A Novel Linking-Domain Extraction Decomposition Method for Parallel Electromagnetic Transient Simulation of Large-Scale AC/DC Networks

Tong Duan, *Student Member, IEEE*, and Venkata Dinavahi, *Fellow, IEEE*

***Abstract*—Domain decomposition of the network conductance matrix is one of the efficient approaches to solve large-scale networks in parallel, wherein the most commonly-used non-iterative method is the Schur complement (SC) method. However, the SC method could not obtain the network conductance matrix inversion directly, and the computational cost will increase fast when the overlapping domain expands. In this work, a novel Linking-Domain Extraction (LDE) based decomposition method is proposed, in which the network matrix is expressed as the sum of a linking-domain matrix (LDM) and a diagonal block matrix (DBM) composed of multiple block matrices in diagonal. Through mathematical analysis over LDM, one *lemma* about the nature of LDM and its proof are proposed. Based on this *lemma*, the general formulation of the inverse matrix of the sum of LDM and DBM can be found using the Woodbury matrix identity, and based on the formulation the network matrix inversion can be directly computed in parallel to significantly accelerate the matrix inversion process. Test systems were implemented on both the FPGA and GPU parallel architectures, and the simulation results and speed-ups over the SC method and Gauss-Jordan elimination demonstrate the validity and efficiency of the proposed LDE method.**

***Index Terms*—Circuit simulation, electromagnetic transients, field programmable gate arrays, graphics processors, inverse matrix calculation, network decomposition, parallel processing.**

I. Introduction

ELECTROMAGNETIC transient (EMT) simulation is a paramount tool to study the electrical system's behavior and reproduce the transient waveforms prior to manufacturing and deployment [1]. Typically, in EMT simulation a specific circuit can be formulated as a set of linear equations or differential equations (when containing dynamic elements such as inductors and capacitors) of unknown state variables [2], [3]; solving these equations requires discretizing them (i.e., modeling the dynamic circuit elements in discrete time) and solving the resulting linear algebraic equations.

Manuscript received October 8, 2019; revised January 30, 2020 and May 1, 2020; accepted May 26, 2020. Date of publication May 28, 2020; date of current version March 24, 2021. This work was supported in part by the Natural Science and Engineering Research Council of Canada (NSERC) and in part by the China Scholarship Council (CSC). Paper no. TPWRD-01150-2019. *(Corresponding author: Tong Duan.)*

The authors are with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 2V4, Canada (e-mail: tduan@ualberta.ca; dinavahi@ualberta.ca).

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TPWRD.2020.2998397

Most EMT programs use Gaussian Elimination or LU factorization method to solve the linear algebraic equations. However, the simulation process slows down significantly when the circuit scale expands, especially for large-scale AC/DC grids where the MMC converters composed of hundreds of submodules generate a large matrix. The essential problem is that the computational complexity of direct matrix inversion or solving linear algebraic equations using other related algorithms such as the Gauss-Jordan elimination method is $O(N^3)$, which result in an extremely high latency for large-scale networks. To deal with the complexity of simulating large-scale systems, network domain decomposition [4] is a commonly-used method that splits the large network into small subsystems to simulate them in parallel. One main challenge of using domain decomposition is how to uncouple the inter-connected subsystems, which leads to two representative categories of decomposition methods [5]: overlapping domain decomposition and non-overlapping domain decomposition. In overlapping domain decomposition, the basic logic is to allocate the overlapping domain between two connected subsystems (multiple subsystems have the same procedure) into both subsystems, so that each subsystem can compute the values of the overlapping domain simultaneously. However, to obtain the correct values of the overlapping domain, data exchange and iteration are required to guarantee the difference of results between the two subsystems are smaller than a predetermined threshold. In addition, when the number of decomposed subsystems increases, the convergence time will become much longer, and thus the overlapping domain decomposition method is not the scope of this work.

In non-overlapping domain decomposition, the decomposed subsystems have no overlapping domains thus they could be simulated in parallel while not requiring iteration to synchronize the connected subsystems. For the non-overlapping domain decomposition, the most widely-used methods are the transmission line modeling (TLM) [6], [7], latency insertion method (LIM) [8], [9] and Schur Complement (SC) method [5]. The TLM and LIM methods are latency-based decomposition methods, which leverage the transmission latency between two ends of a line or the latency produced by the LC circuits to decompose the network. Both methods need to consider the simulation time-step size. For example, if the transmission latency of a line is smaller than the time-step size, then the two ends of the line could not

be calculated simultaneously. For networks where transmission lines do not exist, the SC method is most commonly used [10], [11], which is a matrix-based decomposition method. It moves all the overlapping area in the network conductance matrix to the bottom right and the remaining parts are diagonal block matrices that can be handled in parallel. However, the SC method could not obtain the network matrix inversion directly, thus the corresponding procedure need to be executed in each time-step. In addition, the efficiency reduces quickly when the overlapping domain expands.

Different from the SC method that ignores the special features of the overlapping part between decomposed block matrices, in this work, a novel linking-domain extraction (LDE) based decomposition method is proposed. In LDE, the linking-domain matrix (LDM) is extracted from the original network conductance matrix, and the remaining matrix is a diagonal block matrix (DBM) composed of multiple block matrices in diagonal. Through mathematical analysis over LDM, an important nature of LDM is found and formulated in *Lemma 1:* the linking domain matrix can be transformed from a diagonal matrix through a transformation matrix that only contains elements with 0, 1, -1 values. Based on this *lemma*, the general formulation of the matrix inversion of the sum of LDM and DBM can be found via the Woodbury matrix identity [12] in linear algebra, by which the network matrix inversion can be computed in parallel. If the network conductance matrix does not change during the simulation, the LDE method can compute the matrix inversion in advance to significantly accelerate the simulation process; if the conductance matrix changes, the LDE method may also be more suitable than SC method in many cases. Compared to the LU factorization method that can also benefit from a constant conductance matrix, computing the matrix inversion makes a certain sense because after the L, U matrices are computed, in the subsequent time slots solving $\mathbf{L}\mathbf{U}\boldsymbol{x} = \boldsymbol{b}$ (assume solving $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$) also occupies a longer time than only computing $\mathbf{A}^{-1}\boldsymbol{b}$ in parallel.

The paper is organized as follows: Section II introduces the traditional SC method. Section III presents the proposed LDE method, including the derivation of the important *lemma* and the general formulation of the network matrix inversion, as well as the advantages and limitations of the LDE method. In Section IV, large-scale test systems are simulated on both FPGA and GPU parallel architectures to verify the accuracy and efficiency of the proposed LDE method. Finally in Section V conclusions are drawn.

II. Schur Complement Method

Once the network equations have been gathered, the solution of a linear network reduces to a solution of the linear algebraic system:

$$\mathbf{G}\,\boldsymbol{v} = \boldsymbol{i}^{eq} \tag{1}$$

The SC method is a non-overlapping method, which means the network conductance matrix $\mathbf{G}$ is decomposed into m uncoupled small block matrices ($\hat{\mathbf{G}}_1, \hat{\mathbf{G}}_2, \ldots, \hat{\mathbf{G}}_m$). The overlapping domains between block matrices are moved to the overlapping domain matrix $\mathbf{D}_t$, as shown in Fig. 1(a). The nodes located in $\mathbf{D}_t$ actually represent the interface nodes used to connect decoupled subsystems. Applying the block matrix multiplication, the following equations can be obtained:

$$\begin{bmatrix} \hat{G}_1 & 0 & \cdots & 0 & \mathbf{E}_1 \\ 0 & \hat{\mathbf{G}}_2 & \cdots & 0 & \mathbf{E}_2 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & \hat{\mathbf{G}}_m & \mathbf{E}_m \\ \mathbf{F}_1 & \mathbf{F}_2 & \cdots & \mathbf{F}_m & \mathbf{D}_t \end{bmatrix} \begin{pmatrix} \boldsymbol{v}_{g1} \\ \boldsymbol{v}_{g2} \\ \vdots \\ \boldsymbol{v}_{gm} \\ \boldsymbol{v}_t \end{pmatrix} = \begin{pmatrix} \boldsymbol{i}_{\mathbf{g1}} \\ \boldsymbol{i}_{g2} \\ \vdots \\ \boldsymbol{i}_{gm} \\ \boldsymbol{i}_t \end{pmatrix} \tag{2}$$

$$(\mathbf{D}_t - \mathbf{F}\hat{\mathbf{G}}_d^{-1}\mathbf{E})\,\boldsymbol{v}_t = \boldsymbol{i}_t - \mathbf{F}\hat{\mathbf{G}}_d^{-1}\boldsymbol{i}_g \tag{3}$$

$$\hat{\mathbf{G}}_d\,\boldsymbol{v}_g = \boldsymbol{i}_g - \mathbf{E}\,\boldsymbol{v}_t \tag{4}$$

where $\hat{\mathbf{G}}_d$, $\mathbf{F}$ and $\mathbf{E}$ are the combination of the corresponding block matrices, and $\boldsymbol{v}_g, \boldsymbol{i}_g$ are the combination of the corresponding node voltages and current injections.

The computation process is executed as follows:

1) compute $\hat{\mathbf{G}}_d^{-1}$ and $\hat{\mathbf{G}}_d^{-1}\boldsymbol{i}_g$;
2) compute $\mathbf{F}\hat{\mathbf{G}}_d^{-1}\boldsymbol{i}_g$ and $\mathbf{F}\hat{\mathbf{G}}_d^{-1}\mathbf{E}$;
3) solve equation (3) to get $\boldsymbol{v}_t$;
4) solve $\boldsymbol{v}_g = \hat{\mathbf{G}}_d^{-1}\boldsymbol{i}_g - \hat{\mathbf{G}}_d^{-1}\mathbf{E}\,\boldsymbol{v}_t$.

Parallel computing can be exploited based on the computation that involves the diagonal block matrix $\hat{\mathbf{G}}_d$, such as computing $\hat{\mathbf{G}}_d^{-1}$, $\mathbf{F}\hat{\mathbf{G}}_d^{-1}\mathbf{E}$ and $\boldsymbol{v}_g\,(=\hat{\mathbf{G}}_d^{-1}\boldsymbol{i}_g - \hat{\mathbf{G}}_d^{-1}\mathbf{E}\,\boldsymbol{v}_t)$. However, it can be observed that the SC method could not obtain $\mathbf{G}^{-1}$ directly due to the involvement of the current injections even though $\hat{\mathbf{G}}_d^{-1}$ and the other matrices could be computed in advance for linear circuits, and thus in each time-step the four processing steps could not be avoided. In addition, when the number of decomposed matrices increases, the amount of interface nodes will increase quickly, which significantly influences the overall performance due to the large computational effort in solving $\boldsymbol{v}_t$.

III. Proposed Linking-Domain Extraction Based Decomposition Method

Focusing on the overlapping domain between subsystems, the linking-domain matrix is defined and extracted from the original network matrix. Through mathematical analysis over the linking-domain matrix, an important *lemma* is put forward, based on which the inverse matrix of the network matrix can be computed in parallel.

A. LDE Matrix Decomposition

Different from the logic of the SC method that operates on the decomposition of the original conductance matrix, the proposed LDE method decomposes the original matrix into two separate matrices: the diagonal block matrix $\mathbf{G}_d$ and the linking-domain matrix $\mathbf{L}$.

$$\mathbf{G} = \mathbf{G}_d + \mathbf{L} \tag{5}$$

As shown in Fig. 1(b), the sizes of the decomposed block matrices ($\mathbf{G}_1, \mathbf{G}_2, \ldots, \mathbf{G}_m$) are larger than using the SC method,

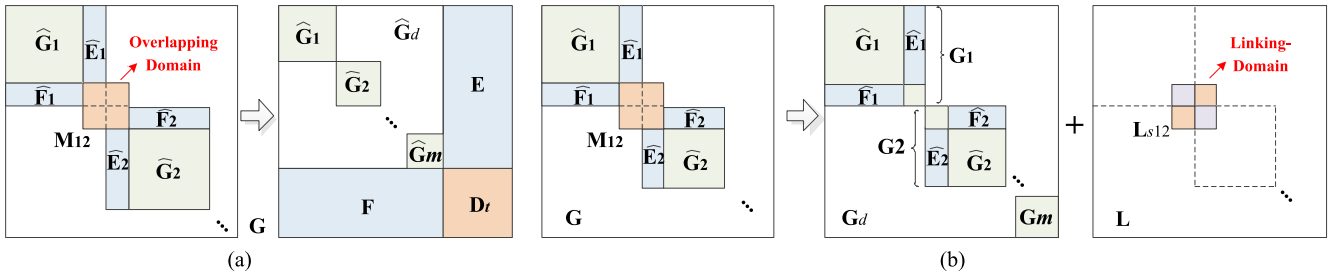


Fig. 1. Example of matrix decomposition: (a) Schur decomposition method; (b) Proposed LDE method.

because in the SC method all the overlapping areas and their corresponding rows and columns are removed from the block matrices, while in the LDE method only the linking-domain matrix is extracted from the original matrix and the size of each block matrix does not decrease. Note that the linking-domain has a different meaning from the overlapping domain, because the elements in the overlapping domain actually are the sum of the corresponding values in the linking-domain matrix and the block matrices, as marked in Fig. 1.

The construction of $\mathbf{L}$ matrix is as follows: let the linking-domain matrix contain a small matrix with non-zero diagonal elements ($\mathbf{L}_s$) and the other all-zero matrices, as illustrated in Fig. 1(b). The location of $\mathbf{L}_s$ is the same as the overlapping domain in $\mathbf{G}$ and the size of $\mathbf{L}_s$ is $(n_1+n_2)\times(n_1+n_2)$, where n_1 is the number of interface nodes belonging to the first subsystem and n_2 is the number of interface nodes belonging to the second subsystem (taking two decomposed subsystems as an example). $\mathbf{L}_s$ can be regarded as the combination of four block matrices: the top left block matrix and the bottom right block matrix are both diagonal matrices, and the top right and bottom left block matrices are transpose of each other. The top right $(n_1\times n_2)$ and bottom left $(n_2\times n_1)$ block matrices are the same as those of the overlapping domain matrix, and after the two parts are assigned, the diagonal elements in the top left $(n_1\times n_1)$ and bottom right $(n_2\times n_2)$ block matrix are determined as follows:

$$\mathbf{L}_s=\begin{bmatrix}\Sigma_{1,1} & 0 & \cdots & 0 & \sigma_{1,1} & \cdots & \sigma_{1,n_2}\\ 0 & \Sigma_{1,2} & \cdots & 0 & \sigma_{2,1} & \cdots & \sigma_{2,n_2}\\ \vdots & \vdots & & \vdots & \vdots & & \vdots\\ 0 & 0 & \cdots & \Sigma_{1,n_1} & \sigma_{n_1,1} & \cdots & \sigma_{n_1,n_2}\\ \sigma_{1,1} & \sigma_{2,1} & \cdots & \sigma_{n_1,1} & \Sigma_{2,1} & \cdots & 0\\ \vdots & \vdots & & \vdots & \vdots & & \vdots\\ \sigma_{1,n_2} & \sigma_{2,n_2} & \cdots & \sigma_{n_1,n_2} & 0 & \cdots & \Sigma_{2,n_2}\end{bmatrix} \tag{6}$$

$$\Sigma_{1,i}=-\sum_{j=1}^{n_2}\sigma_{i,j},\quad \forall 1\leqslant i\leqslant n_1 \tag{7}$$

$$\Sigma_{2,i}=-\sum_{j=1}^{n_1}\sigma_{j,i},\quad \forall 1\leqslant i\leqslant n_2 \tag{8}$$

Since the inverse of the diagonal block matrix $\mathbf{G}_d$ can be calculated in parallel, if the inverse matrix of $\mathbf{G}=\mathbf{G}_d+\mathbf{L}$ can also be calculated in parallel, then the simulation process can be accelerated significantly. Therefore, the core task of LDE method is to find a general formulation of the inverse of $\mathbf{G}$. Fortunately, taking advantage of the special features of linking-domain matrix, the relationship between $\mathbf{G}_d^{-1}$ and $\mathbf{G}^{-1}$ can be found.

B. Mathematical Analysis Over LDM

For a common linear network, the linking-domain matrix $\mathbf{L}$ $(N\times N)$ has some specific characteristics. For example, $\mathbf{L}$ is a symmetric matrix, and the rank of $\mathbf{L}$ is usually smaller than N (except that all of the nodes are interface nodes). But more importantly, it can be observed that the sum of each row (or column) of $\mathbf{L}$ is always equal to 0, that is:

$$\sum_{j=1}^{N}\mathbf{L}_{i,j}=0,\quad \forall 1\leqslant i\leqslant N \tag{9}$$

$$\sum_{i=1}^{N}\mathbf{L}_{i,j}=0,\quad \forall 1\leqslant j\leqslant N \tag{10}$$

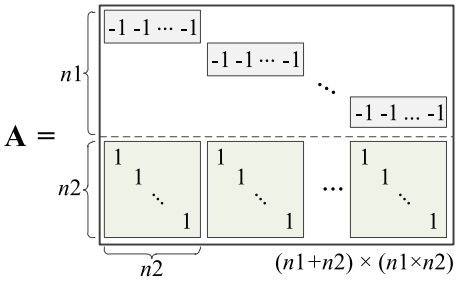
Based on this observation, Lemma 1 that decomposes the linking-domain matrix into the multiplication of three matrices is proposed.

Lemma 1: The linking-domain matrix $\mathbf{L}$ can be expressed as $\mathbf{L}=\mathbf{C}\Lambda\mathbf{C}^T$, where Λ is a diagonal matrix with non-negative real numbers on the diagonal, and the transform matrix $\mathbf{C}$ is a rectangular matrix of which the element values are only equal to 1, -1, or 0.

Proof: To simplify the proving process, we first start with the two connected subsystems and then extend it to multiple subsystems.

Step 1: prove that $\mathbf{L}_s=\mathbf{A}\Delta\mathbf{A}^T$, where the size of Δ is $(n_1\times n_2)\times(n_1\times n_2)$ and the size of transform matrix $\mathbf{A}$ is $(n_1+n_2)\times(n_1\times n_2)$. More specifically, Δ and $\mathbf{A}$ can be expressed as:

$$\Delta=\begin{bmatrix}-\sigma_{1,1} & 0 & \cdots & 0 & 0\\ 0 & -\sigma_{1,2} & \cdots & 0 & 0\\ \vdots & \vdots & & \vdots & \vdots\\ 0 & 0 & \cdots & -\sigma_{n_1,n_2-1} & 0\\ 0 & 0 & 0 & \cdots & -\sigma_{n_1,n_2}\end{bmatrix} \tag{11}$$



(12)

The elements of the diagonal matrix Δ are exactly the negative values of the elements in the top right block matrix of $\mathbf{L}_s$. Note that the blank area in transform matrix $\mathbf{A}$ is filled with zero. The transform matrix $\mathbf{A}$ shown in (12) has some important features:

$$\mathbf{A}^{(i,r)} = -1,\ \forall 1 \leqslant i \leqslant n_1, (i-1)n_2 < r \leqslant i \times n_2 \tag{13}$$

$$\mathbf{A}^{(i,r)} = 1,\ \forall n_1 < i \leqslant n_1 + n_2,$$
$$r = n_2 \times \{0, 1, \ldots, n_1 - 1\} + (i - n_1) \tag{14}$$

$$\mathbf{A}^{(i,r)}\mathbf{A}^{(j,r)} = 0,\ \forall 1 \leqslant r \leqslant (n_1 \times n_2),$$
$$\forall 1 \leqslant i \neq j \leqslant n_1 \text{ or } n_1 < i \neq j \leqslant n_1 + n_2 \tag{15}$$

$$\mathbf{A}^{(i,r)}\mathbf{A}^{(j,r)} = -1,\ \forall 1 \leqslant i \leqslant n_1,$$
$$(i-1)n_2 < r \leqslant i \times n_2;\ j = r - n_2 \times (i-1) + n_1 \tag{16}$$

Then the elements in $\mathbf{L}_s$ can be expressed as:

$$\mathbf{L}_s^{(i,j)} = -\sum_{r=1}^{n_1 \times n_2} \mathbf{A}^{(i,r)}\mathbf{A}^{(j,r)}\sigma_{p,q},\ \forall 1 \leqslant i, j \leqslant (n_1 + n_2) \tag{17}$$

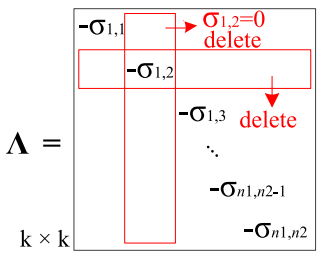
$$p = ceil(r/n_2),\ \ q = r - n_2 \times (p-1) \tag{18}$$

where the function $ceil(r, n_2)$ rounds up the division of r by n_2 to an integer, which means the maximum values of p and q are n_1 and n_2 respectively. For the diagonal elements, the correctness of the matrix elements values can be verified based on (9), (10), (13), (14). For example, if $i = 2$, then $\mathbf{L}_s^{(2,2)} = \Sigma_{1,2} = -(\sigma_{2,1} + \sigma_{2,2} + \cdots + \sigma_{2,n_2})$, which matches with (17). For the other non-diagonal elements in $\mathbf{L}_s$, the correctness can also be verified based on (15), (16).

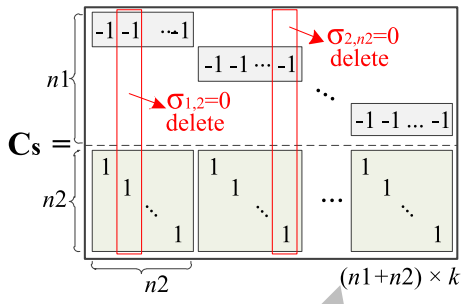
Step 2: delete the diagonal elements with zero values in Δ to generate Λ, and delete their corresponding columns in $\mathbf{A}$ to generate $\mathbf{C}_s$, then:

$$\mathbf{L}_s = \mathbf{C}_s\Lambda\mathbf{C}_s^T \tag{19}$$

where the sizes of Λ and $\mathbf{C}_s$ are $k \times k$ and $(n_1 + n_2) \times k$ respectively:



(20)



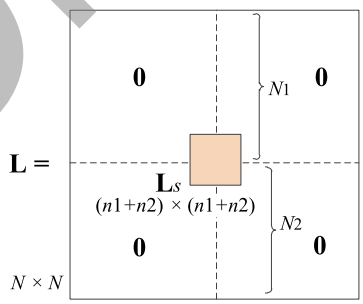
(21)

Typically, the new transform matrix size k is much smaller than $n_1 \times n_2$, because most elements in the $(\sigma_{1,1}, \ldots, \sigma_{n_1,n_2})$ are zero if the connection is sparse. In fact, $k = n_1 \times n_2$ is only valid for the extreme case that all the interface nodes are connected with each other via a conductance.

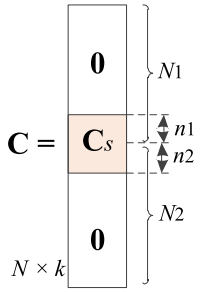
The equation (19) can be verified in the same way as in Step 1, because all the items in (17) corresponding to the deleted elements are equal to zero, and thus deleting these elements does not influence the result of $\mathbf{L}_s$.

Step 3: extend the small matrices $\mathbf{L}_s$ and $\mathbf{C}_s$ into bigger matrices $\mathbf{L}$ $(N \times N)$ and $\mathbf{C}$ $(N \times k)$ with zero elements, then:

$$\mathbf{L} = \mathbf{C}\Lambda\mathbf{C}^T \tag{22}$$



(23)



(24)

This step just adds some block matrices with all-zero elements to extend $\mathbf{L}_s$ and $\mathbf{C}_s$, which actually increases the size of the resulting matrix $\mathbf{C}\Lambda\mathbf{C}^T$ while maintaining the correct values at the non-zero area in $\mathbf{L}$.

The above three steps actually provides the proof of Lemma 1 in the situation of two decomposed matrices. For the case of multiple decomposed matrices, the proof just has the same logical steps: the linking-domain matrix $\mathbf{L}$ is the sum of more than one small block matrix $(\mathbf{L}_{s,(i,j)})$, where the subscript (i, j) refers to the linking-domain between subsystem S_i and subsystem S_j. Accordingly, the transform matrix $\mathbf{C}$ is also composed of more than one non-zero block matrix $(\mathbf{C}_{s,(i,j)})$ at the corresponding area. The matrix Λ has the same format because its diagonal elements are all non-zero values.

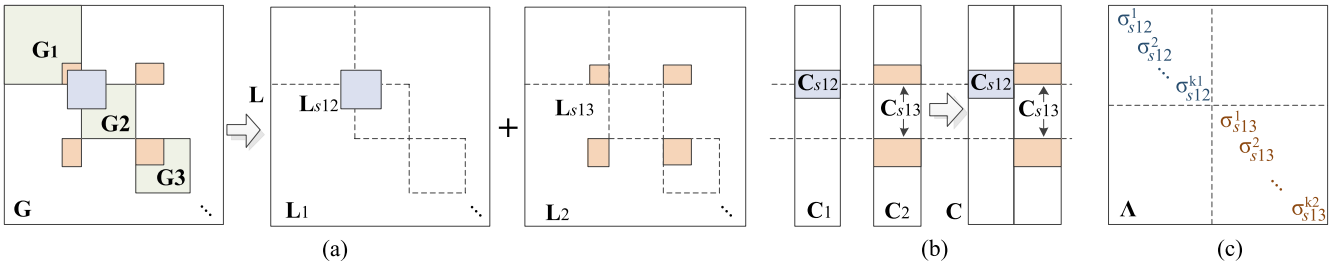


Fig. 2. Example of multiple linking-domain matrices: (a) linking-domain matrix decomposition; (b), (c) $\mathbf{C}$ and Λ matrix construction.

The general extension of (22) with multiple linking-domains is illustrated in Fig. 2, there are two cases for multiple linking-domain matrices ($\mathbf{L}_{s,(i,j)}$): adjacent case and non-adjacent case.

1) In the adjacent case, as illustrated in Fig. 2(a), subsystem S_1 and subsystem S_2 are adjacent in the network conductance matrix, thus their linking-domain is extracted as $\mathbf{L}_1$ containing the small matrix $\mathbf{L}_{s,(1,2)}$, and that is just the case proved in the above three steps.
2) In the non-adjacent case, for example, subsystem S_1 and subsystem S_3 are not adjacent in the network conductance matrix, and their connection results in a split linking-domain matrix containing the split $\mathbf{L}_{s,(1,3)}$. In this case, the $\mathbf{C}_{s,(1,3)}$ matrix just needs to add a all-zero block matrix between the split part, as shown in Fig. 2(b).

For the two cases, the corresponding linking-domain matrix $\mathbf{L}$ is the sum of the two sub-LDM $\mathbf{L}_{1,2}$, while the transform matrix $\mathbf{C}$ is not the sum but the combination of the two small transform matrices $\mathbf{C}_1$ and $\mathbf{C}_2$, as shown in Fig. 2(b). The Λ matrix is also the combination of the σ values of the two small $\Lambda_{1,2}$ matrix. The correctness of this alignment can be verified through block matrix multiplication, which just has the same procedure as Step 2. To conclude, the above proof can not only be used to prove the correctness of *Lemma 1*, but also to construct the transform matrix $\mathbf{C}$ and Λ.

C. Inverse Matrix of the Sum of LDM and DBM

Generally, the inverse matrix of the sum of two matrices is not equal to the sum of the two inverse matrices, i.e., $(A+B)^{-1} \neq A^{-1}+B^{-1}$. Therefore, it is difficult (or impossible) to find a general formulation of the inverse matrix of the sum of two matrices. However, since the $\mathbf{L}$ matrix in the network matrix $\mathbf{G}=\mathbf{G}_d+\mathbf{L}$ can be expressed as $\mathbf{L}=\mathbf{C}\Lambda\mathbf{C}^T$, the general formulation of $\mathbf{G}^{-1}$ can be found using the Woodbury matrix identity [12]:

$$\mathbf{G}^{-1}=(\mathbf{G}_d+\mathbf{L})^{-1}=\mathbf{G}_d^{-1}-\mathbf{G}_d^{-1}\mathbf{P}\mathbf{G}_d^{-1} \tag{25}$$

where:

$$\mathbf{P}=\mathbf{C}\mathbf{Q}\mathbf{C}^T \tag{26}$$

$$\mathbf{Q}=(\Lambda^{-1}+\mathbf{C}^T\mathbf{G}_d^{-1}\mathbf{C})^{-1} \tag{27}$$

The correctness of (25), (26), (27) can be verified by reconstructing the linking-domain matrix $\mathbf{L}$ using $\mathbf{G}_d$ and $\mathbf{P}$:

$$\begin{aligned}\mathbf{L}&=\mathbf{C}\Lambda\mathbf{C}^T\\&=\mathbf{C}[\Lambda\mathbf{Q}^{-1}\mathbf{Q}]\mathbf{C}^T\\&=\mathbf{C}[(\mathbf{I}+\Lambda\mathbf{C}^T\mathbf{G}_d^{-1}\mathbf{C})(\Lambda^{-1}+\mathbf{C}^T\mathbf{G}_d^{-1}\mathbf{C})^{-1}]\mathbf{C}^T\\&=\mathbf{C}\mathbf{Q}\mathbf{C}^T+\mathbf{C}\Lambda\mathbf{C}^T\mathbf{G}_d^{-1}\mathbf{C}\mathbf{Q}\mathbf{C}^T\\&=\mathbf{P}+\mathbf{C}\Lambda\mathbf{C}^T\mathbf{G}_d^{-1}\mathbf{P}\\&=(\mathbf{I}+\mathbf{L}\mathbf{G}_d^{-1})\mathbf{P}\end{aligned} \tag{28}$$

After (28) is obtained, (25) can be verified directly:

$$\begin{aligned}&(\mathbf{G}_d+\mathbf{L})(\mathbf{G}_d^{-1}-\mathbf{G}_d^{-1}\mathbf{P}\mathbf{G}_d^{-1})\\&=\mathbf{I}+\mathbf{L}\mathbf{G}_d^{-1}-\mathbf{P}\mathbf{G}_d^{-1}-\mathbf{L}\mathbf{G}_d^{-1}\mathbf{P}\mathbf{G}_d^{-1}\\&=\mathbf{I}+(\mathbf{I}+\mathbf{L}\mathbf{G}_d^{-1})\mathbf{P}\mathbf{G}_d^{-1}-\mathbf{P}\mathbf{G}_d^{-1}-\mathbf{L}\mathbf{G}_d^{-1}\mathbf{P}\mathbf{G}_d^{-1}\\&=\mathbf{I}\end{aligned} \tag{29}$$

which means the inverse matrix of $(\mathbf{G}_d+\mathbf{L})$ is exactly $\mathbf{G}_d^{-1}-\mathbf{G}_d^{-1}\mathbf{P}\mathbf{G}_d^{-1}$. This important feature could be used to accelerate the computation of inverse matrix because the diagonal block matrix $\mathbf{G}_d^{-1}$ is easier to compute.

D. Parallel Computation Using LDE

Based on (25), computing the inverse matrix of $\mathbf{G}$ actually only needs to know the value of $\mathbf{G}_d^{-1}$, because the diagonal elements of Λ^{-1} are just reciprocals of those of Λ, and $\mathbf{C}$ is constant once the network circuit topology is determined. Since $\mathbf{G}_d$ is a diagonal block matrix, the inverse matrix of its block sub-matrices on the diagonal can be computed in parallel. Therefore, the parallel computation can be executed as follows:

1) compute $\mathbf{G}_d^{-1}$ and Λ^{-1};
2) compute $\mathbf{Q}$ and $\mathbf{P}$;
3) compute $\mathbf{G}^{-1}=\mathbf{G}_d^{-1}-\mathbf{G}_d^{-1}\mathbf{P}\mathbf{G}_d^{-1}$;

Complexity analysis: For the first step, the computation of the inverse matrix can be executed in parallel with a computational complexity of $O(N_j^3)$, where N_j is the maximum size of the diagonal block matrices. For the second step, since the transform matrix $\mathbf{C}$ is already known and only contains $+1$, -1, or 0, the elements in $\mathbf{C}^T\mathbf{G}_d^{-1}\mathbf{C}$ actually can be obtained from $\mathbf{G}_d^{-1}$ with only plus or minus operations. This means, $\mathbf{Q}^{-1}$ can be obtained without multiplication operations after the first step. Then the computational complexity of computing $\mathbf{Q}$ is $O(k^3)$. Similarly, after $\mathbf{Q}$ is obtained, the elements in $\mathbf{P}$ can be obtained directly. For the third step, the complexity is determined by the computation of $\mathbf{G}_d^{-1}\mathbf{P}\mathbf{G}_d^{-1}$, which depends on the parallel technique applied to calculate the matrix multiplication. If the matrix multiplications run in parallel for each block matrix, the complexity will be $O(N_j^3)$; if the matrix multiplication run in

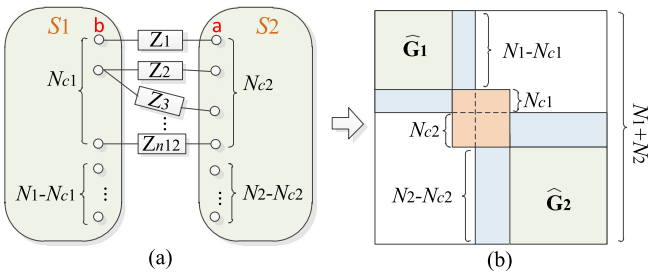


Fig. 3. Example of two decomposed subsystems: (a) subsystem connection; (b) matrix decomposition.

massively parallel fashion for each row and each column, then the complexity is $O(N_j^2)$. Thus the total complexity of the LDE method is $O(N_j^3 + k^3)$ if block-based parallel processing is exploited for the third step.

E. Advantages and Limitations of LDE

Compared to the SC method that requires the information of the equivalent current injections, the biggest advantage of the LDE method is that it does not need to know the right hand side of the network matrix equation, because it could directly compute the inverse matrix of the network conductance matrix. Therefore, the LDE method is essentially a matrix inversion method rather than a circuit solution method. If the network conductance matrix does not change during the simulation, the LDE method will accelerate the simulation process dramatically.

In addition, if the network conductance matrix changes over the simulation duration, the LDE method may also possibly run faster than SC method in many cases. According to the complexity analysis, the LDE method has higher complexity in solving the block matrix inversion because the sizes of the decomposed block matrices (N_j) in LDE are usually larger than (or equal, when the decomposed matrices have no connections) that of SC method. However, the matrix inversion of the Λ matrix in LDE and $\mathbf{D}_t$ matrix in SC method also contribute a considerable part in complexity, ignoring the matrix multiplication operations since they can be calculated in massively parallel fashion. Thus if the Λ matrix is smaller than the $\mathbf{D}_t$ matrix in SC method and the benefits introduced by a smaller Λ matrix are larger than the extra computational cost caused by a larger N_j, then LDE will be more suitable over SC method under a variable network conductance matrix.

To demonstrate this, the simple two decomposed block matrices case is illustrated in Fig. 3. There are N_1 and N_2 nodes in the two decomposed subsystems (S1 and S2), and N_{c1} nodes in S1 and N_{c2} nodes in S2 are connected through n_{12} conductors. Then the corresponding matrix decomposition is shown in Fig. 3(b). The basic precondition to make LDE better is that the size of Λ matrix is smaller than the size of overlapping domain, i.e., the connections between interface nodes are not dense:

$$k = n_{12} < N_{c1} + N_{c2} \tag{30}$$

In contrast to the above advantages, there are also limitations for the application of LDE method and they are summarized as follows:

1) The connections between interface nodes are not dense, otherwise the Λ matrix will be too large to achieve desired performance;
2) There is no trans-conductance between the interface nodes and the other nodes, which means the non-zero elements in the linking-domain matrix only refer to the relationship between the interface nodes;
3) The diagonal block matrix $\mathbf{G}_d$ should be invertible, because all the computation is based on $\mathbf{G}_d^{-1}$.

In fact, if the network is decomposed properly, an invertible $\mathbf{G}_d$ can usually be guaranteed. The case where $\mathbf{G}_d$ is not invertible is that there exist elements with 0 values in the diagonal location of $\mathbf{G}_d$. That is, the resulting matrix after the overlapping domain matrix subtracts linking-domain matrix has 0 elements in diagonal locations, which means that the corresponding interface nodes only connect with other nodes via the linking conductors. For an example of this case: in Fig. 3(a) the interface node a only connects with the other interface node b via a conductor, but the other side of node a connects with a voltage source. Then the resulting $\mathbf{G}_d$ will not be invertible because the diagonal location of a in $\mathbf{G}_d$ will be equal to 0. In practical AC/DC networks, this type of node usually exists at the "edge," and assigning this type of node as interface nodes can neither achieve a good acceleration nor result in an invertible $\mathbf{G}_d$. Avoiding this type of nodes as interface nodes when decomposing the network may lead to an invertible $\mathbf{G}_d$.

F. Optimal Decomposition Based on LDE

LDE is a matrix-based decomposition method, which means, the overall speed-up is actually determined by the number of decomposed subsystems and matrix size of these subsystems given a specific network topology. However, how to decompose the network and how to allocate the power system equipment into properly decomposed subsystems to achieve optimal speed-up remains a problem to be solved. This problem is similar to the *minimum k-cut* problem in graph theory, which aims to minimize the links between decomposed sub-graphs. However, the optimal decomposition problem for LDE decomposition is a little different from the *minimum k-cut* problem, because both the number of links and the size of decomposed subsystems will affect the overall performance. For large-scale systems with large number of nodes, this problem is NP-hard and is difficult to solve. Therefore, there is a trade-off between the achieved speed-up and latency to perform the decomposition algorithm. Some heuristic methods could be exploited to solve this problem, and they are left for future research.

IV. Simulation Results and Speed-Up

In this section, two test systems are simulated on the FPGA and GPU board respectively. The speed-up of computing matrix inversion is also evaluated in comparison to the SC method and Gauss-Jordan (GJ) method.

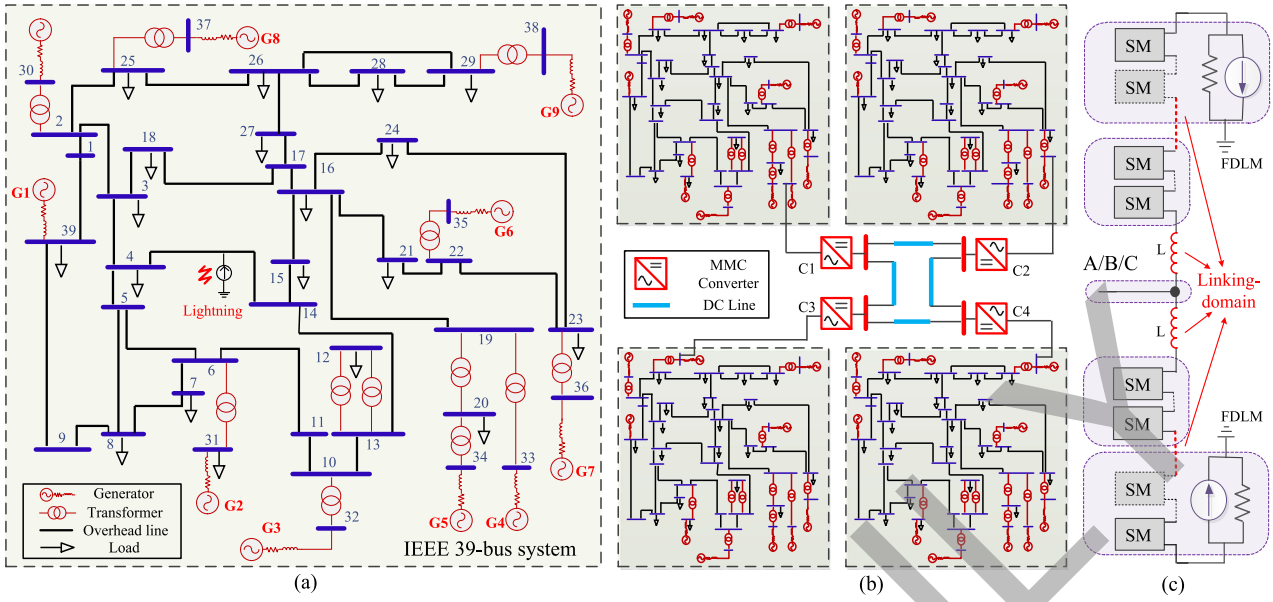


Fig. 4. Test circuits: (a) IEEE 39-bus test system; (b) hybrid AC/DC grid; (c) decomposition of each phase circuit of MMC.

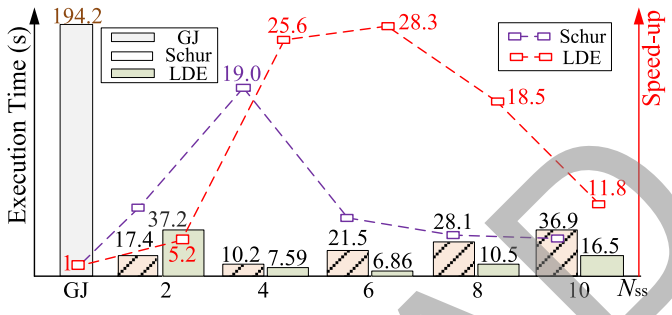


Fig. 5. IEEE 39-bus execution time comparison between the GJ, SC, and LDE method under varying number of decomposed subsystems on the FPGA.

A. Speed-Up of Matrix Equation Solution on FPGA

To verify the validity and effectiveness of the LDE method in solving large matrix equations with fully exploited parallelism, the IEEE 39-bus system [13] shown in Fig. 4(a) is simulated on the Xilinx VCU-118 board with the XCVU9P FPGA [14] at 100 MHz frequency. The system is not split through transmission lines; thus a 39×39 matrix is generated. The execution time of the LDE method with varying number of subsystems (N_{ss}) is compared with that of the GJ method and SC method, as shown in Fig. 5. Although in this case the LDE method can (but the SC method can not) pre-compute the matrix inversion before the simulation due to the constant network matrix, the latency of LDE method in Fig. 5 only shows the version that computes the matrix inversion in each time-step for pure comparison of solving linear matrix equations. In this case the conductance matrix is constant, then using LU factorization can greatly reduce the latency compared to GJ method since it can pre-compute the $\mathbf{L}$ and $\mathbf{U}$ matrices, just as the LDE method pre-computes the matrix inversion. However, since the LU factorization has nearly the same complexity as GJ method (or even larger than GJ due to the extra time to solve $\mathbf{L}(\mathbf{U}\boldsymbol{v}) = \boldsymbol{i}^{eq}$) and it cannot be paralleled well unlike the GJ method, it is better to use GJ as the base method in the parallel platform. The duration of the simulation is 1 s, and the time-step is 10 μs. Note that the latency depends on the specific network decomposition, and Fig. 5 only shows one of the possible cases.

The GJ-based matrix inversion can not be decomposed [15], [16], thus the 39×39 matrix inversion consumes the most latency. It can be observed that if the network is decomposed into 2 subsystems, the SC method has the smaller latency. Because in this case, the number of interface nodes is 8 and the number of links between interface nodes is 4, then the maximum matrix sizes of $\hat{\mathbf{G}}_i$ and $\mathbf{D}_t$ in the SC method are 16×16 and 8×8, while the maximum matrix sizes of $\mathbf{G}_i$ and Λ in LDE method are 20×20 and 4×4. Since in FPGA the matrix multiplication operation can be parallelized efficiently, the matrix inversion consumes the majority of the latency. Therefore, the inversion of a 20×20 matrix consumes much more latency than 16×16, which can not be compensated by a smaller inversion of a 4×4 matrix compared to a 8×8 matrix. When the number of decomposed subsystems increases, the number of interface nodes increases quickly, which makes the size of $\mathbf{D}_t$ in SC method to increase. Therefore, when N_{ss} increases to 6 or larger, the latency of matrix inversion of $\mathbf{D}_t$ in SC method boosts rapidly resulting in a larger overall latency. However, the Λ matrix size is only determined by the number of connecting links between subsystems but not the number of interface nodes, thus the latency of matrix inversion increases not so fast compared to that of the SC method. This result also confirms the discussion in Section III-E. The maximum speed-up of 28.3 can be achieved when the system is decomposed into 6 subsystems.

In terms of the hardware resource consumption on FPGA, the GJ method consumes the most resources (LUT, near 73%) due to the largest matrix inversion; the LDE method consumes the

fewest (LUT, near 63%) when the speed-up is larger than that of SC method.

B. Large-Scale AC/DC Network Simulation on GPU

The large-scale hybrid AC/DC network composed of 4 IEEE 39-bus systems and 4 MMC converters is simulated on the NVIDIA Tesla V100 GPU with 5012 cores, as shown in Fig. 4(b). In this simulation, detailed models for all the equipment are applied. The frequency-dependent model [17] (FDLM) is utilized for the AC and DC transmission lines; the voltage-behind-reactance model [18] is utilized for the synchronous machines; the conductance matrix model [19] is utilized for the transformers. Since FDLM can be used to decompose the network between two line ends, the 4 39-bus systems and 4 MMC converters connected through long DC lines can be decomposed to simulate in parallel; however, within each 39-bus system, the system decomposition through FDLM can be customized by users: for example, if the line between buses are too short to apply latency-based decomposition, then the two ends are aligned; or if the user want to combine some buses together to avoid large storage consumption, then FDLM can be computed without decomposition. Then within the subsystems decomposed via FDLM, the LDE method can be applied.

Inside each MMC, the system-level two-state switching model (TSSM) [20] for converters C1/C2/C3 and device-level curve-fitting model (CFM) [21] for converter C4 are applied. The TSSM models each half-bridge submodule (HBSM) as a serial connection of an equivalent resistor and a voltage source, thus the HBSM equivalent circuits in each arm can be merged as a arm resistor and voltage source. However, the CFM equivalence for each HBSM cannot be merged because the switching transient details will be lost after merging. To avoid solving the resulting extremely large matrix (usually composed of hundreds of HBSMs in each arm), a traditional method is to use the current source of the last time-step on the HBSM side to calculate the voltage source on the main circuit side, for which the one-step delay will generate an error [22]. In this work, the large matrix is inverted using the LDE method. Figure 4(c) illustrates the decomposition between HBSMs in one phase leg using the LDE method.

The four 39-bus systems and three TSSM-based MMC converters (101-level) are simulated at system-level with the time-step of 10 μs, while the device-level CFM-based MMC converter is simulated at 0.1 μs. To synchronize these decomposed systems, the system-level simulations need to wait for the completion of device-level simulation every 10 μs, thus the overall speed-up is actually produced by the acceleration of CFM-based MMC module [23], [24]. The overall speed-ups for the hybrid AC/DC network with varying number of decomposed HBSM systems in each MMC arm are recorded in Table I. Note that if there are m_{arm} HBSM systems decomposed in each arm, then the total MMC circuit is actually decomposed into $m_{MMC} = (6m_{arm} - 1)$ subsystems, where the HBSM systems connecting the DC line equivalence consist of three phases while the others only contain one phase. The case $m_{arm} = 2$ is shown in Fig. 4(c). In Table I, n_L is the number of linking-domains,

TABLE I
EXECUTION TIME AND SPEED-UP OF DIFFERENT DECOMPOSITIONS FOR ONE CYCLE (16.67 MS) SIMULATION ON GPU

m_{arm}	m_{MMC}	n_L	N_{max}	Latency (s)	Speed-up
1	5	6	301×301	213331s	1
2	11	12	76×76	11241s	19
3	17	18	44×44	4293s	50
4	23	24	31×31	2919s	73
5	29	30	30×30	2703s	79
6	35	36	36×36	3232s	66

and N_{max} denotes the size of the largest matrix to be inverted. Since inverting the entire matrix (605×605) involves too huge computational effort that exceeds the parallel capability, the MMC circuit is first decomposed to 5 subsystems ($m_{arm} = 1$) as the basic decomposition. Although the GPU-based implementation consumes much more latency than FPGA due to the limitations of parallelism, the maximum speed-up of matrix inversion process can be even larger because of the large system scale and the sparse connection links between HBSMs in each arm. As show in the Table, the maximum speed-up over the basic decomposition is 79, which indicates the latency can be reduced by much more than 79 times compared to directly solving the whole MMC circuit. Besides, when the number of decomposed subsystems exceeds 29, the speed-up starts to decrease due to the increased size of Λ matrix.

V. CONCLUSION

Domain decomposition is an efficient approach to deal with large-scale circuit simulation. In this paper, a new non-overlapping domain decomposition method is proposed: the linking-domain extraction (LDE) based decomposition method. In LDE, the linking-domain matrix (LDM) is first extracted from the original network conductance matrix, which makes the remaining part to be a diagonal block matrix (DBM) that is easy to calculate in parallel. An important *lemma* describing the feature of LDM and its proof are presented, which indicates that the LDM can be transformed from a diagonal matrix. Based on this *lemma*, the general formulation of the inverse matrix of the sum of LDM and DBM can be found using the Woodbury matrix identity, and then the network matrix inversion can be computed for each block in parallel. Compared to the Schur complement (SC) method, LDE method can compute the matrix inversion directly; and when the network matrix changes, the LDE method can also run faster than the SC method in many cases. The simulation results for the IEEE 39-bus system and speed-ups over the SC method and Gauss-Jordan method demonstrate the validity and efficiency of the proposed LDE method. The special acceleration over the nonlinear test cases and larger AC/DC test systems will be considered in future work.

APPENDIX

Parameters of the 39-bus test system: Base values: 100MVA, 230 kV, 60 Hz; Generator: V_{rms}=230 kV. Transformers: 230 kV/230 kV, leakage inductance 0.2pu, copper loss 0.004pu; Transmission lines: the length of lines are given in [13], and

the FDLM equivalence parameters are obtained in PSCAD accordingly. Parameters of MMC: $V_{dc} = 400$ kV, 101-level, $C_{sm} = 2.5$ mF, $f_C = 2000$ Hz, $L_{arm} = 0.0189$ H; device-level: $t_{d,on} = 0.4\mu$s, $t_r = 0.45$ μs, $t_{d,off} = 3.0$ μs, $t_f = 0.3$ μs.

REFERENCES

[1] F. N. Najm. *Circuit Simulation.* Hoboken, NJ, USA: Wiley, 2010.

[2] H. W. Dommel, "Digital computer solution of electromagnetic transients in single-and multiphase networks," *IEEE Trans. App. Syst.*, vol. 88, no. 4, pp. 388–399, Apr. 1969.

[3] C. W. Ho, A. Ruehli, and P. Brennan, "The modified nodal approach to network analysis," *IEEE Trans. Circuits Syst.*, vol. 22, no. 6, pp. 504–509, Jun. 1975.

[4] T. F. Chan and T. P. Mathew, *Domain Decomposition Algorithms.* Cambridge, U.K.: Cambridge Univ. Press, 1994.

[5] A. Toselli and O.B. Widlund, *Domain Decomposition Methods: Algorithms and Theory.* Berlin, Germany: Springer-Verlag, 2005.

[6] S. Y. R. Hui, K. K. Fung, and C. Christopoulos, "Decoupled simulation of multi-stage power electronic systems using transmission-line links," in *Proc. Rec. IEEE Power Electron. Specialists Conf.*, 1992, pp. 1324–1330.

[7] S. Y. R. Hui and K. K. Fung, "Fast decoupled simulation of large power electronic systems using new two-port companion link models," *IEEE Trans. Power Electron.*, vol. 12, no. 3, pp. 462–473, May 1997.

[8] J. E. Schutt-Aine, "Latency insertion method (LIM) for the fast transient simulation of large networks," *IEEE Trans. Circuits Syst. I: Fundam. Theory Appl.*, vol. 48, no. 1, pp. 81–89, Jan. 2001.

[9] S. N. Lalgudi, M. Swaminathan, and Y. Kretchmer, "On-chip power-grid simulation using latency insertion method," *IEEE Trans. Circuits Syst. I*, vol. 55, no. 3, pp. 914–931, Apr. 2008.

[10] P. Aristidou, D. Fabozzi, and T. V. Cutsem, "Dynamic simulation of large-scale power systems using a parallel Schur complement-based decomposition method," *IEEE Trans. Parallel Distribution Syst.*, vol. 25, no. 10, pp. 2561–2570, Oct. 2014.

[11] P. Aristidou, S. Lebeau, and T. V. Cutsem, "Power system dynamic simulations using a parallel two-level Schur complement decomposition," *IEEE Trans. Power Syst.*, vol. 31, no. 5, pp. 3984–3995, Sep. 2016.

[12] M. A. Woodbury, *Inverting Modified Matrices.* Memorandum Rept. 42, Princeton, NJ, USA: Princeton Univ. Press, 1950.

[13] *PSCAD IEEE 39 Bus System*, Revision 1, Manitoba HVDC Research Centre, Winnipeg, Manitoba, Canada, 2010.

[14] *VCU118 Evaluation Board User Guide (UG1224)*, Xilinx Inc., San Jose, CA, USA, Oct. 2018.

[15] Y. Chen and V. Dinavahi, "Multi-FPGA digital hardware design for detailed large-scale real-time electromagnetic transient simulation of power systems," *IET Gener. Transmiss. Distribution*, vol. 7, no. 5, pp. 451–463, May 2013.

[16] Y. Chen and V. Dinavahi, "Hardware emulation building blocks for real-time simulation of large-scale power grids," *IEEE Trans. Ind. Informat.*, vol. 10, no. 1, pp. 373–381, Feb. 2014.

[17] B. Gustavsen and A. Semlyen, "Simulation of transmission line transients using vector fitting and modal decomposition," *IEEE Trans. Power Del.*, vol. 13, no. 2, pp. 605–614, Apr. 1998.

[18] L. Wang and J. Jatskevich, "A voltage-behind-reactance synchronous machine model for the EMTP-type solution," *IEEE Trans. Power Syst.*, vol. 21, no. 4, pp. 1539–1549, Nov. 2006.

[19] J. Liu and V. Dinavahi, "A real-time nonlinear hysteretic power transformer transient model on FPGA," *IEEE Trans. Ind. Electron.*, vol. 61, no. 7, pp. 3587–3597, Jul. 2014.

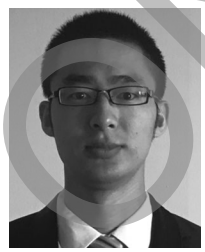
[20] Z. Shen and V. Dinavahi, "Real-time device-level transient electrothermal model for modular multilevel converter on FPGA," *IEEE Trans. Power Electron.*, vol. 31, no. 9, pp. 6155–6168, Sep. 2016.

[21] Z. Huang and V. Dinavahi, "A fast and stable method for modeling generalized nonlinearities in power electronic circuit simulation and its real-time implementation," *IEEE Trans. Power Electron.*, vol. 34, no. 4, pp. 3124–3138, Apr. 2019.

[22] N. Lin and V. Dinavahi, "Detailed device-level electrothermal modeling of the proactive hybrid HVDC breaker for real-time hardware-in-the-loop simulation of DC grids," *IEEE Trans. Power Electron.*, vol. 33, no. 2, pp. 1118–1134, Feb. 2018.

[23] Z. Zhou and V. Dinavahi, "Parallel massive-thread electromagnetic transient simulation on GPU," *IEEE Trans. Power Del.*, vol. 29, no. 3, pp. 1045–1053, Jun. 2014.

[24] Z. Zhou and V. Dinavahi, "Fine-grained network decomposition for massively parallel electromagnetic transient simulation of large power systems," *IEEE Power Energy Technol. Syst. J.*, vol. 4, no. 3, pp. 51–64, Sep. 2017.



Tong Duan (Student Member, IEEE) received the B.Eng. degree in electronic engineering from Tsinghua University, Beijing, China, in 2013. He is currently pursuing the Ph.D. degree in electrical and computer engineering at the University of Alberta, Edmonton, Alberta, Canada. His research interests include real-time simulation of power systems, power electronics, and parallel computing.

Venkata Dinavahi (Fellow, IEEE) received the B.Eng. degree in electrical engineering from the Visveswaraya National Institute of Technology (VNIT), Nagpur, India, in 1993, the M.Tech. degree in electrical engineering from the Indian Institute of Technology (IIT) Kanpur, Kanpur, India, in 1996, and the Ph.D. degree in electrical and computer engineering from the University of Toronto, Toronto, ON, Canada, in 2000. Presently he is a Professor with the Department of Electrical and Computer engineering, University of Alberta, Edmonton, AB, Canada. His research interests include real-time simulation of power systems and power electronic systems, electromagnetic transients, device-level modeling, large-scale systems, and parallel and distributed computing.