Variable Time-Stepping Parallel Electromagnetic Transient Simulation of Hybrid AC–DC Grids

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

***Abstract*—The complicated hybrid ac–dc network architecture creates new challenge for electromagnetic transient (EMT) simulation of large-scale systems in terms of accuracy and efficiency. By utilizing the variable time-stepping (VTS) method and graphics processing unit (GPU) parallelism, this article proposes a four-level dynamic parallelism architecture for variable time-stepping EMT simulation of hybrid ac–dc grids. By applying the proposed hierarchical system decomposition and VTS scheme, multiple time-step areas (TSAs) that contain subsystems with the same time-step size and adaptation criteria can be computed in different GPU blocks in parallel. Taking advantage of the dynamic parallelism feature of GPUs, a four-level dynamic parallelism is proposed to fully exploit the possibility of parallelizing the VTS simulation, though which the subsystems within each TSA and the detailed equipment models within each subsystem can run also in parallel via elaborate configurations. The transient waveforms and execution time speed-ups indicate that the proposed method can extremely accelerate the simulation process while guaranteeing reasonable accuracy compared to the fixed time-step based simulation.**

***Index Terms*—Dynamic parallelism (DP), electromagnetic transients (EMTs), graphics processing unit (GPU), hybrid ac–dc network, modular multilevel converter, parallel processing, variable time-step (VTS).**

I. Introduction

With the development of high-voltage direct current transmission technology, the traditional pure ac power system is transitioning into the hybrid ac–dc grid. In ac–dc grids, transmitting large amount of electricity over long distances is more efficient due to the benefits of using direct current for the bulk transmission of electrical power and ease of controlling active power in the link [1]. In the interwoven network of ac and dc systems, the equipment-level and even the electronic-level behaviors are closely coupled with the system wide responses, which creates new challenges for electromagnetic transient (EMT) simulation in terms of achieving higher accuracy while simultaneously maintaining high efficiency (i.e., fast computation) especially of large-scale ac–dc grids.

Manuscript received February 4, 2020; revised June 5, 2020 and August 4, 2020; accepted September 25, 2020. Date of publication September 30, 2020; date of current version December 18, 2020. This work was supported by the Natural Science and Engineering Research Council of Canada (NSERC) and China Scholarship Council (CSC). *(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/JESTIE.2020.3028007

Most existing offline EMT-type simulation tools (such as ATP, PSCAD/EMTDC, EMTP-RV, etc.) whether commercial or noncommercial use a fixed time-step (FTS), which is decided *a priori* before the start of the simulation based on the user's judgement of the fastest possible transient during the simulation run. This FTS is often small enough to meet the accuracy requirement during transient and is not suitable for modeling all the time-varying transient components contained in the entire system, especially in a hybrid ac–dc system. For example, a small time-step (of the order of tens of nanoseconds) that is chosen to capture the device-level switching transients of ac–dc converters is too small resulting in excessive execution run time of the entire system simulation, while a relatively large time-step chosen for modeling only the system-level transients would be obviously ineffectual in reproducing the device transients. This dilemma often requires the power system engineer to develop and maintain different EMT system models (sometimes on different tools) with varying equipment model complexities to strike a careful balance between the desired simulation accuracy and the incurred computational burden.

To accelerate the simulation process without losing accuracy, the variable time-step (VTS) method [2]–[4] that adaptively changes the time-step during simulation execution has obvious advantages over the fixed time-stepping simulation, wherein the time-step generally changes according to specific accuracy requirements that are evaluated by some predetermined criteria such as the local truncation error (LTE) and dv/dt (DVDT). However, the existing VTS simulation research either focuses on the single equipment such as the transmission line model [3] and modular multilevel converter (MMC) model [5], or was implemented on field-programmable gate array (FPGA) with limited system scale [6]; how to simulate EMTs in large-scale ac–dc systems with VTSs remains to be investigated. Since the variable time-stepping simulation for large-scale ac–dc networks involves extremely high computational effort, parallel execution of the EMT simulation code is a necessity on massively parallel hardware, such as the FPGA and graphics processing unit (GPU). The FPGA-based EMT simulation enables fully parallel computation within each time-step in real-time; however, usually the FPGA board has limited hardware resources to accommodate large systems. Although the GPU-based EMT simulation for ac–dc grids has also been studied in previous works [7]–[16], the VTS EMT simulation is very different from the FTS-based simulation on GPUs, and the challenges mainly come from the following two aspects: 1) it is difficult to synchronize between processing kernels since the subsystems

(SSs) calculated in different kernels may use different time-step size and the adaptation criteria may differ; 2) it is difficult to achieve enough parallelism because the simulation latencies of different kernels are changing, and global parallelism requires elaborate coordination between different SSs computed in different kernels. The VTS simulation for MMC converters on GPU is studied in [5], but it is purely targeting the MMC circuit while not involving ac system. Thus to authors' best knowledge, the parallel VTS-based EMT simulation of hybrid ac–dc networks has not been evaluated on the GPU platform.

Based on the abovementioned observations, this article proposes a variable time-stepping massively parallel architecture for EMT simulation of hybrid ac–dc grids with detailed equipment models. The contributions of the proposed parallel VTS simulation architecture can be summarized into the following two aspects.

1) A novel hierarchical variable time-stepping scheme is proposed, wherein the whole ac–dc system is decomposed into several time-step areas (TSAs) that contain the SSs with the same time-step size and adaptation criteria, and each TSA holds its own time-step control logic while synchronizing to the global simulation time.
2) A fine-grained four-level parallelism is achieved to fully exploit the possibility of parallel EMT simulation, by taking advantage of the dynamic parallelism (DP) feature of GPUs [17]: each TSA is executed as the first level kernel; the SSs within each TSA run as the second level function; the equipment models within each SS run for the third level; then within some specific equipment models the processing steps can also run in parallel as the fourth level parallelism.

Based on the proposed architecture, the test case composed of the IEEE 118-Bus system and three MMC ac–dc converters is implemented on GPU, while the implementation details are also described, including the interaction between TSAs and parallelism options for different levels. The rest of this article is organized as follows. Section II provides the proposed hierarchical variable time-stepping scheme. Section III presents the four-level parallel architecture with VTS schemes on the many-core GPU architecture. In Section IV, the GPU-based implementation details for the hybrid ac–dc grid test case are described, whose results are compared with existing EMT simulation tools in Section V. Finally, Section VI conclusions are drawn.

II. Hierarchical Variable Time-Stepping Scheme

In the hybrid ac–dc system, the time-constants of different equipment are quite different considering the device-level and system-level transient simulations. Applying a global VTS scheme for the whole system may be inefficient, thus, in this section, the hierarchical VTS scheme is proposed based on the specific time-step control schemes.

A. Time-Step Adaptation Criteria

In hybrid systems, ac and dc grids are interconnected, wherein linear and nonlinear elements coexist. In such a system, measuring the system perturbation is the prerequisite to determine the time-step change and control scheme. In this work, different methods are applied to estimate the accuracy.

1) Linear Equipment: It is easy to find the solution for linear elements even with VTSs because the network conductance matrix only depends on the history items at t_{n-1}. The LTE is usually used to estimate the accuracy of the solved variable x, given by [18]

$$\mathrm{LTE}(t_n) \approx C_{p+1}\Delta t_n^{p+1}(p+1)!\, g[t_n, \ldots, t_{n-1-p}] \tag{1}$$

where C_{p+1} is the error constant of a specific discretization method, p is the order, and $g[t_{n-1}, \ldots, t_{n-1-k}]$ can be calculated step-by-step

$$g(t_{n-1}) = x_{n-1} \tag{2}$$

$$g[t_{n-1}, \ldots, t_{n-k}] = \frac{g[t_{n-1}, \ldots, t_{n-k+1}] - g[t_{n-2}, \ldots, t_{n-k}]}{t_{n-1} - t_{n-k}}. \tag{3}$$

2) Nonlinear Equipment: Finding x_n for nonlinear equipment requires solving the nonlinear system using an iterative approach. The standard method is to first use an explicit method or interpolation polynomial (called the predictor) to get a candidate value of x_n, and then use it as the initial solution to apply Newton's iterative method for the implicit integrator (called the corrector) until convergence is achieved. For the predictor, the interpolation polynomial is commonly used

$$x_n^{(0)} = x_{n-1} + \sum_{k=1}^{p}\left[\prod_{j=1}^{k}(t_n - t_{n-j})\right] g[t_{n-1}, \ldots, t_{n-1-k}]. \tag{4}$$

Then, the LTE can be estimated by comparing the initial solution x_n^0 and final solution x_n [18]

$$\mathrm{LTE}(t_n) \approx \frac{C_{p+1}}{1 - C_{p+1}}(x_n - x_n^0). \tag{5}$$

3) AC–DC Converter: Although an MMC is also made up of linear and nonlinear equipment such as the insulated gate bipolar transistor (IGBT) switches, capacitors and inductors, the LTE method may not be suitable to measure the precision because it is hard to find which state variable is most representative among the thousands of switches and capacitors and six arm inductors. Thus, for the system-level simulation, the differential value DVDT or di/dt of dc voltage or current is computed to measure the system disturbance and determine the time-step change; for the device-level simulation, the switching operation is used to trigger the time-step change.

B. Hierarchical VTS Method

Generally, changing the time-step arbitrarily is not practical for simulation of large-scale hybrid ac–dc grids because the time constants of various equipment are quite different and using a same time-step for the whole system will be inefficient and inaccurate. Although the whole system can be divided into SSs based on the traveling wave line model or frequency-dependent line model and each SS may run in different time-steps [19], their time-step size also cannot be arbitrarily assigned due to the necessity of synchronization. For example, if an SS uses a 3 μs

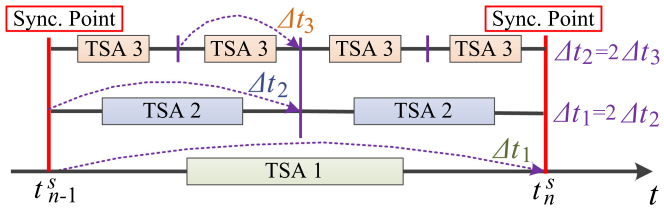

Fig. 1. Example of synchronization between TSAs.

time-step and its connected SS uses a 7 μs time-step, then it will be extremely complicated to synchronize the two SSs because they could not reach at the same synchronization point after each time-step. Another reason for specific consideration on system decomposition and time-step size is that the transmission delay of the transmission lines between decomposed SSs should be larger than the time-step size, which is prerequisite to decouple the connected systems.

Therefore, the hierarchical VTS scheme is proposed: in the first level, the hybrid ac–dc system is decomposed into several TSAs, and all the equipment models within the same TSA always utilize the same time-step size; in the second level, each TSA is then decomposed into various SS for parallel processing; in the third level, each SS then contains several power equipment; finally, each equipment within an SS calculates its own LTE. After each time-step, each TSA $(A_1, \ldots, A_N)$ compares the LTEs or DVDTs of the contained equipment to increase or decrease the time-step based on the time-step adaptation threshold. There are several LTE thresholds corresponding to different time-step sizes. For example, if there are n candidate time-step sizes $(\Delta t_1, \ldots, \Delta t_n)$ then there will have $(n-1)$ LTE thresholds $(\xi_1, \ldots, \xi_{n-1})$; then if the LTE is bigger than ξ_i and smaller than ξ_{i+1}, the time-step size will change to Δt_{i+1}.

C. Synchronization Between TSAs

The interactions between TSAs mainly include the following two aspects: 1) data exchange between TSAs, referring to the history items of the transmission line model; 2) time-step coordination and synchronization, indicating how to determine the synchronization point based on the dynamically changed time-step sizes. To handle these two aspects, the time-step set of each TSA should be first configured properly, and in this work, the time-step sizes for different TSAs always belong to the same time-step set $\Delta T = \{\Delta t_{\min} \times (2^0, 2^1, \ldots, 2^n)\}$, where $\Delta t_{\min}$ is minimum time-step size for the system. Then, the large time-step size of difference TSAs are always multiple times of the smaller time-step, which makes synchronization easy.

The synchronization process refers to the concept that each TSA proceeds to the same synchronization point with different time-steps to exchange data with connected TSAs. The time-space between synchronization points is a variable, which is the maximum time-step of these TSAs after the last synchronization point. Since their time-steps are always linearly proportional, the other TSAs with smaller time-steps are only required to execute several times to reach that synchronization point. An example is shown in Fig. 1, after time-step change at the $(n-1)$th synchronization point, the time-step of TSA-1 (Δt_1) becomes the largest one and, thus, determines the time-space to the next synchronization point. Then, TSA-2 and TSA-3 execute several steps using their own time-step size to the next synchronization point so that all the TSAs could exchange their data and proceed the simulation.

III. Four-Level Parallel VTS Simulation Architecture

The DP feature of GPUs enables nested kernel functions execution, which is suitable for the hierarchical VTS processing architecture. Through proper system decomposition and GPU run-time configurations, the massively parallel VTS simulation can be achieved.

A. Dynamic Parallelism

The GPU-based programming involves two parts of the hardware resources: host, on which the CPU programs run serially, and device, on which the GPU programs run in parallel. The GPU programming model is based on primitives of threads, blocks, and grids: a grid is a collection of threads, and the threads in a grid execute a kernel function and are divided into blocks that is a group of threads, which execute on the same multiprocessor and have access to the same shared memory. Typically, the kernel function defining the program executed by individual threads within a block and grid can only be called by the host, which involves sophisticated execution control and frequent data transfer between host and device. As an extended capability to the GPU programming model, the DP feature enables the kernel function to create and synchronize with new kernel functions on the GPU device dynamically at whichever point in a program. The grid that has launched new grid(s) is called a "parent" grid, and the one is launched by a parent grid is called "child" grid. Grids launched with dynamic parallelism are fully nested, which means the parent is not considered completed until all of its launched child grids have also completed, as shown in Fig. 2(a).

Despite the advantages of DP, it also introduces a cost in launching kernels, which is considerable compared with the execution time of child kernels. If the child kernels do not extract much parallelism and there is not much benefit against their nonparallel counterparts, then the little benefit may be canceled out by the child kernel launching overheads. Thus, when applying the DP, the massive parallelism of child kernel functions is preferred to guarantee the performance gain in global scope.

B. VTS Simulation Architecture

The aforementioned considerations of the programming architecture and memory model of DP form the basics of parallel EMT simulation, which enables the proposed hierarchical VTS scheme to be executed in a massively parallel way. Considering the nested "parent-child" grids have almost the same hierarchical structure as the hierarchical VTS method, it is a natural idea to map the TSAs and SSs into specific virtual processing units. As shown in Fig. 2(b), the hybrid ac–dc system is divided into N TSAs using the equivalent circuit of transmission lines, and the hierarchy of the simulation is listed as follows and illustrated in Fig. 2(c).

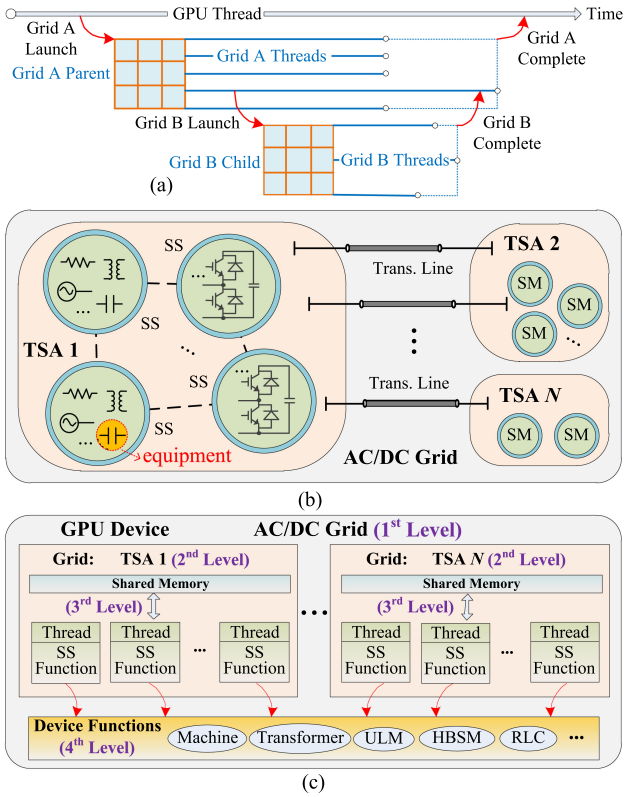

Fig. 2. GPU-based VTS simulation. (a) Illustration of DP. (b) Hierarchical VTS architecture. (b) DP-based simulation on GPUs.

1) *First-level function:* The top kernel function used to simulate the whole system, which is called by the CPU program directly.
2) *Second-level function:* A TSA function used to simulate one specific TSA, containing SSs that have the same time-step sizes and changing rate during the simulation.
3) *Third-level function:* An SS function used to simulate a small circuit containing various power equipment or power electronic devices.
4) *Fourth-level function:* an equipment function used to calculate a specific device model such as machines, transformers, loads, and power converters.

Generally, the first-level function must be a kernel function to run the simulation program on the GPU device, but the necessity of applying DP in the 2–4 level function should be evaluated before the simulation since the overhead of launching kernels is not negligible. Assume the time of launching child kernels is t_c, and processing time of K SSs in a TSA is $t^i_{\rm ss}, i = 1, \ldots, K$, then the second-level function should be a kernel function running as the "child" grid of the first-level kernel function when

$$t_c + \max(t^i_{\rm ss}) < \sum_{i=1}^{K} t^i_{\rm ss}. \tag{6}$$

The consideration of applying DP in each SS function and device function is the same as (6). Generally, once the scale of a TSA is determined, the granularity of system decomposition will significantly impact the application of DP. For example, if the system decomposition is fine-grained, which means an SS contains very limited equipment and a TSA is composed of large number of SSs, then the second-level function is usually running as a kernel function to improve the parallelism but the third-level function is not required to be a kernel function. On the contrary, if there are not many SSs in each TSA and each SSs contains numerous devices then the third-level kernel function should be generated.

Note that the equipment models can also run in parallel in the SS function even though the fourth-level kernel function is not used because if the SS function is a kernel function then it can be divided into blocks and threads to run the equipment models in parallel. In fact, the main application of DP for the fourth-level function is the frequency-dependent transmission line equipment model, because it involves many convolution processes that can run in massively parallel. If an SS is connected with many other SSs via transmission lines, then these transmission line equipment models should be executed in kernel functions to achieve fully parallel calculation and improve the overall performance.

The synchronization process is required between TSA functions since they use different time-step sizes, and the main issue to be solved during synchronization process is the consistency of exchanged data between TSAs. For example, if the TSA A_i needs the results v_{Aj} from TSA A_j at simulation time t, then A_i should interpolate the results received from A_j into the data for its own use. Typically, the interpolation is required to synchronize the results with different time-steps, but in this work the results of each TSA can be directly exchanged at the synchronization point since the time space between synchronization points is the largest time-step of TSAs and just integer times of other small time-steps.

IV. GPU-Based Parallel Implementation

To test and verify the advantages of the proposed GPU-based VTS parallel simulation architecture, the integrated ac–dc grid composed of one IEEE 118-bus system [20] and three MMC converter stations is selected as the case study. As shown in Fig. 3, the ac power system consists of 118 buses, 54 generators, 177 lines, 9 transformers, and 91 loads; the dc power system consists of three ac–dc converter stations connected via dc transmission lines.

A. Detailed Equipment Models

For the dc grid, the common modular multilevel converter (MMC) structure [21] is built based on the half-bridge submodule (HBSM) that consists of two IGBT switches and a capacitor. In this work, both the system-level and device-level models are utilized and the phase-disposition sinusoidal pulsewidth modulation method [22] is adopted in this work for attaining desired voltage and power flow characteristics.

In the system-level model, the IGBT and diode switching transients are ignored while only the electrical model is presented. The terminal and internal dynamics of an individual HBSM are equivalenced to a cascaded resistance and voltage source, and the

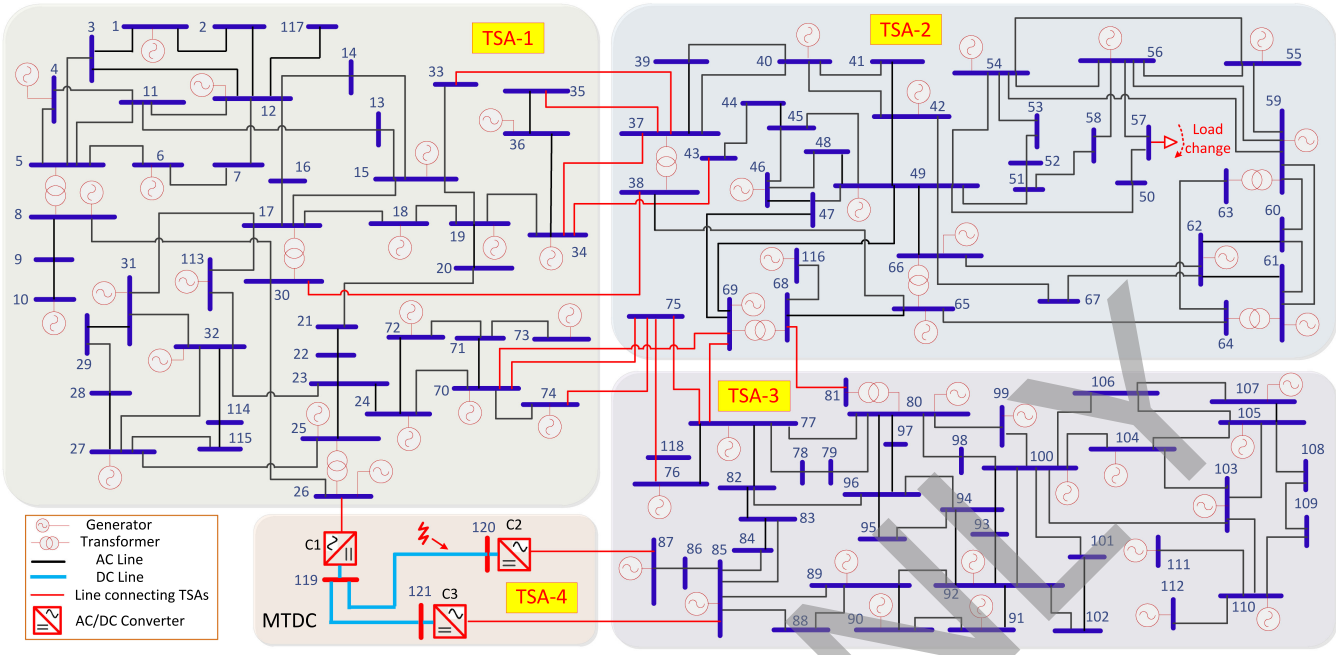

Fig. 3. Topology of the ac–dc grid test case with TSAs.

value depends on the switch state. Then, the system-level equivalent conductance matrix and current injection can be updated accordingly. The detailed equations for the equivalent circuit can be found in [23] and [24]. The device-level model focuses on the switching transients during turn-ON and turn-OFF operations and, therefore, requires smaller time-steps. The switches within each HBSM are equivalent to voltage sources or current sources depending on turn-ON or turn-OFF operations, of which the value can be obtained through curve-fit process of the known IGBT and diode user guide [25]. Then, the equivalent conductance matrix and current injection for MMC can be computed and the node voltage can be solved.

The phase-domain frequency-dependent transmission line model is utilized, i.e., the universal line model (ULM) [26]. The detailed model representation can be found in [27]. The Thévenin voltage source model [28] obtained by trapezoidal rule (TR) discretization is utilized. The AC4A type exciter control model [29] is attached with the machine to provide a feedback for the field voltage. The nonlinear transformer model is used to guarantee accuracy by combining the linear conductance matrix model [30] and the nonlinear compensation method of PSCAD/EMTDC [29]. The compensation method is used to account for the nonlinear part of the model, in which the core saturation is considered and controlled through a compensating current source injection across selected winding terminals and the value is determined by the nonlinear $\Psi - I$ curve function.

B. Implementation on GPU

The GPU device used in this article is the NVIDIA Tesla V100 GPU featured with 5120 cores, 16 GB HBM2 memory and a memory path with bandwidth of 900 GB/s. The V100 GPUs 7.0 computation capability enables the application of DP, and the large number of cores allows the utilization of detailed equipment models and massively parallel calculation of large-scale EMT simulation. Using such an architecture, the hybrid ac–dc test system can be mapped and computed efficiently.

System partition: To decompose the system into TSAs, the following two problems are involved: 1) determining the number of TSAs; 2) partitioning the topology given the number of TSAs. The number of TSAs is configured by users, for example, in the test system in Fig. 3, the 118-bus is partitioned into 3 TSAs to demonstrate the hierarchical time-step control scheme. But it can also be configured as just one TSA, if only distinguishing the ac system from the dc system. Once the number of TSAs is determined, how to partition the topology is an optimization problem when taking the synchronization latency into account. Although the partitioning for the case study in Fig. 3 is performed manually for simplicity, minimizing the connection links between different TSAs may be a good optimization goal if automatic partition algorithm is exploited. Based on the abovementioned discussion, the hybrid ac–dc grid is first decomposed into four TSAs to apply the proposed hierarchical VTS method on the GPU cores, where the 118-bus system is decomposed into three TSAs and the dc system is separated as one TSA. Every TSA is only connected with two adjacent TSAs to reduce the data exchange. Then, each TSA is decomposed into small SSs based on the connected transmission lines since the two ports of the transmission line can be calculated separately. The principle of SS decomposition is decomposing the system as fine-grained as possible according to the abundant GPU cores. Therefore, if there are transmission lines connecting SSs, then these SSs could be decomposed for parallel computing.

After decomposition, TSA-1 contains 42 SSs (45 buses), TSA-2 contains 30 SSs (35 buses), TSA-3 contains 37 SSs (38 buses), and TSA-4 contains 3 SSs. The SSs in ac side have

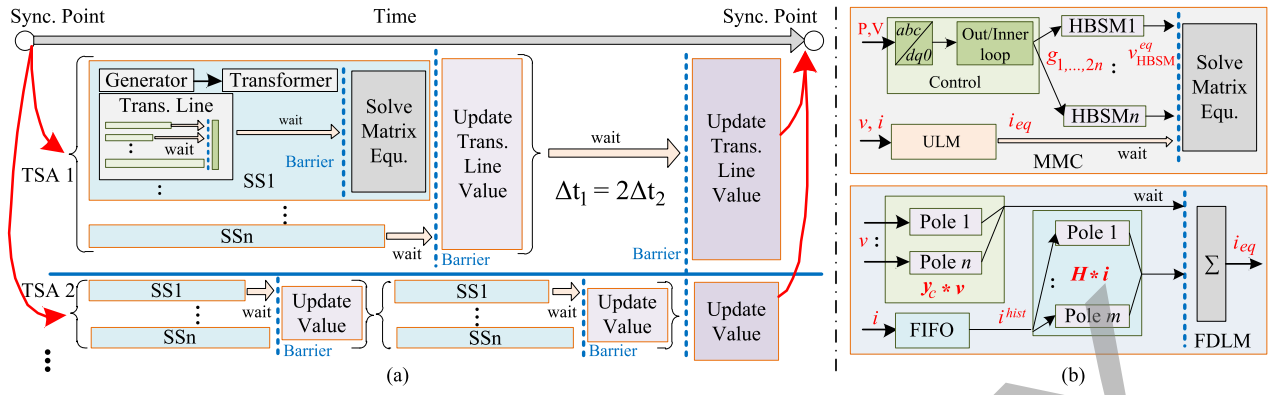

Fig. 4. GPU implementation. (a) Detailed parallel processing on GPU. (b) Parallel computing for MMC and ULM.

TABLE I
APPLICATION OF DP AND CORES USED IN EACH LEVEL

DP	TSA-1 (N_c)	TSA-2	TSA-3	TSA-4
2^{nd}-DP/TSA	$\surd$ (42)	$\surd$ (30)	$\surd$ (37)	$\surd$ (3)
3^{rd}-DP/SS	optional	optional	optional	$\surd$ (97)
4^{th}-DP/Device	optional	optional	optional	optional

simple equipment and small matrix (6 × 6 in maximum) to solve; however, in TSA-4, each SS refers to an MMC converter with dc transmission line connections, which involves heavy computational task of the equivalent circuit calculation, value-level switch control, and system-level power flow control.

Processing hierarchy: The processing hierarchy is illustrated in Fig. 4(a). The first-level kernel function run the TSA functions in parallel, and at the synchronization point the TSAs exchange the transmission line data for consistence. The application of DP in 2–4 level function is decided by evaluating the processing time over equation (6), and is shown in Table I. The number of used cores N_c in each level is also listed, note that the four grids are generated in the first level parallelism as there are four TSAs divided. For the second-level TSA-1–3 function, K (number of SSs) is so large that the parallel processing will benefit more and, thus, DP is necessary; for the TSA-4 function although K (=3) is small, the long processing time of each SS makes DP also necessary to improve the overall performance. Note that all of the parallelism in each level function ends with a barrier, at which point all the threads must reach to synchronize the data with each other. Within each SS, parallel processing is required if containing many equipment. For example, the SS composed of bus 68 and bus 69 has a coupled transformer and generator, and 7 transmission line connections. The generator-transformer pair must be executed in serial, but the transmission lines can run in parallel. However, the SS only composed of Bus 2 does not require parallel processing of equipment model because it only contain two transmission lines. Thus, the application of DP for third-level SS functions is optional in TSA-1–3, but is necessary in TSA-4 due to the huge amount of HBSMs in MMC converters. As shown in Fig. 4(b), although the control logic can not expose enough parallelism, the HBSMs should run in parallel to obtain the equivalent voltage source of each HBSM. The four-level device function refers to the computation of detailed equipment models such as the synchronous machine, transformer, ULM and HBSM, and the parallel processing is also optional because only the ULM model requires to apply DP due to the benefits by applying parallel calculation for convolution process while the other equipment models cannot run in parallel sufficiently.

Time-step size and control: As analyzed in Section II, the time-step sizes of different TSAs always belong to the same time-step set $\Delta T^{\text{sys}} = \{\Delta t_{\min} \times (2^0, 2^1, \ldots, 2^n)\}$. In this work, $n = 4$, and $\Delta t_{\min}$ is set at 10 μs for the system-level simulation. The time-step set for device-level simulation is $\Delta T^{\text{dev}} = \{0.05\ \mu\text{s}, 0.1\ \mu\text{s}, 0.2\ \mu\text{s}, 0.5\ \mu\text{s}\}$. Note that in the device-level simulation, ΔT^{dev} is only applied for MMC converters, the 118-bus system is still simulated with ΔT^{sys}. The time-step increase or decrease is determined by comparing the LTE or DVDT of the previous step and the predetermined threshold, wherein the threshold of various SSs are also different. Since the unknown variables and discretization methods applied vary between equipment models, there are different thresholds for different equipment and the time-step of a TSA will change no matter which equipment exceeds its own LTE threshold.

Typically, the threshold is determined by experience, experimental results, as well as the specific accuracy requirements. In this work, the LTE thresholds of linear elements are calculated and assigned based on the LTE (1) given the desired values of state variables; the thresholds of nonlinear elements or DVDT of MMCs are determined using the presimulation results to just demonstrate the work principle of the proposed VTS scheme. However, how to determine the threshold based on precise mathematical analysis remains a topic to be studied, and is left for future research.

V. SIMULATION RESULTS AND VERIFICATION

The advantages of the proposed GPU-based VTS simulation architecture are verified on the implemented ac–dc hybrid system and the simulation results are compared with compared with PSCAD/EMTDC and SaberRD to show the effectiveness of the applied system decomposition and detailed equipment models. The CPU simulations are conducted on the 64-b Windows10 operating system with 2.2 GHz Intel Xeon E5-2698 v4 CPU and 192 GB RAM.

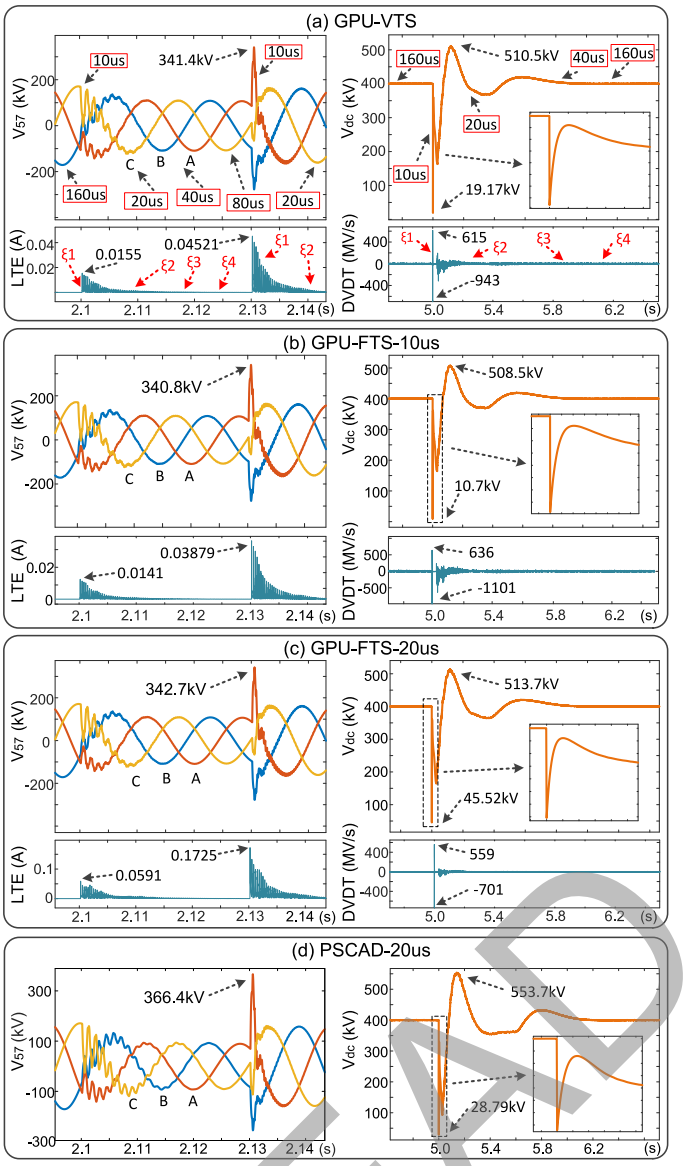

Fig. 5. Comparison between FTS and VTS simulation. (a) Results of VTS simulation on GPU. (b) and (c) GPU results with detailed equipment models and FTS (10/20 μs) applied. (d) PSCAD results with 20 μs time-step.

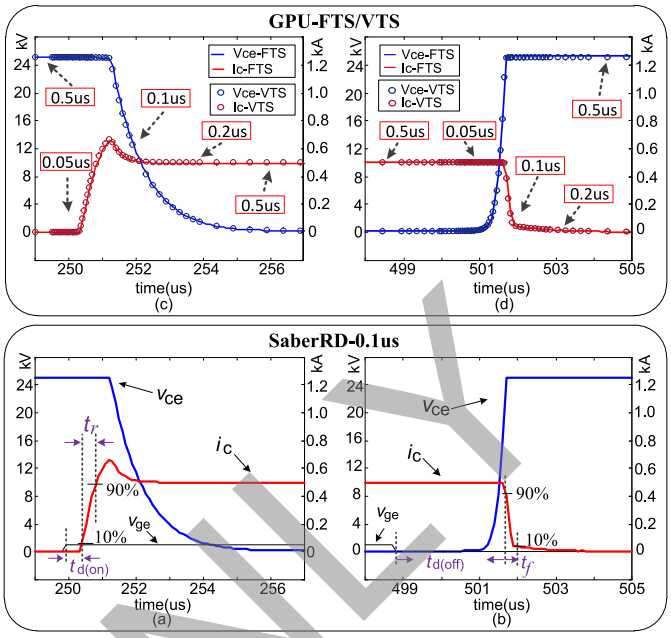

Fig. 6. Device-level switching transients. (a) and (b) Turn-ON and turn-OFF transients of GPU-based implementation with FTS and VTS. (c) and (d) Turn-ON and turn-OFF transients of SaberRD with 0.1 μs time-step.

A. Transient Results

To test the applied VTS scheme, the load-change operation at Bus 57 and ground fault at dc line $L_{119-120}$ (line between bus 119 and bus 120) are chosen as the transient test. The results are evaluated by the proposed GPU based simulator with VTSs and PSCAD/EMTDC and SaberRD that use FTSs of 20/10 μs and 0.1 μs, respectively. The load change operation and ground fault are applied at exactly 2.1 s and 5 s of the simulation, and the duration is 0.03 s.

First, the system-level transient behavior is captured and compared with PSCAD/EMTDC, as shown in Fig. 5. From the peak value and LTE, it can be observed that, using VTS scheme will result in a smaller LTE and a larger DVDT compared to FTS simulation with 20 μs time-step, and the value of LTE is close to the results with 10 μs time-step rather than that of 20 μs. Because the time-step sizes decrease to 10 μs accordingly to improve the accuracy during transient, and increase again to accelerate the simulation process under normal conditions. The difference between GPU-FTS and PSCAD/EMTDC is caused by the different generator model applied, and the TR-based machine is more stable as indicated by the lower peak values. The time-step change is based on the predetermined threshold, for example, the time-step is 160 μs before the load change, and when the transient happens, the LTE directly rises to a large value that exceed the max threshold ξ_1 immediately, then the time-step changes to the minimum one (10 μs) directly. When the LTE decreases below $\xi_{2/3/4}$, the time-step increases into a larger one (20/40/80 μs), and as the LTE is reduced below the minimum threshold ξ_5, the time-step regains to the maximum one (160 μs). The time-step change under the dc fault transient has the same process although their time-step change criteria are different and belong to different TSAs.

Second, the simulation results of device-level switching transient at the first SS of MMC C1 is also compared with SaberRD, as shown in Fig. 6. Note that this is a separate case, where the device-level IGBT model is utilized for the three MMC converters and the power equipment models in the 118-bus system remain the same as those of the system-level simulation. The four representative parameters during the turn-ON and turn-OFF switching transient are taken into concern: turn ON delay time $t_{d(\text{on})}$, rise time t_r, turn OFF delay time $t_{d(\text{off})}$, and fall time t_f. The waveforms using FTS look similar (with differences less than 2%) between GPU-based emulator and SaberRD since the curve-fit method is applied for device-level simulation. For the VTS simulation, the time-step change is triggered by the change of gate voltage v_{ge}. After the gate voltage changes, the time-step decreases to the minimum one (0.05 μs) immediately to precisely obtain the concerned parameters; and then the time-step increases gradually until the maximum size (0.5 μs).

TABLE II
EXECUTION TIME AND SPEED-UP OF DIFFERENT METHODS FOR 10 S SIMULATION

Version	FTS-CPU (Base)	VTS-CPU/Sp-CPU	VTS-DP0/Sp-0	VTS-DP1/Sp-1	VTS-DP2/Sp-2	VTS-DP3/Sp-3	VTS-DP4/Sp-4
System-Level	641.4 s	82.2s/7.8	356.3s/1.8	85.5s/7.5	30.7s/20.9	1.56s/411.2	1.51s/424.8
Device-Level	4142.7s	881.4s/4.7	3452.3s/1.2	1183.6s/3.5	505.2s/8.2	20.4s/203.3	20.3s/204.1

B. Latency and Speed-Up

Generally, there are two-stage speed-ups of the GPU-based VTS simulation that should be evaluated compared to the CPU-based FTS simulation: the speed-up by applying VTS scheme on CPU, and the speed-up by conducting the four-level parallel simulation on GPU. The two-stage speed-ups of system-level and device-level simulations are all recorded in Table II, and the duration of simulation is 10 s. Note that the CPU simulation time is measured on the developed C-code program on visual studio 2018 but not on existing EMT simulation software because the c-program is more dedicated to the case study and can achieve a better performance. The time-step of FTS simulation is set at 20 μs for system-level simulation; in the device-level simulation, the time-step is set at 0.1 μs for MMC converters and 20 μs for the 118-bus system. The VTS simulation uses the VTSs assigned in Section IV(B).

Since the VTS simulation uses large time-steps to proceed under the steady-state conditions, the speed-up of the VTS simulation on CPU is nearly 8 and 5 for system-level and device-level simulation, respectively, which is fairly considerable. Parallel processing on GPU will accelerate the simulation process, however, the performance slows down significantly if only using GPU but not applying the parallel mechanism, denoted as Sp-0. The reason is that the frequency of GPU is not as high as that of the CPU. For the system-level simulation, if the first-level DP is utilized, the speed-up increases to 7.5, but the acceleration effect is still not obvious compared to CPU because only four TSAs run in parallel but each TSA still runs in serial. The second-level should introduce massive parallelism due to the numerous SSs in the ac system, but the speed-up is not very high because TSA-4 can only be decomposed into three SSs and the lowest speed of TSAs determine the overall speed-up due to the requirement of synchronization. Once the third-level DP is applied, the speed-up increases dramatically because of the parallel computation of HBSMs in MMC and device functions in TSA-1–3. The speed-up by adding the fourth-level DP is not as obvious as the second-level and third-level DP, however, it is also necessary to fully exploit the abundant GPU cores.

For device-level simulation, the speed-up is not as large as that of system-level simulation because TSA-4 consumes much more latency than TSA-1–3 then the speed-ups of TSA-1–3 cannot be revealed in the overall performance. Compared to the FTS simulation on CPU, the overall speed-ups of GPU-based VTS simulation are 424.8 and 204.1, respectively, for system-level and device-level simulation.

VI. CONCLUSION

For the EMT simulation of hybrid ac–dc networks that contain various power system equipment with widely different time constants, system decomposition, and variable time-stepping are crucial methods to accelerate the simulation process. Taking advantage of the emerging DP feature of GPU programming, the four-level hierarchical VTS simulation architecture is proposed, in which each level can be launched as a kernel function to fully exploit the possibility of parallelism. The LTE/DVDT-based time-step control scheme is adopted, which enables accurate estimation and fast calculation of the system disturbance. The hybrid ac–dc network composed of an IEEE 118-bus system and three MMC converters is simulated on the GPU platform to verify the proposed architecture. The numerical results compared with PSCAD/EMTDC and SaberRD show the effectiveness of the applied hierarchical VTS scheme. By adopting the VTSs, the simulation process can be accelerated nearly 8 times at system-level; and by utilizing the four-level DP, the overall speed-ups of 424.8 and 204.1 for system-level and device-level simulation, respectively, are achieved. The proposed hierarchical VTS simulation architecture can be used for fast simulation of large-scale ac–dc systems that consist of various types of elements with different requirements of accuracy. Since it is a common architecture based on the parallel platforms, it can be expected to be applied in commercial simulators or to be used for customized analysis by research groups. In the future work, the more complicated nonlinear device-level models for conventional energy conversion equipment as well as MMCs will be taken into consideration.

APPENDIX

Parameters of the test system: Base values: 100 MVA, 230 kV, 60 Hz; Synchronous generator and loads: the same as [20]; Transformers: 230 kV/230 kV, positive sequence leakage inductance 0.2 p.u., copper loss 0.004 p.u., knee voltage (saturation) 1.17 p.u., magnetizing current (saturation) 2%; MMC: 17-level, $V_{\rm dc} = 400$ kV, $C_{\rm sm} = 2.5$ mF, $f_C = 2000$ Hz, $N = 16$, $L_{\rm arm} = 0.0189$ H; device-level MMC: $t_{d,\rm on} = 0.4\,\mu$s, $t_r = 0.45\,\mu$s, $t_{d,\rm off} = 3.0\,\mu$s, $t_f = 0.3\,\mu$s; load change at bus 57: $L = 0.1$ H, $R = 600\,\Omega$ change to $R = 100\,\Omega$ between 2.1 and 2.13 s.

REFERENCES

[1] B. Wu, *High-Power Converters and AC Drives*. Hoboken, NJ, USA: Wiley, 2006, pp. 3–13.
[2] J. J. Sanchez-Gasca *et al.*, "Extended-term dynamic simulation using variable time step integration," *IEEE Comput. Appl. Power*, vol. 6, no. 4, pp. 23–28, Oct. 1993.
[3] S. Hui, K. Fung, M. Zhang, and C. Christopoulos, "Variable time step technique for transmission line modeling," *IEEE Proc. A - Sci., Measurement Technol.*, vol. 140, no. 4, pp. 299–302, Jul. 1993.
[4] J. J. Sanchez-Gasca, R. D'Aquila, W. W. Price, and J. J. Paserba, "Variable time step, implicit integration for extended-term power system dynamic simulation," in *Proc. IEEE Power Ind. Comput. Appl. Conf.* , May 1995, pp. 183–189.

[5] N. Lin and V. Dinavahi, "Variable time-stepping modular multilevel converter model for fast and parallel transient simulation of multiterminal DC grid," *IEEE Trans. Ind. Electron.*, vol. 66, no. 9, pp. 6661–6670, Sep. 2019.
[6] Z. Shen and V. Dinavahi, "Dynamic variable time-stepping schemes for real-time FPGA-based nonlinear electromagnetic transient emulation," *IEEE Trans. Ind. Electron.*, vol. 64, no. 5, pp. 4006–4016, Jan. 2017.
[7] 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.
[8] J. K. Debnath, A. M. Gole, and W. K. Fung, "Graphics-processing-unitbased acceleration of electromagnetic transients simulation," *IEEE Trans. Power Del.*, vol. 31, no. 5, pp. 2036–2044, Oct. 2016.
[9] Z. Zhou and V. Dinavahi, "Fine-grained network decomposition for massively parallel electromagnetic transient simulation of large-scale power systems," *IEEE Power Energy Technol. Syst. J.*, vol. 4, no. 3, pp. 51–64, Sep. 2017.
[10] Y. Song, Y. Chen, S. Huang, Y. Xu, Z. Yu, and J. R. Marti, "Fully GPU-based electromagnetic transient simulation considering large-scale control systems for system-level studies," *IET Gener. Transmiss. Dis.*, vol. 11, no. 11, pp. 2840–2851, Aug. 3, 2017.
[11] S. Yan, Z. Y. Zhou, and V. Dinavahi, "Large-scale nonlinear device-level power electronic circuit simulation on massively parallel graphics processing architectures," *IEEE Trans. Power Electron.*, vol. 33, no. 6, pp. 4660–4678, Jun. 2018.
[12] Y. Song, Y. Chen, S. Huang, Y. Xu, Z. Yu, and W. Xue, "Efficient GPU-based electromagnetic transient simulation for power systems with thread-oriented transformation and automatic code generation," *IEEE Access*, vol. 6, pp. 25724–25736, 2018.
[13] N. Lin and V. Dinavahi, "Parallel high-fidelity electromagnetic transient simulation of large-scale multi-terminal DC grids," *IEEE Power Energy Technol. Syst. J.*, vol. 6, no. 1, pp. 59–70, Mar. 2019.
[14] N. Lin and V. Dinavahi, "Exact nonlinear micro-modeling for fine-grained parallel EMT simulation of MTDC grid interaction with wind farms," *IEEE Trans. Ind. Electron.*, vol. 66, no. 8, pp. 6427–6436, Aug. 2019.
[15] D. Shu, X. Xie, V. Dinavahi, C. Zhang, X. Ye, and Q. Jiang, "Dynamic phasor based interface model for EMT and transient stability hybrid simulations," *IEEE Trans. Power Syst.*, vol. 33, no. 4, pp. 3930–3939, Jul. 2018.
[16] D. Shu, Y. Wei, V. Dinavahi, K. Wang, Z. Yan, and X. Li, "Cosimulation of shifted-frequency/dynamic phasor and electromagnetic transient models of hybrid LCC-MMC DC grids on integrated CPU-GPUs," *IEEE Trans. Ind. Electron.*, vol. 67, no. 8, pp. 6517–6530, Aug. 2020.
[17] NVIDIA Corporation, *Dynamic Parallelism in Cuda.* New York, NY, USA: Elsevier, Jun. 2018.
[18] F. N. Najm, *Circuit Simulation.* Hoboken, NJ, USA: Wiley, 2010, pp. 241–252.
[19] C. Deml, and P. Turkes, "Fast simulation technique for power electronic circuits with widely different time constants," *IEEE Trans. Ind. Appl.*, vol. 35, no. 3, pp. 657–662, May 1999.
[20] *PSCAD IEEE 118 Bus System*, Revision 1, Manitoba HVdc Research Centre, Winnipeg, MB, Canada, 2010.
[21] A. Lesnicar and R. Marquardt, "An innovative modular multilevel converter topology suitable for a wide power range," in *Proc. IEEE Bologna Power Tech Conf.*, Bologna, Italy, 2003, pp. 1–6.
[22] G. Carrara, S. Gardella, M. Marchesoni, R. Salutari, and G. Sciutto, "A new multilevel PWM method: A theoretical analysis," *IEEE Trans. Power Electron.*, vol. 7, no. 3, pp. 497–505, Jul. 1992.
[23] Q. Song, W. Liu, X. Li, H. Rao, S. Xu, and L. Li, "A steady-state analysis method for a modular multilevel converter," *IEEE Trans. Power Electron.*, vol. 28, no. 8, pp. 3702–3713, Aug. 2013.
[24] S. Liu, Z. Xu, W. Hua, G. Tang, and Y. Xue, "Electromechanical transient modeling of modular multilevel converter based multi-terminal HVDC systems," *IEEE Trans. Power Syst.*, vol. 29, no. 1, pp. 72–83, Jan. 2014.
[25] C. Liu *et al.*, "FPGA-based real-time simulation of high-power electronic system with nonlinear IGBT characteristics," *IEEE J. Emerg. Sel. Topics Power Electron.*, vol. 2, no. 4, pp. 1109–1116, Dec. 2018.
[26] A. Morched, B. Gustavsen, and M. Tartibi, "A universal model for accurate calculation of electromagnetic transients on overhead lines and underground cables," *IEEE Trans. Power Del.*, vol. 14, no. 3, pp. 1032–1038, Jul. 1999.
[27] 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.
[28] 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.
[29] *EMTDC User's Guide: A Comprehensive Resource for EMTDC*, Version 4.7, Manitoba HVdc Research Centre, Winnipeg, MB, Canada, 2010.
[30] V. Brandwajn, H. W. Dommel, and I. Dommel, "Matrix representation of three-phase n-winding transformers for steady-state and transient studies," *IEEE Trans. Power Appl. Syst.*, vol. 101, no. 6, pp. 1369–1378, Jun. 1982.

Tong Duan (Student Member, IEEE) received the B.Eng. degree in electronic engineering from Tsinghua University, Beijing, China, in 2013. He is currently working toward Ph.D. degree in electrical and computer engineering with the University of Alberta, Edmonton, AL, 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 from the Visveswaraya National Institute of Technology (VNIT), Nagpur, India, in 1993, the M.Tech. degree from the Indian Institute of Technology (IIT) Kanpur, Kanpur, India, in 1996, both in electrical engineering, and the Ph.D. degree in electrical and computer engineering from the University of Toronto, Toronto, ON, Canada, in 2000.

He is currently 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.