r = 0.03\ \Omega/\mu m$, $l = 1.26\ pH/\mu m$, $c = 0.024\ fF/\mu m$, and $\triangle x = \triangle z = 500\ \mu m$ as the common parameters. Numerical results are carried out by using both the TLM-ADI algorithm and the general circuit simulator SPICE. The comparison of run time between TLM-ADI method and SPICE simulator is shown at Figure 5(a) with 1ps-time-step and 500 time steps. Figure 5(a) shows that the TLM-ADI method is over 10^3 times faster than SPICE even for the circuit with only around 180,000 nodes. In Figure 5(b), 6, we show that the run time and memory requirement for TLM-ADI are both linear.

In Figure 7, we examine the accuracy and unconditional stability of the TLM-ADI algorithm by simulating the DC transient response of a RLC circuit with 100 nodes, 1ps-time-step, and 1 volt DC voltage source excitation. Figure 7(a) shows that TLM-ADI produced almost identical waveform as SPICE's at one node. We demonstrate the unconditional stability of TLM-ADI method in Figure 7(b). In this case, the Courant stability constraint is

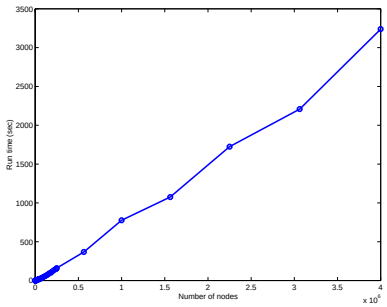

Fig. 6. Linear run time for TLM-ADI

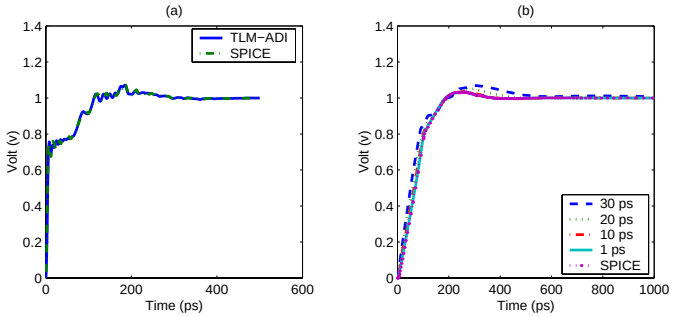

Fig. 7. DC transient response (a) comparison between SPICE and TLM-ADI (b) TLM-ADI with different time steps

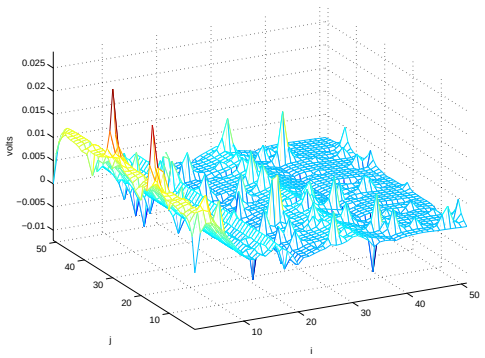

Fig. 8. A snapshot of transient response

1.9442 ps. Figure 7(b) shows that the time step of TLM-ADI is not limited by the stability constraint.

Finally, we use the TLM-ADI algorithm to simulate a 2,601-node power grids in 1-ns clock period and show the result in Figure 8. Each node is attached a time-varying current source to model the drown current.

V. CONCLUSION

An efficient TLM-ADI algorithm for transient power grids simulation is developed. Its unconditional stability and linear run time have been demonstrated. The numerical simulation also shows that the TLM-ADI algorithm not only speeds up orders of magnitude over the SPICE but also cuts down the memory requirement and the results agree very well with the SPICE's.

VI. ACKNOWLEDGMENTS

This work is partially supported by NSF Grant CCR-0093309, Intel Corp., and Avant! Corp..

References

[1] Howard H. Chen, and David D. Ling, Power Supply Noise Analysis Methodology for Deep-Submicron VLSI Chip Design, *DAC*, pp. 638-643, 1997.

[2] Michael K Gowan, Larry L Biro, and Daniel B Jackson, Power Considerations in the Design of the Alpha 21264 Microprocessor, *DAC*, pp. 726-731, 1998.

[3] Adhijit Dharchoudhury, Rajendran Panda, David Blaauw, and Ravi Vaidyanathan, Design and Analysis of Power Distribution Networks in PowerPC^{TM} Microprocessors, *Dac*, pp. 738-743, 1998.

[4] Yi-Min Jiang, and Kwang-Ting Cheng, Analysis of Performance Impact Caused by Power Supply Noise in Deep Submicron Devices, *DAC*, pp. 760-765, 1999.

[5] Min Zhao, Rajendran V. Panda, Sachin S. Sapatnekar, Tim Edwards, Rajat Chaudhry, and David blaauw, Hierarchical Analysis of Power Distribution Networks, *DAC*, pp. 150-155, 2000.

[6] L. W. Nagel, *SPICE2, A Computer Program to Simulate Semiconductor Circuits*, Technical Report ERL-M520, UC-Berkeley, May 1975.

[7] Christos Christopoulos, *The Transmission-Line Modeling Method TLM*, Oxford University Press, 1995.

[8] Matthew N. O. Sadiku, *Numerical Techniques in Electromagentics*, CRC Press, 1992.

[9] John C. Strikwerda, *Finite Difference Schemes and Partial Differential Equations*, Wadsworth & Brooks/Cole Advanced Books Software, 1989.

[10] Allen Taflove, and Susan C Hagness, *Computational Electrodynamics - the finite-difference time-domain method*, Artech House Publishers, 2000.

[11] Wojciech K. Gwarek, *Analysis of Arbitrarily Shaped Two-Dimensional Microwave Circuits by Finite-Difference Time-Domain Method*, IEEE Trans. on Microwave Theory and Techniques, vol. 36, no. 4, pp 738-744, April 1988.

[12] C. P. Chen, Narayanan Murugesan, Tae-Woo Lee, and Susan C. Hagness, FDTD-ADI: An Unconditionally Stable Full Wave Maxwell Equation Solver for VLSI Modeling, *ICCAD*, 2000.

[13] Takefumi Namiki, *A New FDTD Algorithm Based on Alternating-Direction Implicit Method*, IEEE Trans. on Microwave Theory and Techniques, vol. 47, no. 10, pp 2003-2007, 1999.

[14] Fenghua Zheng, Zhizhang Chen, and Jiazong Zhang, *Towards the Development of a Three-Dimensional Unconditionally Stable Finite-Difference Time-Domain Method*, IEEE Trans. on Microwave Theory and Techniques, vol. 48, no. 9, pp 1500-1558, 2000.

[15] D. W. Peaceman, and H. H. Rachford, The numerical solution of parabolic and elliptic differential equations, *J. Soc. Ind. Applicat. Math.*, vol. 3, pp. 28-41, 1955.

Appendix A

Sub-iteration 1:

- The k^{th} diagonal entries of $\mathbf{D}$ matrices, and $\check{\mathbf{D}}_{zi}$.

$$
\begin{aligned}
(\mathbf{D}_{xi})^k &= \frac{4L_{x,i+1/2,k} - R_{x,i+1/2,k}\triangle t}{4L_{x,i+1/2,k} + R_{x,i+1/2,k}\triangle t} \\
(\mathbf{D}_{vxi})^k &= \frac{2\triangle t}{4L_{x,i+1/2,k} + R_{x,i+1/2,k}\triangle t} \\
(\mathbf{D}_{xvi})^k &= \frac{\triangle t}{2C_{i,k}} \\
(\check{\mathbf{D}}_{zi})^k &= \frac{4L_{z,i,k+1/2} - R_{z,i,k+1/2}\triangle t}{4L_{z,i,k+1/2} + R_{z,i,k+1/2}\triangle t}
\end{aligned}
$$

- The k^{th} diagonal $(\mathbf{Z}_{zvi})^k{}_d$, and the k^{th} first-lower-subdiagonal $(\mathbf{Z}_{zvi})^k{}_l$ of $\mathbf{Z}_{zvi}$

$$
\begin{aligned}
(\mathbf{Z}_{zvi})^k{}_d &= \frac{\triangle t}{2C_{i,k}} \frac{4L_{z,i,k+1/2} - R_{z,i,k+1/2}\triangle t}{4L_{z,i,k+1/2} + R_{z,i,k+1/2}\triangle t} \\
(\mathbf{Z}_{zvi})^k{}_l &= -\frac{\triangle t}{2C_{i,k}} \frac{4L_{z,i,k-1/2} - R_{z,i,k-1/2}\triangle t}{4L_{z,i,k-1/2} + R_{z,i,k-1/2}\triangle t}
\end{aligned}
$$

- The k^{th} diagonal $(\mathbf{Y}_{vzi})^k{}_d$, and the k^{th} first-upper-subdiagonal $(\mathbf{Y}_{vzi})^k{}_u$ of $\mathbf{Y}_{vzi}$.

$$
\begin{aligned}
(\mathbf{Y}_{vzi})^k{}_d &= -\frac{2\triangle t}{4L_{z,i,k+1/2} + R_{z,i,k+1/2}\triangle t} \\
(\mathbf{Y}_{vzi})^k{}_u &= \frac{2\triangle t}{4L_{z,i,k+1/2} + R_{z,i,k+1/2}\triangle t}
\end{aligned}
$$

- The tridiagonal matrix $\mathbf{\Phi}_i$

$$
\mathbf{\Phi}_i = \begin{bmatrix}
\beta_{i,1} & \gamma_{i,1} & 0 & \cdots & \cdots & \cdots & 0 \\
\alpha_{i,2} & \beta_{i,2} & \gamma_{i,2} & 0 & \cdots & \cdots & 0 \\
0 & \alpha_{i,3} & \beta_{i,3} & \gamma_{i,3} & 0 & \cdots & 0 \\
 & & & \ddots & \ddots & \ddots & \\
0 & \cdots & 0 & \ddots & \ddots & \ddots & 0 \\
0 & \cdots & \cdots & 0 & \alpha_{i,N_z-1} & \beta_{i,N_z-1} & \gamma_{i,N_z-1} \\
0 & \cdots & \cdots & \cdots & 0 \; \alpha_{i,N_z} & \beta_{i,N_z} &
\end{bmatrix} \tag{34}
$$

$$
\alpha_{i,j} = -\frac{(\triangle t)^2}{C_{i,j}(4L_{z,i,j-1/2} + R_{z,i,j-1/2}\triangle t)}
$$

$$
\begin{aligned}
\beta_{i,j} &= 1 + \frac{(\triangle t)^2}{C_{i,j}}\left(\frac{1}{4L_{z,i,j+1/2} + R_{z,i,j+1/2}\triangle t}\right. \\
&\quad \left. + \frac{1}{4L_{z,i,j-1/2} + R_{z,i,j-1/2}\triangle t}\right) \\
\gamma_{i,j} &= -\frac{(\triangle t)^2}{C_{i,j}(4L_{z,i,j+1/2} + R_{z,i,j+1/2}\triangle t)}
\end{aligned}
$$

- The $\mathbf{i}_x|^n_{i+1/2,*}$, $\mathbf{v}|^n_{i,*}$, and $\mathbf{i}_z|^n_{i,*}$.

$$
\mathbf{i}_x|^n_{i+1/2,*} = \begin{bmatrix} i_x|^n_{i+1/2,1} \\ i_x|^n_{i+1/2,2} \\ \vdots \\ i_x|^n_{i+1/2,N_z} \end{bmatrix} ; \quad \mathbf{v}|^n_{i,*} = \begin{bmatrix} v|^n_{i,1} \\ v|^n_{i,2} \\ \vdots \\ v|^n_{i,N_z} \end{bmatrix}
$$

$$
\mathbf{i}_z|^n_{i,*} = \begin{bmatrix} i_z|^n_{i,1+1/2} \\ i_z|^n_{i,2+1/2} \\ \vdots \\ i_z|^n_{i,N_z-1/2} \end{bmatrix}.
$$

Sub-iteration 2:

- The k^{th} diagonal entries of $\mathcal{D}$ matrices, and $\check{\mathcal{D}}_{xj}$.

$$
\begin{aligned}
(\mathcal{D}_{zj})^k &= \frac{4L_{z,k,j+1/2} - R_{z,k,j+1/2}\triangle t}{4L_{z,k,j+1/2} + R_{z,k,j+1/2}\triangle t} \\
(\mathcal{D}_{vzj})^k &= \frac{2\triangle t}{4L_{z,k,j+1/2} + R_{z,k,j+1/2}\triangle t} \\
(\mathcal{D}_{zvj})^k &= \frac{\triangle t}{2C_{k,j}} \\
(\check{\mathcal{D}}_{xj})^k &= \frac{4L_{x,k+1/2,j} - R_{x,k+1/2,j}\triangle t}{4L_{x,k+1/2,j} + R_{x,k+1/2,j}\triangle t}
\end{aligned}
$$

- The k^{th} diagonal $(\mathcal{Z}_{xvj})^k{}_d$, and the k^{th} first-lower-subdiagonal $(\mathcal{Z}_{xvj})^k{}_l$ of $\mathcal{Z}_{xvj}$

$$
\begin{aligned}
(\mathcal{Z}_{xvj})^k{}_d &= \frac{\triangle t}{2C_{k,j}} \frac{4L_{x,k+1/2,j} - R_{x,k+1/2,j}\triangle t}{4L_{x,k+1/2,j} + R_{x,k+1/2,j}\triangle t} \\
(\mathcal{Z}_{xvj})^k{}_l &= -\frac{\triangle t}{2C_{k,j}} \frac{4L_{x,k-1/2,j} - R_{x,k-1/2,j}\triangle t}{4L_{x,k-1/2,j} + R_{x,k-1/2,j}\triangle t}
\end{aligned}
$$

- The k^{th} diagonal $(\mathcal{Y}_{vxj})^k{}_d$, and the k^{th} first-upper-subdiagonal $(\mathcal{Y}_{vxj})^k{}_u$ of $\mathcal{Y}_{vxj}$.

$$
\begin{aligned}
(\mathcal{Y}_{vxj})^k{}_d &= -\frac{2\triangle t}{4L_{x,k+1/2,j} + R_{x,k+1/2,j}\triangle t} \\
(\mathcal{Y}_{vxj})^k{}_u &= \frac{2\triangle t}{4L_{x,k+1/2,j} + R_{x,k+1/2,j}\triangle t}
\end{aligned}
$$

- The tridiagonal matrix $\mathbf{\Psi}_j$

$$
\mathbf{\Psi}_j = \begin{bmatrix}
\eta_{1,j} & \zeta_{1,j} & 0 & \cdots & \cdots & \cdots & 0 \\
\kappa_{2,j} & \eta_{2,j} & \zeta_{2,j} & 0 & \cdots & \cdots & 0 \\
0 & \kappa_{3,j} & \eta_{3,j} & \zeta_{3,j} & 0 & \cdots & 0 \\
 & & & \ddots & \ddots & \ddots & \\
0 & \cdots & 0 & \ddots & \ddots & \ddots & 0 \\
0 & \cdots & \cdots & 0 & \kappa_{N_x-1,j} & \eta_{N_x-1,j} & \zeta_{N_x-1,j} \\
0 & \cdots & \cdots & \cdots & 0 \; \kappa_{N_x,j} & \eta_{N_x,j} &
\end{bmatrix} \tag{35}
$$

$$
\begin{aligned}
\kappa_{i,j} &= -\frac{(\triangle t)^2}{C_{i,j}(4L_{x,i-1/2,j} + R_{x,i-1/2,j}\triangle t)} \\
\eta_{i,j} &= 1 + \frac{(\triangle t)^2}{C_{i,j}}\left(\frac{1}{4L_{x,i+1/2,j} + R_{x,i+1/2,j}\triangle t}\right. \\
&\quad \left. + \frac{1}{4L_{x,i-1/2,j} + R_{x,i-1/2,j}\triangle t}\right) \\
\zeta_{i,j} &= -\frac{(\triangle t)^2}{C_{i,j}(4L_{x,i+1/2,j} + R_{x,i+1/2,j}\triangle t)}
\end{aligned}
$$

- The $\mathbf{i}_x|^n_{*,j}$, $\mathbf{v}|^n_{*,j}$, and $\mathbf{i}_z|^n_{*,j+1/2}$

$$
\mathbf{i}_x|^n_{*,j} = \begin{bmatrix} i_x|^n_{1+1/2,j} \\ i_x|^n_{2+1/2,j} \\ \vdots \\ i_x|^n_{N_x-1/2,j} \end{bmatrix} ; \quad \mathbf{v}|^n_{*,j} = \begin{bmatrix} v|^n_{1,j} \\ v|^n_{2,j} \\ \vdots \\ v|^n_{N_x,j} \end{bmatrix}
$$

$$
\mathbf{i}_z|^n_{*,j+1/2} = \begin{bmatrix} i_z|^n_{1,j+1/2} \\ i_z|^n_{2,j+1/2} \\ \vdots \\ i_z|^n_{N_x,j+1/2} \end{bmatrix}.
$$